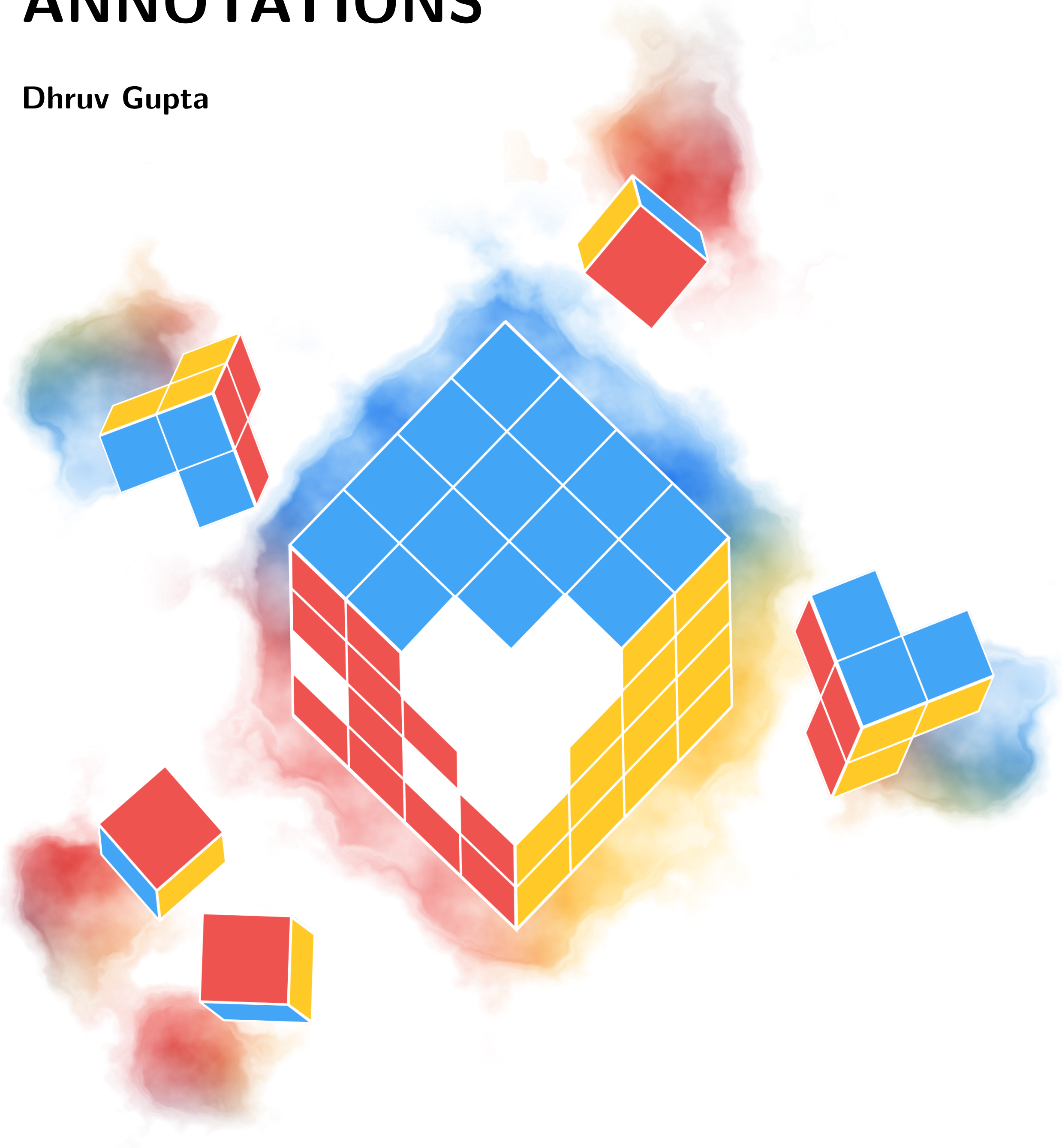
SEARCH AND ANALYTICS USING SEMANTIC ANNOTATIONS

Dhruv Gupta



SEARCH AND ANALYTICS USING SEMANTIC ANNOTATIONS

Dhruv Gupta

Dissertation
zur Erlangung des Grades
Doktor der Ingenieurwissenschaften (Dr.-Ing.)
der Fakultät für Mathematik und Informatik
der Universität des Saarlandes

Saarbrücken
2019

EXAMINATION BOARD

To my parents,
my brother,
and
all my teachers

*“Whenever you are in doubt, or when the self becomes too much with you ...
Recall the face of the poorest and weakest man whom you may have seen, and ask yourself, if the step you contemplate is going to be of any use to him ...
Then you will find your doubts and your self melt away.”*

— MAHATMA GANDHI

ABSTRACT

Search systems help users locate relevant information in the form of text documents for keyword queries. Using text alone, it is often difficult to satisfy the user's information need. To discern the user's intent behind queries, we turn to semantic annotations (e.g., named entities and temporal expressions) that natural language processing tools can now deliver with great accuracy. This thesis develops methods and an infrastructure that leverage semantic annotations to efficiently and effectively search large document collections.

This thesis makes contributions in three areas: indexing, querying, and mining of semantically annotated document collections. First, we describe an indexing infrastructure for semantically annotated document collections. The indexing infrastructure can support knowledge-centric tasks such as information extraction, relationship extraction, question answering, fact spotting and semantic search at scale across millions of documents. Second, we propose methods for exploring large document collections by suggesting semantic aspects for queries. These semantic aspects are generated by considering annotations in the form of temporal expressions, geographic locations, and other named entities. The generated aspects help guide the user to relevant documents without the need to read their contents. Third and finally, we present methods that can generate events, structured tables, and insightful visualizations from semantically annotated document collections.

KURZFASSUNG

Suchmaschinen helfen Nutzern, relevante Informationen in Textdokumenten zu finden. Alleine auf Grundlage des textuellen Inhalts der Dokumente ist es häufig schwierig, die Informationsbedürfnisse der Nutzer zu stillen. Um diese Informationsbedürfnisse besser zu verstehen, lassen sich semantische Annotationen der Dokumente (z.B. Zeitausdrücke und Entitäten) ausnutzen, die von heutigen Methoden der natürlichen Sprachverarbeitung mit hoher Genauigkeit erzeugt werden können. Diese Dissertation beschreibt eine Infrastruktur zur Indexierung semantisch annotierter Dokumente, um so auch in großen Dokumentensammlungen effektiv und effizient suchen zu können.

Die Beiträge dieser Dissertation lassen sich in Methoden zur Indexierung, Anfragebearbeitung und Analyse von semantisch annotierten Dokumentensammlungen gliedern. Zuerst wird eine Infrastruktur zur Indexierung semantisch annotierter Dokumente beschrieben. Diese Infrastruktur kann verschiedene wissensorientierte Aufgaben (z.B. die automatische Extraktion von strukturierten Informationen) auch auf Dokumentensammlungen mit Millionen von Dokumenten effizient unterstützen. Danach wird ein Verfahren vorgestellt, welches es erlaubt, große Dokumentensammlungen anhand semantischer Annotationen zu durchsuchen. Hierzu extrahiert das Verfahren semantische Aspekte als Kombinationen semantischer Annotationen (z.B. Zeit- und Ortsausdrücke sowie andere Entitäten). Diese semantischen Aspekte erlauben es den Benutzern, direkt zu relevanten Dokumenten zu gelangen, ohne vorab deren Inhalt zu kennen. Zuletzt wird ein Verfahren beschrieben, welches es erlaubt, automatisch Ereignisse, strukturierte Tabellen und Visualisierungen aus semantisch annotierten Dokumentensammlungen zu generieren.

SUMMARY

Current information retrieval systems are limited to text in documents for helping users with their information needs. With the progress in the field of natural language processing, there now exists the possibility of enriching large document collections with accurate semantic annotations. Annotations in the form of part-of-speech tags, temporal expressions, numerical values, geographic locations, and other named entities can help us look at terms in text with additional semantics. This doctoral dissertation presents methods for search and analysis of large semantically annotated document collections. Concretely, we make contributions along three broad directions: indexing, querying, and mining of large semantically annotated document collections. In the following paragraphs, we outline the key contributions made along these three research directions.

Indexing Annotated Document Collections. Knowledge-centric tasks such as information extraction, question answering, and relationship extraction require a user to retrieve text regions within documents that detail relationships between entities. Current search systems are ill-equipped to handle such tasks, as they can only provide phrase querying with Boolean operators. To enable knowledge acquisition at scale, we propose GYANI, an indexing infrastructure for knowledge-centric tasks. GYANI enables search for structured query patterns by allowing regular expression operators to be expressed between word sequences and semantic annotations. To implement GREP-like search capabilities over large annotated document collections, we present a data model and index design choices involving word sequences, annotations, and their combinations. We show that by using our proposed indexing infrastructure we bring about drastic speedups in crucial knowledge-centric tasks: 95× in information extraction, 53× in question answering, and 12× in relationship extraction.

Hyper-phrase queries are multi-phrase set queries that naturally arise when attempting to spot knowledge graph facts or subgraphs in large document collections. An example hyper-phrase query for the fact ⟨MAHATMA GANDHI, NOMINATED FOR, NOBEL PEACE PRIZE⟩ is: ⟨{*mahatma gandhi*, *m k gandhi*, *gandhi*}, {*nominated for*, *nominee for*, *nomination received*}, {*nobel peace prize*, *nobel prize for peace*, *nobel prize in peace*}⟩. Efficient execution of hyper-phrase queries is of essence when attempting to verify and validate claims concerning named entities or emerging named entities. To do so, it is required that the fact concerning the entity can be contextualized in text. To acquire text regions given a hyper-phrase query, we propose a retrieval framework using combinations of n-gram and skip-gram indexes. Concretely, we model the combinatorial space of the phrases in the hyper-phrase query to be retrieved using vertical and horizontal operators and propose a dynamic programming approach for optimized query processing. We show that using our proposed optimizations we can retrieve sentences in support of knowledge graph facts and subgraphs from large document collections within seconds.

Querying Annotated Document Collections. Users often struggle to convey their information needs in short keyword queries. This often results in a series of query reformulations, in an attempt to find relevant documents. To assist users navigate large document collections and lead them to their information needs with ease, we propose methods that leverage semantic annotations. As a first step, we focus on temporal information needs. Specifically, we leverage temporal expressions in large document collections to serve time-sensitive queries better. Time-sensitive queries, e.g., *summer olympics* implicitly carry a temporal dimension for document retrieval. To help users explore longitudinal document collections, we propose a method that generates time intervals of interest as query reformulations. For instance, for the query *world war*, time intervals of interest are: $[1914, 1918]$ and $[1939, 1945]$. The generated time intervals are immediately useful in search-related tasks such as temporal query classification and temporal diversification of documents.

Time-sensitive queries can be classified into classes that can help serve their information needs better. For instance, the event described by the query *summer olympics* is periodically recurring at the year granularity, and thus the next event and its related information should be retrieved for the user. To classify temporal queries at different levels of granularity we perform Bayesian analysis of the distribution of generated time intervals for multi- modality. With the help of features designed using the Bayesian analysis, we show that we can accurately classify queries into temporal classes at different granularities.

History-oriented queries concerning entities and events require that the presented list of documents cover the history of the entity or describe the

timeline of the event. To temporally diversify documents, we re-rank the pseudo-relevant set of documents such that there is at least one document that refers to a generated time interval of interest for the query. We show that using our approach we can indeed present a diversified set of documents that reflect summaries corresponding to history-oriented queries.

As a second and final step, we focus on helping the user in navigating large document collections by generating semantic aspects. The aspects are generated using semantic annotations in the form of temporal expressions, geographic locations, and other named entities. Concretely, we propose the xFACTOR algorithm that generates semantic aspects in two steps. In the first step, xFACTOR computes the salience of annotations in models informed of their semantics. Thus, the temporal expressions ***1930s*** and ***1939*** are considered similar as well as entities such as USAIN BOLT and JUSTIN GATLIN are considered related when computing their salience. Second, the xFACTOR algorithm computes the co-occurrence salience of annotations belonging to different types by using an efficient partitioning procedure. For instance, the aspect $\langle\{$USAIN BOLT$\}$, $\{$BEIJING, LONDON$\}$, $[2008, 2012]\rangle$ signifies that the entity, locations, and the time interval are observed frequently in isolation as well as together in the documents retrieved for the query ***olympic medalists***.

Mining Annotated Document Collections. Large annotated document collections are a treasure trove of historical information concerning events and entities. We leverage annotations present in documents to determine time intervals of interest for knowledge graph facts; to mine events; to generate structured tables; and to generate analytical visualizations. First, we present an approach to determine a ranked list of time intervals of interest for knowledge graph facts. For instance, for the knowledge graph fact $\langle$MAHATMA GANDHI, NOMINATION, NOBEL PEACE PRIZE$\rangle$, the time intervals of interest are: $[1937, 1937]$, $[1938, 1938]$, $[1939, 1939]$, $[1947, 1947]$, and $[1948, 1948]$. To determine the time intervals, our approach first retrieves sentences containing the natural language representation of the fact with temporal expressions. We then aggregate the retrieved temporal evidence in an uncertainty-aware time model to generate relevant time intervals for the knowledge graph fact. We show the effectiveness of our approach over a benchmark of facts derived from Freebase and DBpedia.

Second, we present EVENTMINER, a clustering algorithm, that mines events for keyword queries by using annotations in the form of temporal expressions, geographic locations, and other disambiguated named entities present in a pseudo-relevant set of documents. EVENTMINER aggregates the annotation evidences by mathematically modeling their semantics. Temporal expressions are modeled in an uncertainty and proximity-aware time model. Geographic locations are modeled as minimum bounding rectangles over their geographic coordinates. Other disambiguated named entities are modeled as a set of links corresponding to their Wikipedia articles. For a set of history-oriented queries concerning entities and events, we show that our approach can truly identify event clusters when compared to approaches that disregard annotation semantics.

Third, we present JIGSAW, an end-to-end query driven system that generates structured tables for user-defined schema from unstructured text. To define the table schema, we describe query operators that help perform structured search on annotated text and fill in table cell values. To resolve table cell values whose values can not be retrieved, we describe methods for inferring NULL values using local context. JIGSAW further relies on semantic models for text and numbers to link together near-duplicate rows. This way, JIGSAW is able to piece together paraphrased, partial, and redundant text regions retrieved in response to structured queries to generate high-quality tables within seconds.

Fourth and finally, we present DIGITALHISTORIAN, a time-sensitive search engine that allows users to navigate longitudinal document collections using keyword queries. For an implicit time-sensitive query DIGITALHISTORIAN suggests time intervals of interest as query reformulations. In addition to this, DIGITALHISTORIAN diversifies the retrieved document set so that each document covers at least one of the generated time intervals of interest. Furthermore, DIGITALHISTORIAN generates analytical visualizations in the form of chord graph and heatmap diagrams that summarize the relationships between named entities and the generated time intervals of interest.

ZUSAMMENFASSUNG

Heutige Suchmaschinen beschränken sich auf den textuellen Inhalt von Dokumenten, um die Informationsbedürfnisse der Benutzer zu stillen. Fortschritte in der natürlichen Sprachverarbeitung haben dazu geführt, dass es nun möglich ist, auch sehr große Dokumentensammlungen mit hoher Genauigkeit semantisch zu annotieren. Semantische Annotationen in Form von z. B. Wortarten, Zeit- und Ortsangaben und anderer Entitäten helfen, den Inhalt der Dokumente besser zu verstehen. In dieser Dissertation werden Verfahren zur Suche und Analyse von großen semantisch annotierten Dokumentensammlungen vorgestellt. Es werden Beiträge zur Indexierung, Anfragebearbeitung und Analyse von semantisch annotierten Dokumentensammlungen beschrieben, die im Folgenden genauer skizziert werden.

Indexierung semantisch annotierter Dokumentensammlungen. Wissensorientierte Aufgaben wie die automatische Extraktion strukturierter Informationen und Beziehungen erfordern es, Textausschnitte zu identifizieren, in denen die Beziehung zweiter Entitäten genauer beschrieben wird. Heutige Suchmaschinen unterstützen solche Aufgaben nur unzureichend, da ihre Möglichkeiten auf einfache Anfragen beschränkt sind. Um einen Wissensgewinn auch auf großen Dokumentensammlungen zu ermöglichen, wird GYANI als Infrastruktur zu Indexierung semantisch annotierter Dokumentensammlungen vorgestellt. GYANI unterstützt eine an reguläre

Ausdrücke angelehnte Anfragesprache, die es erlaubt, Anforderungen an den Text sowie die semantischen Annotationen zu kombinieren. Um eine zum Werkzeug GREP ähnliche Funktionalität zu unterstützen, vertraut GYANI auf ein Datenmodell, welches Dokumente als Sequenzen von Wörtern und semantischen Annotationen betrachtet. Zur Indexierung mit Hilfe eines invertierten Index setzt GYANI auf geschickt ausgewählte Kombinationen von Wörtern und semantischen Annotationen. Experimente zeigen, dass die vorgeschlagene Infrastruktur die Zeit zur Anfragebearbeitung für wissensorientierte Aufgaben drastisch reduzieren kann: 95× bei der Extraktion strukturierter Informationen, 53× beim Beantworten natürlichsprachiger Anfragen und 12× bei der Extraktion von Beziehungen zwischen Entitäten.

Anfragen bestehend aus mehreren Mengen von Phrasen sind nützlich, um Fakten oder Teilgraphen aus Wissensgraphen in großen Dokumentensammlungen zu finden. Um beispielsweise den Fakt ⟨MAHATMA GANDHI, NOMINATED FOR, NOBEL PEACE PRIZE⟩ zu finden, ergibt sich die folgende Anfrage: ⟨{*mahatma gandhi*, *m k gandhi*, *gandhi*}, {*nominated for*, *nominee for*, *nomination received*}, {*nobel peace prize*, *nobel prize for peace*, *nobel prize in peace*}⟩. Solche Anfragen effizient zu bearbeiten ist beispielsweise wichtig, um Fakten bezüglich von Entitäten verifizieren zu können. Hierzu ist es notwendig, Textausschnitte zu finden, in denen der Fakt genauer beschrieben wird. Um solche Textausschnitte effizient finden zu können, wird eine Infrastruktur zur Indexierung semantisch annotierter Dokumentensammlungen vorgestellt, die auf einer Kombination verschiedener Indizes beruht. Zudem wird ein Verfahren vorgestellt, welche einen geeigneten Plan zur Bearbeitung einer Anfrage mit Hilfe von dynamischer Programmierung bestimmt. Die Infrastruktur erlaubt es, selbst auf großen Dokumentensammlungen, Textausschnitte zu einem gegebenen Fakt aus einem Wissensgraph in wenigen Sekunden zu finden.

Anfragen auf semantisch annotierten Dokumentensammlungen. Benutzer haben oft Probleme, ihr Informationsbedürfnis in Form von Anfragen prägnant auszudrücken. Oft sind sie gezwungen, die Anfrage mehrfach zu ändern, um letztlich zu relevanten Dokumenten zu gelangen. Um Nutzern die Suche und das Navigieren in Dokumentensammlungen zu erleichtern, werden Verfahren vorgestellt, welche verfügbare semantische Annotationen ausnutzen. So genannte zeitliche Informationsbedürfnisse (z. B. *summer olympics*) lassen sich mit Hilfe von relevanten Zeitintervallen genauer beschreiben. Das vorgestellte Verfahren nutzt die in Dokumenten enthaltenen Zeitausdrücke aus, um so automatisch Zeitintervalle als sinnvolle Ergänzungen der ursprünglichen Anfrage vorzuschlagen. Für die Anfrage *world war*, als konkretes Beispiel, würde das Verfahren die beiden Zeitintervalle [1914, 1918] und [1939, 1945] als sinnvolle Ergänzungen vorschlagen. Die automatisch vorgeschlagenen Zeitintervalle können zudem andere Aufgaben wie die automatische Klassifikation von Anfragen oder die Diversifizierung von Anfrageergebnissen unterstützen.

Zeitliche Informationsbedürfnisse lassen sich in verschiedene Klassen einordnen. Gelingt dies, so kann ein solches Informationsbedürfnis besser gestillt werden. Stellt man z. B. fest, dass sich die Anfrage *summer olympics* auf ein periodisch wiederkehrendes Ereignis bezieht, so können für die Anfrage Dokumente identifiziert werden, die sich auf das letzte solche Ereignis beziehen. Um zeitliche Informationsbedürfnisse automatisch zu klassifizieren, wird ein Verfahren vorgestellt, welches auf einer Bayesschen Analyse der Verteilung zuvor identifizierter Zeitintervalle beruht. Mit Hilfe daraus abgeleiteter Merkmale lassen sich zeitliche Informationsbedürfnisse genauer automatisch klassifizieren, wie die durchgeführten Experimente zeigen.

Geschichtsorientierte Anfragen mit Bezug zu konkreten Entitäten oder Ereignissen erfordern, dass die zurückgelieferten Dokumente einen Überblick über die Entität oder das Ereignis im Zeitverlauf bieten. Um dies zu erreichen, wird ein Verfahren vorgestellt, welches Anfrageergebnisse automatisch anhand zuvor identifizierter Zeitintervalle diversifiziert. Ziel ist es, sicherzustellen, dass sich zu jedem der identifizierten Zeitintervalle mindestens ein relevantes Dokument im Anfrageergebnis befindet. Experimente zeigen, dass das Verfahren in der Tat zu Anfrageergebnissen führt, die einen Überblick über eine Entität oder ein Ereignis im Zeitverlauf bieten.

Das letzte der vorgestellten Verfahren zielt darauf ab, Benutzern das Navigieren großer semantisch annotierter Dokumentensammlungen zu erleichtern. Hierzu generiert es automatisch so genannte semantische Aspekte als Kombination verschiedener semantischer Annotation. Das Verfahren xFactor generiert semantische Aspekte in zwei Schritten. Im ersten Schritt werden semantische Annotationen bezüglich ihrer gegenseitigen Ähnlichkeit und Wichtigkeit für die aktuelle Anfrage bewertet. Die beiden Zeitausdrücke *1930s* und *1939* oder die beiden Entitäten USAIN BOLT und JUSTIN GATLIN würden so als ähnlich zueinander und wichtig zur Anfrage *olympic medalists* betrachtet. Im zweiten Schritt werden semantische Aspekte identifiziert, wozu xFactor Verfahren der Assoziationsanalyse geeignet anpasst. Für die vorherige Anfrage wäre solch ein semantischer Aspekt $\langle\{$USAIN BOLT$\}$, $\{$BEIJING, LONDON$\}$, $[2008, 2012]\rangle$, der anzeigt, dass die beiden Athleten im angegebenen Zeitintervall häufig gemeinsam in zur Anfrage relevanten Dokumenten gesehen wurden.

Analyse semantisch annotierter Dokumentensammlungen. Semantisch annotierte Dokumentensammlungen können helfen, historische Ereignisse sowie bestimmte Entitäten genauer zu verstehen. Es werden Verfahren vorstellt, die unter Rückgriff auf semantische Annotationen automatisch Zeitintervalle zu Fakten aus Wissensgraphen identifizieren, wichtige Ereignisse erkennen, strukturierte Tabellen generieren und Visualisierungen erzeugen.

Das erste Verfahren schlägt zu einem gegebenen Fakt aus einem Wissensgraph automatisch sinnvolle korrespondierende Zeitintervalle vor. Für den Fakt $\langle$MAHATMA GANDHI, NOMINATION, NOBEL PEACE PRIZE$\rangle$, als konkretes Beispiel, würde das Verfahren automatisch die Zeitintervalle $[1937, 1937]$, $[1938, 1938]$, $[1939, 1939]$, $[1947, 1947]$, und $[1948, 1948]$ vorschlagen. Um dies zu erreichen, identifiziert das Verfahren zunächst Sätze, in denen der gegebene Fakt beschrieben wird. Daraufhin werden in diesen Sätzen gefundene Zeitausdrücke analysiert, um so sinnvolle Zeitintervalle zu identifizieren. Experimente zeigen, dass das Verfahren für Fakten aus den Wissensgraphen Freebase und DBpedia in der Tat sinnvolle Zeitintervalle identifizieren kann.

Das zweite Verfahren EventMiner zielt darauf ab, automatisch wichtige Ereignisse anhand einer gegebenen Anfrage zu identifizieren. Es beruht auf einer Cluster-Analyse der zur Anfrage relevanten Dokumente und nutzt hierbei die enthaltenen semantischen Annotationen aus. Jede Art semantischer Annotationen wird hierbei eigenes derart modelliert, dass das Verfahren die Bedeutung der Annotationen erfassen kann. Zeitausdrücke werden als Mengen von Zeitintervallen modelliert; Ortsangaben als Mengen von geographischen Koordinaten; sonstige Entitäten als Mengen von URLs, die sich auf dem zugehörigen Artikel in der Wikipedia finden. Experimente mit einer ausgewählten Menge von historisch orientierten Anfragen zeigen, dass das Verfahren in der Lage ist, automatisch wichtige Ereignisse zu identifizieren.

Das dritte Verfahren jigsaw erlaubt es, automatisch anhand einer gegebenen strukturierten Anfrage eine Tabelle mit strukturierten Daten zu erzeugen. Als Teil der Anfrage wird das gewünschte Schema der Tabelle angegeben. jigsaw identifiziert dann Textausschnitte, die zur Anfrage passen und semantische Annotationen beinhalten. Diese semantischen Annotationen werden im Folgenden analysiert, um so mögliche Zeilen der Tabelle zu identifizieren. Diese Zeilen werden in einem nächsten Schritt von jigsaw weiterverarbeitet, so dass zueinander ähnliche Zeilen gruppiert und sinnvoll zusammengefasst werden. So ist jigsaw in der Lage, in wenigen Sekunden zu einer gegebenen strukturierten Anfrage eine Tabelle von hoher Qualität zu erzeugen.

DigitalHistorian als letztes der vorgestellten Systeme ist eine Suchmaschine für zeitliche Informationsbedürfnisse. Benutzer können damit langzeitliche Dokumentensammlungen (z. B. Zeitungsarchive) mit Hilfe von Anfragen durchsuchen und die Anfrageergebnisse explorieren. Für Informationsbedürfnisse ohne expliziten Zeitbezug schlägt DigitalHistorian automatisch sinnvolle Zeitintervalle als Ergänzungen der Anfrage vor. Zudem ist DigitalHistorian in der Lage, Anfrageergebnisse automatisch zu diversifizieren, so dass Benutzer einen Überblick über ein Ereignis oder eine Entität im Zeitverlauf erhalten. Visualisierungen der in relevanten Dokumenten enthaltenen semantischen Annotation in Form von Chord-Graphen und Heatmaps helfen Benutzern, einen Überblick über die Menge der relevanten Dokumente zu erhalten.

CONTENTS

CHAPTER 1

INTRODUCTION

Information retrieval (IR) systems help users locate relevant information from vast amounts of text present in large document collections, e.g., news and web archives. To this end, IR systems have largely relied on terms in text to serve users' information needs. With significant progress in the field of natural language processing (NLP), it is now possible to process large document collections with accurate semantic annotations in the form of part-of-speech (POS) tags, named entities, temporal expressions, and numerical values. Such semantic annotations help impose a lexico-syntactic structure on unstructured text. In addition to this, annotations give deeper semantics to terms in text, allowing us to interpret the words in text correctly. With the aid of semantic annotations, it becomes possible to serve the information need of the user with relevant documents. In this chapter, we describe the state of the art that exists in the context of leveraging semantic annotations present in documents, we then sketch the research problems this thesis addresses, the publications produced as part of this doctoral dissertation, and an outline for the remainder of the thesis.

1.1 State of the Art

We next survey the progress already made in the area of analyzing semantically annotated document collections.

Important Events in Annotated Corpora

One of the most important works in identifying existing and emerging events are the tasks defined in the Topic Detection and Tracking (TDT) study [28]. The TDT program aims to "search, organize, and structure" broadcast news media from multiple sources. The five tasks defined in TDT are topic tracking, link detection, topic detection, first story detection, and story segmentation. The first task, topic tracking task, requires the participants to build a system to detect on topic stories from an evaluation corpus after being trained on a pre-defined set of topics. The second task, link detection task, involves answering a Boolean query to decide whether two given stories are related by a common topic. The third task, topic detection task, requires declaring new topics from incoming stories that have not been presented to the system. The fourth task, first story detection task, is another Boolean decision task to determine whether a given story can be used to create a new topic cluster. The fifth and final task, story segmentation task, requires converting an incoming stream of text into stories.

Focusing specifically on extracting and summarizing events in the future, Jatowt and Yeung [124] present a model-based clustering algorithm. The clustering method considers both textual and temporal similarities. For computing temporal similarity, the authors model time using the appropriate families of distributions that are decided based on whether the temporal expression is a singular time point, a starting date, or an ending date. The similarity between two temporal distributions is then computed using Kullback-Leibler divergence. Radinsky et al. [176] present a query-driven system, Pundit, which is able to predict future events based on past events reported in news archives. An event is modeled by time, geographic location, and participating entities. The algorithm derives these events from external text collection and builds an abstraction tree using hierarchical agglomerative clustering. In order to predict the future, Pundit is trained to select the most relevant cluster, as an event, from the abstraction tree with respect to the input query.

Yeung and Jatowt [207] analyze historical events with the help of Latent Dirichlet Allocation (LDA). They use LDA to identify topics and their distributions along time. Thereafter, they perform analysis on the topic distributions to answer questions such as significant years and topics, triggers that caused remembrance of the past, and historical similarity between countries. Abujabal and Berberich [24] present, P^2F Miner, a system which identifies important events in document collections by counting frequent itemsets of sentences containing named entities and temporal expressions. For evaluation, the authors use Wikipedia's event directory as ground truth.

Temporal Information Retrieval and Extraction

Comprehensive overview of advances in the field of temporal information retrieval has been covered by Kanhabua et al. [130] and Campos et al. [53]. Existing work in temporal information retrieval has largely considered only publication dates associated with documents to improve retrieval effectiveness. To this end, there have been works using document publications dates to analyze time-sensitive queries [127, 132], for re-ranking documents [70], and for diversifying documents [40]. One of the seminal works in extraction of temporal events was by Ling and Weld [147]. They outline a probabilistic model to solve the problem of extracting relations from text with temporal constraints. Generating timeline of events is yet another important temporal information extraction task. Swan and Allan [191] present an approach for producing timelines that depicts most important topics and events. Their algorithm analyzes features based on named entities and noun phrases. The analysis first constructs a 2×2 contingency table indicating presence or absence of features. They then measure the χ^2 statistic to establish whether a pair of features co-occur significantly. Baeza-Yates [33] propose a future retrieval (FR) system that considers both text and temporal expressions to identify future events relevant to an input query. Baeza-Yates outlined the components of a FR system to be composed of an information extraction (IE) module, information retrieval (IR) module, and a text mining (TM) module. The IE module would act as a temporal annotator to identify and normalize temporal expressions. The IR system is designed to incorporate the time dimension in an index, thus retrieving documents with text and time similarity. The TM module would identify the most relevant topics given a time period. He then presented a retrieval model, in which each document consists of multiple temporal events. A temporal event consists of a time segment and its associated likelihood of occurring in the future. The score of the document is obtained by its textual similarity and the maximum likelihood of all the temporal events in that document.

Geographic Information Retrieval

Events as a means of exploring document collections has also been proposed by Strötgen and Gertz [187]. Events are modeled by their geographic location and time of their occurrence. For temporal queries expressed in simple natural language, they outline an extended Backus-Naur form (EBNF) language that incorporates time intervals with standard Boolean operations. Geographic queries can also be modeled in their EBNF language, however the input for them is a minimum bounding rectangle (MBR). Using this multidimensional querying model, the user is able to visualize search results in the form of events that are represented on a map. Giving special attention to geographical information retrieval, Samet et al. [180] present a system NewsStand, that is able to resolve and pinpoint a news article based on the geographic information present in its content. To do so, they discuss methods for toponymn resolution that disambiguate the geographic location based on its surface form in news content. The system involves a streaming clustering algorithm that keeps track of emerging news in new locations and presents them on a map.

Semantic Search

Balog [34], gives an extensive overview of entity-oriented search and related tasks. By disambiguating and linking named entities to knowledge graphs, Hoffart et al. [116, 117] provide a framework for semantic search and performing entity-centric analytics. The system provides features such as query reformulations in the form of similar entities and visualizations that depict entity counts and co-occurrence statistics. Bast and Buchhold [36] outline a joint index structure over knowledge graphs and text. Their indexing infrastructure allows for fast semantic search and also provides context sensitive auto-complete suggestions.

1.2 The Big Picture

This thesis is built around the following central hypothesis:

Given semantically annotated document collection, traditional information retrieval models can be improved by utilizing knowledge about semantic annotations and using them as proxies for information need.

We leverage semantic annotations present in documents in three ways. First, the lexico-syntactic structure imposed by semantic annotations on text helps us intelligently index and query annotated document collections for knowledge-centric tasks (e.g., information extraction, relation extraction, and question answering). Second, by leveraging the semantics underlying the annotations, we can help users explore large document collections by suggesting query reformulations in the form of semantic aspects. Third, by combining multiple semantic annotations (e.g., named entities, geographic locations, and temporal expressions), interesting insights (e.g., in the form of events) can be mined from large document collections. To illustrate these contributions, consider the example annotated documents in Figures 1.1 and 1.2 for the remainder of the section.

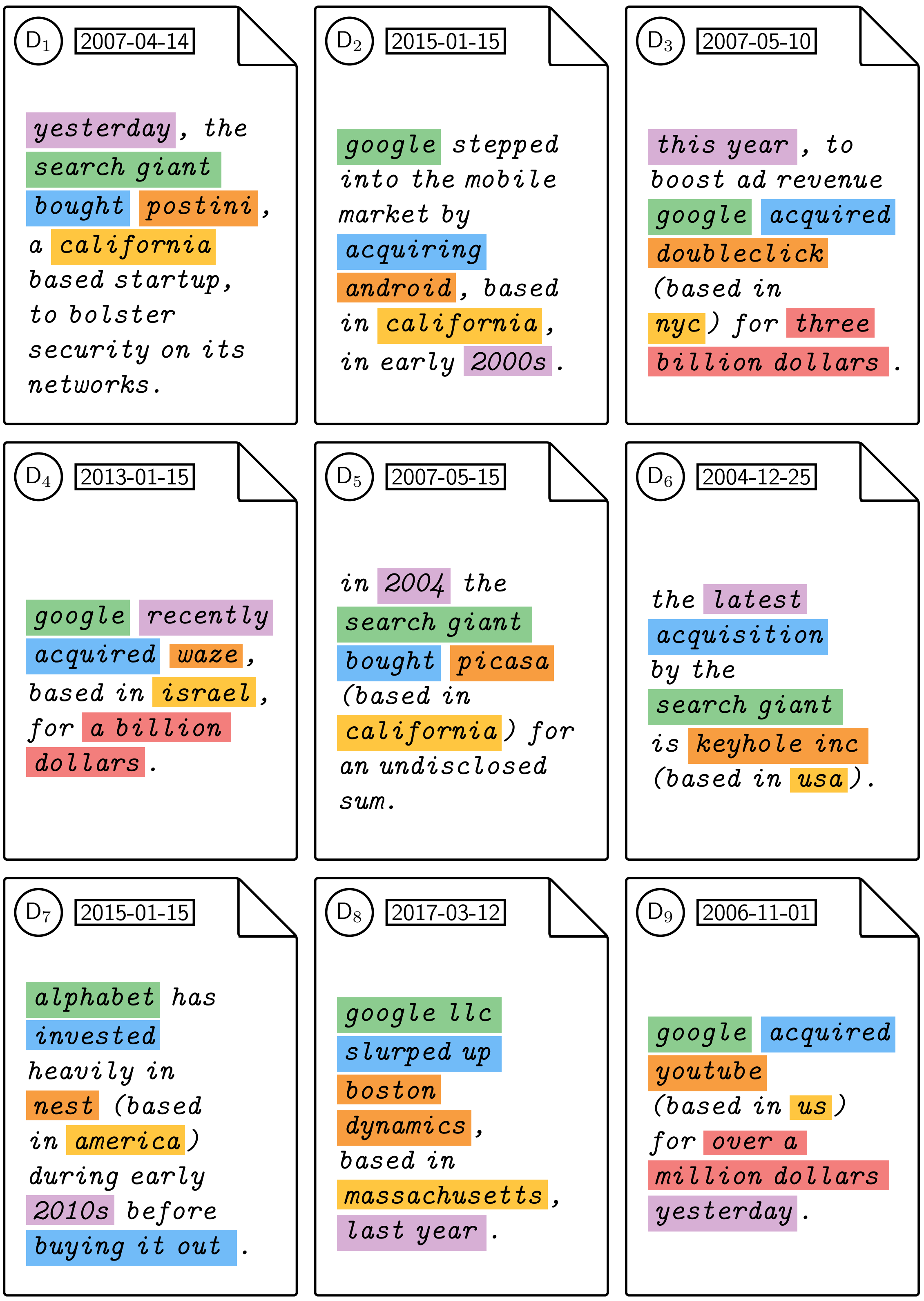
Indexing Annotated Document Collections

Journalists and scholars in humanities often undertake knowledge-centric tasks such as information extraction (e.g., *companies such as* (NNP)$*$), relationship extraction (e.g., [*google* | *search giant*] (WORD)$*$ [*youtube* | *picasa*]), and question answering (e.g., [*google* | *search giant*] $\boxed{.*}$ *acquired* (ORG)). Such knowledge-centric tasks are of importance for knowledge acquisition, for instance when populating a knowledge graph. As mentioned earlier, semantic annotations help us impose a lexico-syntactic structure over text. This lexico-syntactic structure can assist us in querying text using regular-expression patterns expressed between annotations and word sequences. In a manner similar to GREP on Unix, we enable semantic-GREP over large annotated document collections. Publications produced as part of this research are [92, 98, 103].

Facts concerning emerging named entities can not be spotted using annotated document collections. As for them, named entity recognition and disambiguation (NERD) tools can not find an entry in the knowledge graph for linking. To retrieve text regions that detail relationships between emerging named entities, we must resort to finding mentions of all their aliases using only text. For example, to retrieve all sentences that contextualize the acquisitions of GOOGLE, we need to spot sentences that contain the natural language representation of this fact:

$$\left\langle \begin{Bmatrix} \textit{google} \\ \textit{google llc} \\ \textit{search giant} \end{Bmatrix}, \begin{Bmatrix} \textit{acquired} \\ \textit{acquisition} \\ \textit{buys out} \\ \textit{bought} \\ \textit{slurped up} \\ \textit{buying it out} \end{Bmatrix}, \begin{Bmatrix} \textit{youtube} \\ \textit{nest} \\ \textit{doubleclick} \\ \textit{postini} \\ \textit{waze} \\ \textit{android} \\ \textit{picasa} \\ \textit{keyhole inc} \end{Bmatrix} \right\rangle.$$

Hyper-phrase queries generalize such information-seeking behavior. In order to speedup the execution of hyper-phrase queries, we present an optimization framework that can determine an optimal join-order sequence for processing the posting lists of documents that witness entities and their relationships. We show that using our proposed optimization, we can retrieve evidences in the form of text regions in response to hyper-phrase queries within seconds. The indexing infrastructure that supports knowledge-centric tasks and the optimization of hyper-phrase queries are described in Part I of this thesis. Publications produced as part of this research are [100, 102].



D1 2007-04-14

yesterday, the search giant bought postini, a california based startup, to bolster security on its networks.

D2 2015-01-15

google stepped into the mobile market by acquiring android, based in california, in early 2000s.

D3 2007-05-10

this year, to boost ad revenue google acquired doubleclick (based in nyc) for three billion dollars.

D4 2013-01-15

google recently acquired waze, based in israel, for a billion dollars.

D5 2007-05-15

in 2004 the search giant bought picasa (based in california) for an undisclosed sum.

D6 2004-12-25

the latest acquisition by the search giant is keyhole inc (based in usa).

D7 2015-01-15

alphabet has invested heavily in nest (based in america) during early 2010s before buying it out.

D8 2017-03-12

google llc slurped up boston dynamics, based in massachusetts, last year.

D9 2006-11-01

google acquired youtube (based in us) for over a million dollars yesterday.

Figure 1.1: Example document contents with semantic annotations in the form of named entities, temporal expressions, and numerical values.

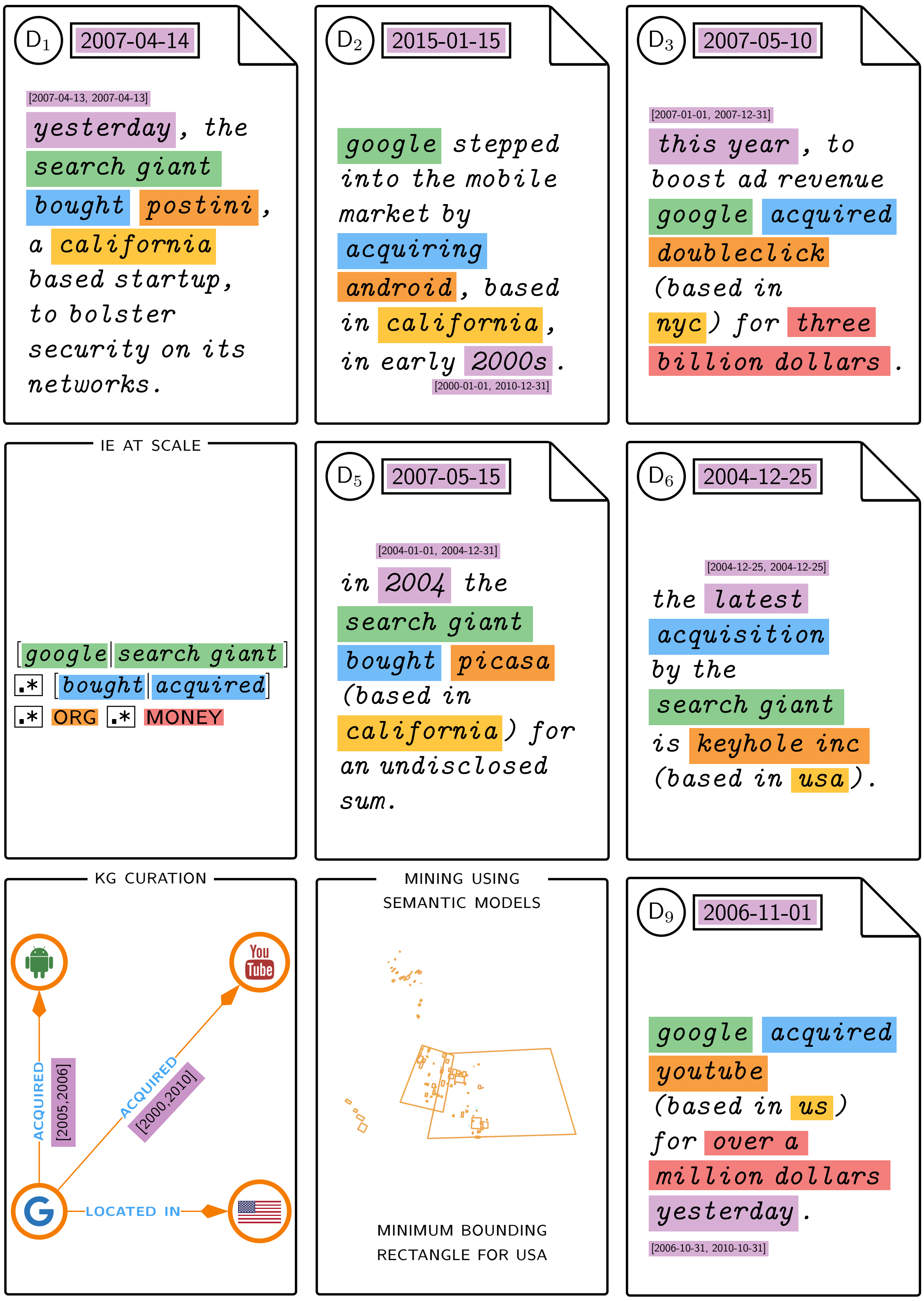

D1 2007-04-14
[2007-04-13, 2007-04-13]
yesterday, the search giant bought postini, a california based startup, to bolster security on its networks.
D2 2015-01-15
google stepped into the mobile market by acquiring android, based in california, in early 2000s.
[2000-01-01, 2010-12-31]
D3 2007-05-10
[2007-01-01, 2007-12-31]
this year, to boost ad revenue google acquired doubleclick (based in nyc) for three billion dollars.
IE AT SCALE
[google|search giant] .* [bought|acquired] .* ORG .* MONEY
D5 2007-05-15
[2004-01-01, 2004-12-31]
in 2004 the search giant bought picasa (based in california) for an undisclosed sum.
D6 2004-12-25
[2004-12-25, 2004-12-25]
the latest acquisition by the search giant is keyhole inc (based in usa).
KG CURATION
ACQUIRED
[2005,2006]
ACQUIRED
[2000,2010]
LOCATED IN
MINING USING SEMANTIC MODELS
MINIMUM BOUNDING RECTANGLE FOR USA
D9 2006-11-01
google acquired youtube (based in us) for over a million dollars yesterday.
[2006-10-31, 2010-10-31]

Figure 1.2: Leveraging semantic annotations for indexing, querying, and mining of large document collections. Using semantic annotations, we implement a GREP-like interface for information extraction at scale. With the ability to acquire knowledge for any combinations of word sequences and annotations, we can augment knowledge graph facts with additional metadata (e.g., time scopes for KG facts). Finally, we can model the semantics of annotations (e.g., minimum bounding rectangles for geographic locations) to compute cohesive clusters of events.

Querying Annotated Document Collections

Short and ambiguous queries often result in long lists of irrelevant documents for the user to explore. This is largely due to inherent ambiguity underlying natural language: meaning of a word depends on its context. Since, IR systems largely rely on text, there is bound to be a mismatch between user's intent and the information conveyed by the document. Annotations offer us deeper semantics to understand text in documents. We make use of these semantic annotations, to assist users in information consumption and leading them to their information need effortlessly. To this end, we first make use of semantic annotations in the form of temporal expressions, that can help us explore longitudinal document collections such as news archives. To do this, we describe an approach to suggest time intervals of interest to keyword queries (e.g., the time intervals $[2000, 2007]$ and $[2010, 2015]$ are of relevance for the query *google acquisitions*). Building on this work, we then describe methods to classify temporal queries at different levels of granularity (e.g., the query *google quarterly report* is a temporally ambiguous query at month granularity) and diversify documents retrieved for temporal queries using time intervals of interest (e.g., re-ranking documents that cover the time interval $[2000, 2007]$ and $[2010, 2015]$ higher for the query *google acquisitions*). The aforementioned contributions appear in [93, 94, 96, 97].

We next consider annotations in the form of geographic locations and other named entities in addition to temporal expressions, for assisting the user query annotated document collections. To this end, we describe an approach that generates semantic aspects for short and ambiguous queries (e.g., $\langle\{$POSTINI, ANDROID, PICASA$\}$, $\{$CALIFORNIA$\}$, $[2000, 2007]\rangle$ for the query *google acquisitions*). The semantic aspects generated by our approach, help expose documents' key facets, such that the user can explore documents without the need to read their contents. With the help of semantic aspects, we therefore have the potential of uplifting unstructured text in documents to an aspect-structured representation. Methods for assisting users in querying semantically annotated document collections are described in Part II of this thesis. Publications that were produced as part of this research are [104, 105].

Mining Annotated Document Collections

Semantic annotations can be further used to mine interesting insights from large document collections. In this direction, we make four contributions. First, we make use of temporal information present in documents to augment knowledge graph (KG) facts. KGs describe relationships between entities that can often be anchored in time (e.g., see $\langle$GOOGLE, ACQUIRED, YOUTUBE$\rangle$ in Figure 1.2). To ascertain the validity of time intervals for such temporal relations we can extract and aggregate their evidence from annotated document collections. Second, we can leverage semantic annotations in the form of named entities, geographic locations, and temporal expressions to mine events from news archives. To do so, we propose an algorithm that aggregates the event evidences, retrieved for a keyword query (e.g., *google acquisitions*), using semantic models for named entities, geographic locations, and temporal expressions. Third, we present JIGSAW, an end-to-end query driven system that generates structured tables for user-defined schema from unstructured text. To generate tables, JIGSAW provides operators that help define the table schema, fill in NULL cell values, and link together near-duplicate rows to generate high-quality tables from large annotated document collections. Fourth and finally, semantic annotations in documents can lend themselves for insightful analytics. For instance, we can depict the co-occurrence statistics of named entities and time intervals in documents retrieved for the query *google acquisitions*. Such analytics give an visual abstract of the information contained in large semantically annotated document collections. Contributions described for mining semantically annotated document collections are described in Part III of this thesis. Publications produced as part of the aforementioned contributions are [99, 101, 106, 107].

Applications

The goal of this doctoral dissertation has been to devise intelligent indexing methods and algorithms that leverage semantic annotations to search large document collections and for mining insights (e.g., real-world events). The research has potential applications in *digital humanities*, where social scientists are interested in history-oriented study of large born-digital text collections. Anthropologists are interested in cultural and linguistic shifts that occur in such collections. Collectively, we can perform *computational culturomics* [156] on document collection to study anthropological trends. For instance, events can be used to link and summarize information in multiple and diverse text collections. In short, semantic annotations provide us a way to explore large document collections, which otherwise through manual effort is impossible.

1.3 Publications

The publications produced as part of this doctoral research are categorized under three broad themes: indexing, querying, and mining of semantically annotated document collections. The contributions have appeared in fourteen peer-reviewed publications and are listed below.

I. Indexing Annotated Document Collections

1. Dhruv Gupta. Event Search and Analytics. In WSDM 2016.

2. Dhruv Gupta and Klaus Berberich. GYANI: An Indexing Infrastructure for Knowledge-Centric Tasks. In CIKM 2018.

3. Dhruv Gupta and Klaus Berberich. Structured Search in Annotated Document Collections. In WSDM 2019.

4. Dhruv Gupta and Klaus Berberich. Efficient Retrieval of Knowledge Graph Fact Evidences. In ESWC 2019.

5. Dhruv Gupta and Klaus Berberich. Optimizing Hyper-Phrase Queries. Under Submission.

II. Querying Annotated Document Collections

6. Dhruv Gupta and Klaus Berberich. Identifying Time Intervals of Interest to Queries. In CIKM 2014.

7. Dhruv Gupta and Klaus Berberich. Temporal Query Classification at Different Granularities. In SPIRE 2015.

8. Dhruv Gupta and Klaus Berberich. Diversifying Search Results Using Time - An Information Retrieval Method for Historians. In ECIR 2016.

9. Dhruv Gupta and Klaus Berberich. A Probabilistic Framework For Time-sensitive Search. In NTCIR-12 2016.

10. Dhruv Gupta, Klaus Berberich, Jannik Strötgen, and Demetrios Zeinalipour-Yazti. Generating Semantic Aspects for Queries. In JCDL 2018.

11. Dhruv Gupta, Klaus Berberich, Jannik Strötgen, and Demetrios Zeinalipour-Yazti. Generating Semantic Aspects for Queries. In ESWC 2019.

III. Mining Annotated Document Collections

12. Dhruv Gupta and Klaus Berberich. Identifying Time Intervals for Knowledge Graph Facts. In WWW 2018.

13. Dhruv Gupta, Jannik Strötgen, and Klaus Berberich. EVENTMINER: Mining Events from Annotated Documents. In ICTIR 2016.

14. Dhruv Gupta and Klaus Berberich. JIGSAW: Structuring Text into Tables. In ICTIR 2019.

15. Dhruv Gupta, Jannik Strötgen, and Klaus Berberich. DIGITALHISTORIAN: Search & Analytics Using Annotations. In HistoInformatics 2016.

1.4 Outline

The rest of the thesis is organized as follows. In Chapter 2, we cover the required background for the technical discussion. Part I of the thesis covers the contributions made towards indexing of annotated document collections. In Chapter 3, we describe, GYANI, the indexing infrastructure that supports knowledge-centric tasks. In Chapter 4, we propose an optimization framework to speed up the retrieval of evidences for hyper-phrase queries.

Part II of the thesis covers the contributions made towards querying of annotated document collections. In Chapter 5, we describe the generative model for identifying time intervals of interest to keyword queries. In Chapter 6, we describe our model for classifying temporal queries at different granularities. In Chapter 7, we describe the diversification of search results using temporal expressions. In Chapter 8, we describe the Temporalia-2 tasks at the NTCIR-12 competition and the effectiveness results of our time-sensitive search engine at those tasks. In Chapter 9, we describe the XFACTOR algorithm that generates semantic aspects for short ambiguous keyword queries using temporal expressions, geographic locations, and other named entities.

Part III of the thesis covers the contributions made towards mining of annotated document collections. In Chapter 10, we describe our approach that determines relevant time intervals for knowledge graph facts. In Chapter 11, we describe EVENTMINER, a clustering algorithm that considers semantics of annotations for aggregating event evidences. In Chapter 12, we describe JIGSAW, an end-to-end query-driven system that generates structured tables for user-defined schema from unstructured text. In Chapter 13, we describe DIGITALHISTORIAN, a complete time-sensitive search engine that allows users to explore longitudinal document collections via temporal expressions. Chapter 14 concludes this thesis by presenting a summary of contributions this doctoral dissertation has made to the research community.

CHAPTER 2

PRELIMINARIES

In this chapter, we cover the notations and concepts required for the technical discussion of the thesis. First, we describe how documents in a collection can be modeled using text and semantic annotations. Second, we outline the basic framework of an information retrieval (IR) system and cover the fundamental data structure required for the storage of documents in a collection. Third, we describe the models that leverage only text for ranking documents given a query. Fourth, we describe the semantic annotations we use and how these can be obtained from natural language processing (NLP) tools. We end this chapter with a discussion of the effectiveness measures that we use to assess the quality of results obtained from our methods and systems.

2.1 Document Models

Collection Types. Consider a collection of documents, $\mathcal{D} = \{d_1, d_2, \ldots, d_{|\mathcal{D}|}\}$. Publicly available document collections can be put into four different categories. In the first category, there are archives of news articles. Examples of news archives are: the New York Times Annotated corpus [18], the fifth Edition of English Gigaword [5], and the GDelt news archives [7]. In the second category, there are archives of web pages. Examples of web archives are: the Living Knowledge corpus [15], ClueWeb'09 [1] web archive, and ClueWeb'12 [2] web archive. In the third category, there are archives of books, e.g., Project Gutenberg [20] stores works that are in the public domain. In the fourth and final category are encyclopedic collections, for example Wikipedia [23].

Bag-of-Words Document Model. Each document in the collection $d \in \mathcal{D}$ can be modeled in several ways. For our contributions, there are three different ways in which we model the documents. In the first model, each document $d \in \mathcal{D}$ contains only a bag of words $d_{\mathcal{V}}$ drawn from vocabulary $\Sigma_{\mathcal{V}}$. Thus, each document in such a model can simply be represented by the words it contains disregarding their positional information: $d = \{d_{\mathcal{V}}\}$.

Bag-of-Annotations Document Model. In the second model, we can augment the first representation with bags-of-annotations. The bags-of-annotation can contain annotations for part-of-speech tags, named entities, or temporal expressions. We shall discuss in detail the annotations that we utilize and how they are obtained in Section 2.4. For now, let the alphabet concerning an annotation type be denoted by $\Sigma_{\mathcal{L}}$. We can then represent each document in the collection $d \in \mathcal{D}$ using the second representation for documents as:

$$d = \{d_{\mathcal{V}}, d_{\mathcal{L}_1}, d_{\mathcal{L}_2}, \ldots, d_{\mathcal{L}_n}\}. \tag{2.1}$$

Lexico-Syntactic Document Model. The first and second representation for documents disregard the positional information among words and the annotations. Furthermore, for the second representation we do not keep the information as to which annotations adorn which words in the document.

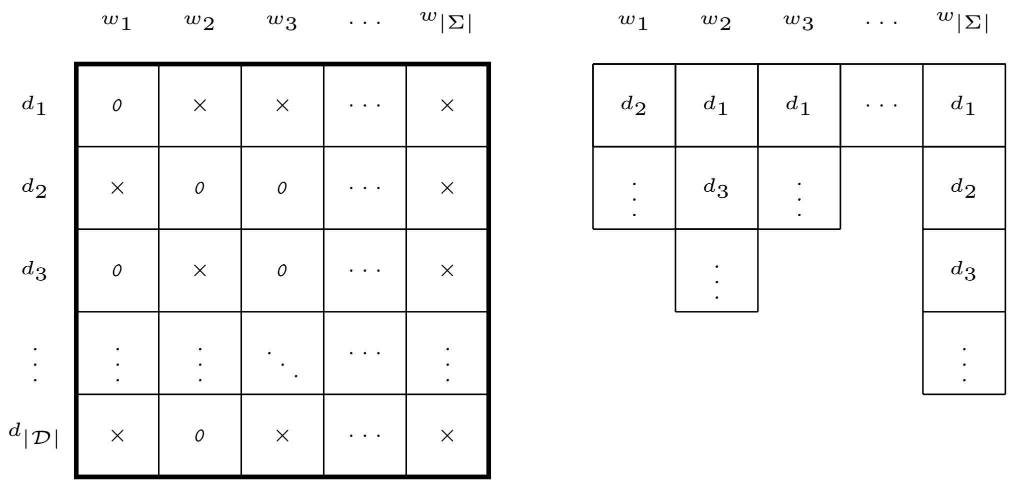
In the third model, we add syntactical information regarding words, annotations, and the sentences which contain them to the document model. We therefore associate positional information regarding the words and annotations in each document. By doing so, we can then identify which annotations adorn which word sequences. Within this document model, we assume each annotation (including words) ℓ are obtained from an alphabet $\Sigma_{\mathcal{L}}$. Each document in the collection $d \in \mathcal{D}$ thus consists of layers of annotations $d_{\mathcal{L}}$. Each annotation layer is modeled as:

$$d_{\mathcal{L}} = \langle \ell_{[i,j]}, \ldots, \ell_{[k,l]} \rangle, \tag{2.2}$$

where, each annotation ℓ *spans* the positions described by the interval in subscript ($[i, j]$ with $i \leq j$).

2.2 Information Retrieval Framework

Given a keyword query q, a set of pseudo-relevant documents $\mathcal{R}$ for it is retrieved using the method IR($\bullet$). We follow the notational convention in [55] to describe a simple search system: $\mathcal{R} = \text{IR}(q, \Theta, \mathcal{D})$. IR($\bullet$) is a retrieval method wherein the argument q specifies the query keywords, $\Theta \in \mathbb{R}^m$ specifies a set of parameters relevant for IR($\bullet$), and $\mathcal{D}$ specifies the document collection. We next illustrate how to implement an IR retrieval system using only text within documents.


w_1 w_2 w_3 $\cdots$ $w_{|\Sigma|}$
d_1 d_2 d_3 $d_{|\mathcal{D}|}$
0
×

(a) Document-Term Matrix. (b) Inverted List Representation.

Figure 2.1: Illustration of indexing documents using a simple document-term matrix. The ×s denote either 1 (a Boolean value indicating presence of terms) or its term frequency (a non-negative integer). Note that the matrix is sparse, i.e., the number of zeros is much larger than the number of non-zero values. To avoid materializing the complete sparse matrix for document retrieval, inverted lists that contain document identifiers can be created instead.

Document-Term Matrix. The simplest data structure to organize documents for retrieval is using a matrix. The matrix $M_{|D|\times|\Sigma|}$ stores the association between documents (as rows) to terms (as columns) (see Figure 2.1). Each cell in the document-term matrix can then either record a 1 (a Boolean value indicating presence of terms) or a its term frequency (a non-negative integer). Absence of terms in documents is simply recorded by a 0. Boolean retrieval of documents can be done by selecting the documents that correspond to the query terms (i.e., words in the column). Document-term matrices are useful in scenarios when its representation can fit into main memory. That is, it is only useful for a small number of documents. As the collection size grows, the document-term matrix may turn out to be sparse. That is, the number of zero values are much larger than non-zero values. In such a case, the amount of memory required to materialize a dense matrix allocates more memory to zero values than to non-zero values.

Inverted Indexes. To avoid materializing the complete matrix that is sparse, we can store the association of document and terms using a hash map or an inverted index. An inverted index, records for each word in the collection's vocabulary, a list of the document identifiers containing that word: $w_i \rightarrow \cup\, d$. To further save space, the document identifiers in the list can be arranged in an ascending order and compressed using techniques such as Delta encoding. Modern IR systems, often require more metadata than just the document identifiers. Additional metadata in the form of publication dates, positional offsets, annotations, and term frequencies are then also accommodated along with compressed lists of document identifiers as payload.

Index Compression. Inverted indexes comprise of posting lists for indexing units that are essentially arrays of sorted integers [50]. As mentioned before, the lists of integers in inverted indexes can be compressed by computing the differences (deltas) between subsequent elements and storing the resulting differences. Improving on this, lists of integers can be further partitioned into fixed sized pages or blocks to determine the begin and end bounds against which the deltas need to be computed. This technique known as *Frame of Reference*, is used so that the deltas can be stored in a fixed number of bits [56, 87]. Zukowski et al. [216] describe how to determine the page or block size such that it provides optimal compression. This technique known as *Patched Frame of Reference*, amends the Frame of Reference technique to avoid selecting partitions that contain large values, thus causing more bits to be allocated for storing the deltas [56]. An excellent implementation of the Patched Frame of Reference in the Java programming language is provided by Lemire [11].

2.3 Ranking Documents

IR($\bullet$) represents a *totally ordered* relation in $\mathcal{R} \subseteq \mathcal{D}$. That is, given $d, d' \in \mathcal{R}$ then either $d \preceq d'$ or $d' \preceq d$ [71], where ties are broken arbitrarily. Internally, IR($\bullet$) utilizes a score($\bullet$) function to assign *relevance* of documents for the given query to produce the total order,

$$\text{score}(d_{\mathcal{W}}, \mathcal{R}) : \{d_{\mathcal{W}} \in d \mid \forall d \in \mathcal{R}\} \rightarrow \mathbb{R}^{+}. \tag{2.3}$$

Okapi BM25 is an example scoring function that is based on the Binary Independence Model [50]. The Binary Independence Model assigns relevance to documents by accounting for the presence of query terms. Two key assumptions that the Binary Independence Model posits are: that the terms occur independent to each other and that a document is relevant to a query if and only if the query terms are present in the document [50]. The Okapi BM25 is one such derivative of the Binary Independence Model that accommodates parameters that can be optimized for different types of document collections. To this end, it provides parameters $\Theta \in \mathbb{R}^m$ to adjust the importance given to the query terms (k_3), document terms (k_1), and document length (b). Without getting into the details of its derivation, the ranking formula for Okapi BM25 can be stated as [150, 178]:

$$\sum_{w \in q} \log \left[\frac{|\mathcal{D}|}{\text{df}(w)} \right] \cdot \frac{(k_1 + 1) \cdot \text{tf}(w, d)}{k_1((1 - b) + b \cdot {}^{|d|}/_{\mu(d)}) + \text{tf}(w, d)} \cdot \frac{(k_3 + 1) \cdot \text{tf}(w, q)}{k_3 + \text{tf}(w, q)} \tag{2.4}$$

In Equation 2.4, df(w) gives the document frequency of the word in entire collection $\mathcal{D}$ and $\mu(d)$ the average document length in the collection [150, 178].

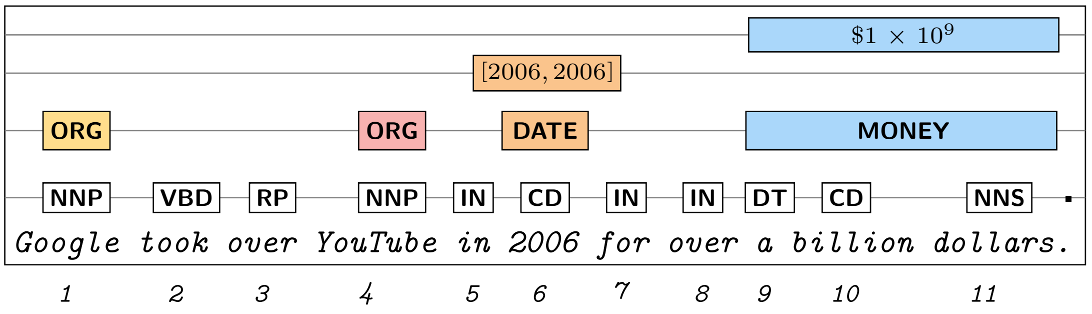

$1 × 10^9
[2006, 2006]
ORG
ORG
DATE
MONEY
NNP VBD RP NNP IN CD IN IN DT CD NNS .
Google took over YouTube in 2006 for over a billion dollars.
1 2 3 4 5 6 7 8 9 10 11

Figure 2.2: An example of an annotated text region. Annotations shown are part-of-speech, named entities, resolved temporal and numerical expressions.

Unigram Language Model with Dirichlet Smoothing is an example scoring function inspired from probability theory. To rank documents $d \in \mathcal{R}$ retrieved for a given query q, a generative model is assumed. The generative story radically differs from the Binary Independence Model and is as follows. The document collection represents the distribution of words and documents represent a sample of words from the collection [150]. Using the documents as a sample of words, the unigram language model then measures what is the likelihood of generating the query terms from the document [150]. An important assumption that unigram language models make is that each word occurs independent from each other (just like the Binary Independence Model). That is, dependence or positional information amongst words is discarded [150]. The query likelihood of a given keyword query q can be estimated by using term frequencies of words in document $\text{tf}(w, d)$. Additionally, Dirichlet smoothing is used to address the zero-probability problem of having words that could not be found in the document. The unigram language model with Dirichlet smoothing can be stated as:

$$P\,(\,q\,|\,d\,) = \prod_{w \in q} \frac{\text{tf}(w, d) + k \cdot \frac{\text{tf}(w, \mathcal{D})}{|\mathcal{D}|}}{|d| + k}\,, \tag{2.5}$$

where, $P(q|d)$ the likelihood of generating the query q from document d, $|d|$ denotes document length, $|\mathcal{D}|$ denotes the collection size, and k is a constant.

2.4 Semantic Annotations

Natural language processing (NLP) tools allow us to markup various kinds of annotations in text. We specifically focus on five fundamental types of semantic annotations that are commonly provided by off-the-shelf NLP toolkits: part-of-speech, temporal expressions, numerical values, geographic locations and other named entities. An example of an annotated text is shown in Figure 2.2. We describe these five different types of semantic annotations in the following paragraphs and justify their importance.

Part-of-Speech (PoS) Tags are assigned to a word based on common linguistic characteristics derived from their surrounding terms [128]. Thus, a word's PoS tag can quickly help to describe the context in which it occurs [128]. Examples of PoS tags are: nouns (NN), verbs (VB), and quantities (CD). PoS tags are significant as they form the basis for many tasks in computational linguistic. For instance, PoS annotations such as nouns can be used to generate text summaries. They can also be employed for complex natural language processing tasks such as named entity recognition and named entity disambiguation [128].

Temporal Expressions convey mentions of time in text. Natural language descriptions of time are often convoluted, as time can be explicit as a concrete date (e.g., July 4, 1776), it can be implicit emphasizing that it is commonsense knowledge and refers to already known facts (e.g., independence day of the usa), or it can be relative with respect to dates mentioned in the narrative of the text (e.g., yesterday). Temporal annotators are able to detect these implicit, explicit, and relative temporal expressions. Moreover, the detected expressions can be resolved to definite time intervals by using metadata such as publication dates. Formally, each temporal expression that is annotated is represented as an interval ($t = [b, e]$) with a begin and end time point. Temporal expressions are significant in computational linguistics as they signify the presence of events.

Numerical Values are resolved values of words tagged as cardinal numbers (CD) that are not temporal expressions. Examples of such annotations are percentage values, monetary values, and other numerical figures. Just like temporal expressions, these numerical values can be explicit (e.g., 95%) or implicit (e.g., a dozen). Numerical values are useful in quantifying tragic events, e.g., identifying casualties of a natural disaster or for quantifying financial events, e.g., profit and loss for a company.

Named Entities are mentions of persons, organization, locations etc. in text. These annotations are classified based on context surrounding their mentions. For instance, in the sentence, ⟨alan turing was a scientist who lived in britain⟩, the mentions alan turing and britain are classified as person and location, respectively. Named entities can be further disambiguated to their canonical entries (URI) in knowledge graphs. Entity annotations allow for advanced text analytics, e.g., the ability to perform entity summarization by retrieving all sentences in documents that contain mentions of that entity.

Geographical Locations can be obtained as a subset of named entity annotations. Geographic locations such as those known to be cities, countries, or continents, can be filtered from named entities by using geographical relations stored in a knowledge graph. These relations can be further used to resolve the locations to their geographical coordinates. With these set of coordinates, we can subsequently construct complex models for their representation, e.g., minimum bounding rectangles over the coordinates.

These five kinds of annotations offer an immense opportunity for various text analytics applications that can be developed for linguists, scholars in humanities, and journalists.

2.5 Effectiveness Measures

To assess the quality of results produced by our methods, we measure precision and recall of the generated result set $\mathcal{R}$ with respected to the acquired ground truth $\mathcal{G}$. Precision is measured by computing the fraction of results (e.g., documents) that are deemed relevant with respect to the ground truth from the total result set size [150]:

$$\text{Precision} = \frac{\sum_{1 \le i \le |\mathcal{R}|} \mathbb{1}\big(\mathcal{R}_i = \text{Relevant}\big)}{|\mathcal{R}|}. \tag{2.6}$$

In the equation above, the indicator function $\mathbb{1}(\mathcal{R}_i = \text{Relevant})$ evaluates whether the document $\mathcal{R}_i$ is relevant or not. If the method produces a ranked list of results, then we can compute precision at k ($P@k$) by truncating the list at length k. The precision is then computed as:

$$\text{Precision}@k = \frac{\sum_{1 \le i \le k} \mathbb{1}\big(\mathcal{R}_i = \text{Relevant}\big)}{k}. \tag{2.7}$$

Recall, on the other hand, measures the number of relevant results identified by the system from the total number of relevant results in the associated ground truth $\mathcal{G}$. This is can be formalized as [150]:

$$\text{Recall} = \frac{\sum_{1 \le i \le |\mathcal{R}|} \mathbb{1}\big(\mathcal{R}_i = \text{Relevant}\big)}{|\mathcal{G}|}. \tag{2.8}$$

Again, if the method produces a ranked list of results, then recall at k ($R@k$) can be computed by truncating the result list at length k. Both precision and recall can be summarized by using the F_1 measure. The F_1 computes the harmonic mean of precision and recall to combine both measures:

$$F_1 = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}. \tag{2.9}$$

In ranked list of results, it is often desirable to assess the quality of ranking by identifying the position of the relevant result. To assess this quality, the reciprocal rank (RR) metric is used. RR computes the inverse of the position at which the first relevant result was identified [50] (ignoring zero valued results):

$$\text{Reciprocal Rank} = \left(\min_{1 \le i \le |\mathcal{R}|} \mathbb{1}\big(\mathcal{R}_i = \text{Relevant}\big) \cdot i \right)^{-1}. \tag{2.10}$$

Mean reciprocal rank (MRR) reports the average of the reciprocal rank over a query workload presented to the system.

PART I

INDEXING ANNOTATED DOCUMENT COLLECTIONS

CHAPTER 3

GYANI: AN INDEXING INFRASTRUCTURE FOR KNOWLEDGE-CENTRIC TASKS

• Dhruv Gupta and Klaus Berberich. GYANI: An Indexing Infrastructure for Knowledge-centric Tasks. In CIKM 2018, pages 487–496.

• Dhruv Gupta and Klaus Berberich. Structured Search in Annotated Document Collections. In WSDM 2019, pages 794–797.

3.1 Introduction

Knowledge-centric tasks such as information extraction rely on meaningful text regions extracted from the analysis of large semantically annotated document collections. A semantically meaningful text region connects, for instance, a named entity to a literal via a paraphrase of a knowledge graph predicate (e.g., alan turing published papers during the 30s) within a context window of few sentences. Currently there exists no indexing infrastructure that allows for interactive querying of such text regions on large annotated document collections. To enable such complex knowledge-centric tasks at scale, we are faced with three key challenges. The first challenge is to provide a more expressive query language to enable knowledge acquisition at scale. The second challenge is to identify a data model that respects the salient relationships between sequences of words and semantic annotations. The third challenge is to identify key indexes as building blocks using which we can then assemble the required search results quickly. By building such a core infrastructure and an expressive language for querying such text regions at scale we can provide more training data for improving the effectiveness of complex machine learning algorithms that are developed to serve these knowledge-centric tasks.

A flexible and more expressive query language for us is one that goes beyond simple Boolean operators. What is thus needed is a counterpart to GREP for analysis of large semantically annotated document collections. GREP is a powerful Unix utility that allows for regular expression based search over text documents. It is an indispensable tool when trying to find and manipulate lines of text matching a particular regular expression. However, when it comes to searching millions of annotated documents, a counterpart to GREP is missing. This is challenging, as natural language text, unlike field delimited files (e.g., TSV) is not structured. However, with the help of modern natural language processing tools we can impose a lexico-syntactic structure over text. Such tools now allow us to annotate large document collections with various kinds of semantic annotations, such as part-of-speech (e.g., NN), temporal expressions, (e.g., last year), and named entities (e.g., Alan Turing). The annotations give deeper semantics to the terms as well as provide canonical linguistic structure to text. This lexico-syntactic structure thus offers us an opportunity to implement a counterpart to GREP over annotated document collections.

Queries composed of regular expressions over word sequences and annotations offer us an opportunity for knowledge acquisition at scale. With such a query language we can simplify many knowledge-centric tasks. For instance, a template to identify n-ary relations about disasters can be expressed as: ⟨(NUMBER) were killed in (LOCATION) on (DATE)⟩. Below we discuss existing solutions for knowledge-centric tasks and outline how these approaches can benefit from our infrastructure.

Information Extraction requires one to acquire facts that hold between named entities from unstructured text [112, 162]. For this task, extraction templates are employed, e.g., Hearst patterns [112]. A template for identifying scientists in a document collection, can be written as: ⟨scientists such as $\text{NNP}_1 \ldots \text{NNP}_n$⟩. To execute this example, documents that contain the terms ⟨scientists such as⟩ are retrieved and the sentences are annotated for part-of-speech tags and subsequently filtered to produce relations. With GYANI this can be achieved interactively by issuing the following query: ⟨scientists such as (PERSON)$*$⟩.

Question Answering. Knowledge Graphs encode relationships or facts in the form of ⟨SUBJECT, PREDICATE, OBJECT⟩ triples. Natural language questions such as: ⟨which countries joined nato and when?⟩, need to be transformed into a structured query using the SPARQL language [80, 183]. However, knowledge graphs are rarely complete (e.g., with respect to temporal information) and thus we are forced to spot the answers for such questions in large annotated document collections. By providing the facility to formulate a question as a template that expresses the relationship between a named entity and temporal expression we can provide a large extracted set of sentences as an input to machine learning algorithms that can further reason about the correct time interval for this question [99]. For example, with GYANI we can extract training data with the following query: ⟨(LOCATION) joined nato (DATE)⟩.

Fact Spotting. The inverse task of information extraction is to find textual evidence in support of the facts in a knowledge graph [154]. To deal with the linguistic variations in which many of the canonical relations are phrased, paraphrase dictionaries are employed [162]. To account for the many different ways a named entity can be mentioned, its surface forms are added to increase the recall of the pattern-matching procedure. The procedure to perform this task can be greatly simplified by combining regular expressions, annotations, and word sequences. This has important consequences when we are trying to spot out-of-knowledge-graph entities, for which we may have to manually specify many surface forms for a yet to be canonicalized named entity. For example, consider the query: ⟨[united states | us | usa] to criticize [russia | russland]⟩.

Semantic Search. Semantic annotations are an important building block for many linguistic and information retrieval tasks as they lend themselves for conveying deeper semantics to the terms in text. Thereby, allowing the user to convey her information need in a more structured manner [36, 80]. However, current inverted indexes offer only limited capabilities to search and analyze semantically annotated text. Using prior art many documents that might qualify to satisfy the user's information need are lost due to this semantic gap. For instance, a user issuing the query ⟨nineties decade⟩, will miss documents that also contain the time interval $1990 - 1999$ or the query ⟨paris hilton⟩ may retrieve documents matching the location and hotel when the intent was related to the celebrity. We propose to bridge the semantic gap by providing operators to attach meanings to terms. For example, to retrieve sentences in document collections detailing relationships between the person, paris hilton and the time period, nineties decade: ⟨(paris hilton)⊕(PERSON) (WORD)$*$ (nineties decade)⊕([1990,1999])⟩.

Implementing a GREP-like interface over large annotated document collections poses many challenges. At the modeling level, we need to identify a data model in which we can represent a text document along with a multitude of annotations while preserving the sequential order of words. At the index implementation level, we need to work through all possible choices for indexing units in the design space such that the index structures provide fast query execution times (i.e., order of millisecond query execution times for complex and lengthy queries). To address these challenges, we describe GYANI, an infrastructure that indexes semantically annotated text using a novel data model and provides a highly expressive language for queries involving regular expressions over word sequences and annotations. Furthermore, we show how we model the possible combinations of word sequences and annotations in a multi-layered data model to process complex and verbose GREP-like queries for knowledge-centric tasks quickly and at scale.

Outline. First, we describe a novel data model to represent text with various layers of semantic annotations (Section 3.3.1). Second, we propose a novel query language involving regular expressions between words and annotations (Section 3.3.2). Third, we present an efficient implementation of our data model that provides fast query execution times (Section 3.3.3). Finally, we construct a comprehensive testbed of knowledge-centric tasks to show the efficiency of GYANI at the tasks of information extraction, relation extraction, question answering, fact spotting, and semantic search (Section 3.4).

3.2 Related Work

Searching Semi-Structured Text. The earliest attempts in information extraction from semi-structured text documents relied on region algebras [61, 64, 179]. The PAT system [179] supported expressions that could match SGML tags to match text regions for information extraction. Their approach also allowed the user to query for regions of text with the help of the *region expressions*. Clarke et al. [61] proposed a data model that relied on maintaining *generalized concordance lists* to index positional spans for SGML tags. A clear contrast between these early works and our work is in accommodating semantic annotations in text.

Lalmas [139] provides an overview of work on XML retrieval including expressive query languages such as XPath and XQuery. While documents with semantic annotations could be represented as XML, this would entail a blowup in space and formulating regular expression queries for knowledge-centric tasks in the aforementioned languages is all but intuitive.

Miller and Myers [157] were the first to realize the unavailability of popular Unix delimited-text manipulators such as GREP for semi-structured documents. Their data model represented text regions by Cartesian coordinates. The authors then leveraged R*-trees to intersect and compute proximity between rectangles. Cho and Rajagopalan [60] focused on how to allow queries to contain regular expressions at character level. Their proposal was the concept of a k-gram index that indexed selective n-grams for efficient regular expression based search. However, both these approaches do not provide any scope for handling semantically annotated text.

Searching Annotated Text. Ferrucci and Lally [80] presented *Unstructured Information Management Architecture* (UIMA), a comprehensive and integrated suite of annotators and text analytics pipeline. UIMA supported modeling implicit annotations present in text as “common analysis structure” that allowed overlaying of annotations to cover common portions of text. Cafarella and Etzioni [51] proposed the “neighbor index” that provided Boolean queries involving phrases, annotations, and functions over annotations. Both the UIMA framework and “neighbor index” aimed at providing the functionality of Boolean queries (not regular expressions) over annotated text.

Li and Rafiei [143] described how to execute queries involving wildcards, part-of-speech tags, and words. Their implementation relied on commercial search engines for retrieving text snippets. Bast and Buchold [36] proposed an index architecture that incorporates both knowledge graph relations associated with entities and the contextual text containing that entity. This thus allows for search over a combined index of knowledge graph and unstructured text. Massung et al. [209] investigated how to index aggregated feature vector representations of text along with documents for a unified framework for analysis. A recent survey on information extraction over text and knowledge graphs [37] lacks any mention of an implementation that allows for structured search involving word sequences, annotations, and regular expressions.

The computational linguistics community also looked into query languages for annotated corpora (including additional annotations such as dependencies) [83, 136]. Scalability, though, has not been a focus in those works and the considered corpora were at least an order of magnitude smaller than the ones we consider in this work.

Searching Text Using RDBMS. Solutions to enable structured search over semantically annotated text can also addressed using conventional database technologies [66, 214]. However, none of these approaches supports wildcard operators. By adopting the RDBMS approach, Zhou et al. [214] described a data model that encodes words, annotations, the confidence of the accompanying annotations, and its positional span. Queries over this data model are mapped to SQL queries for execution. Cornacchia et al. [66] considered the problem of implementing IR systems using array databases with efficient storage schemes for sparse arrays.

3.3 GYANI

Gyan in Hindi means *knowledge* and *gyani* refers to a person who possesses knowledge. Our proposed infrastructure is named GYANI to personify an index over text that contains knowledge by virtue of semantic annotations adorned on text. We next describe the data model that consists of layers of annotations that can be attached to sequences of words, thereby allowing different semantic interpretations of natural language. We then describe the query language to perform regular expression based search in semantically annotated text. Lastly, we discuss the design space of the data structures used for indexing annotated documents in GYANI.

3.3.1 Data Model

Consider a document collection, $D = \{d_1, d_2, \ldots, d_N\}$. Each document in the collection $d \in D$ consists of layers of sequences containing words or semantic annotations. Each layer of sequences consists of elements drawn from a particular vocabulary with their positional span. Concretely, each element ℓ in a layer $\mathcal{L}$ is a relation defined over the Cartesian space described by its layer alphabet $\Sigma_{\mathcal{L}}$ and an interval of natural numbers $\mathbb{N}$ indicating their positions in the sequence. Formally, the definition of an element is,

$$\ell \subset \mathbb{N} \times \mathbb{N} \times \Sigma_{\mathcal{L}}. \tag{3.1}$$

The word layer in each document consists of words drawn from vocabulary $\Sigma_{\mathcal{V}}$ with unit length positional spans. Formally,

$$d_{\mathcal{V}} = \langle w_{[1,1]}, \ldots, w_{[|d|,|d|]} \rangle, \tag{3.2}$$

where, $|d|$ denotes the number of words in the document and the interval in subscript records the unit length spans as *positions* of the word in the sequence.

Natural language word sequences present in the documents can be preprocessed with various kinds of semantic annotators, e.g., part-of-speech, temporal expressions, and named entities. Each type of annotation thus signifies an interpretation (semantics) of the terms by inspecting the signals derived from its surrounding context. For instance, in the sentence, ⟨peace was brokered between them during 1903⟩, the term 1903 will be tagged as a cardinal number by part-of-speech tagger and as a date by a temporal tagger. We treat each sequence of annotation derived over a sequence of words as an annotation layer. An annotation layer $d_{\mathcal{L}}$ over a document is denoted by,

$$d_{\mathcal{L}} = \langle \ell_{[i,j]}, \ldots, \ell_{[k,l]} \rangle, \tag{3.3}$$

where, each annotation ℓ *spans* the positions described by the interval in subscript ($[i, j]$ with $i \leq j$). In short-hand notation, we refer to a sequence of words as, $w_{\langle i:j \rangle} = \langle w_{[i,i]}, \ldots, w_{[j,j]} \rangle$, and a sequence of annotations as, $\ell_{\langle i:l \rangle} = \langle \ell_{[i,j]}, \ldots, \ell_{[k,l]} \rangle$, where $i \leq j$, $k \leq l$, and $i < k$. A text region consists of word sequences $w_{\langle i:j \rangle}$ in a document $d \in D$ that satisfies conditions imposed by the query involving word sequences and the semantic annotations. In other words, a document is considered a match if it contains at least one text region that meets the constraints specified in the query involving word sequences and semantic annotations.

Each document is further accompanied by metadata such as its unique identifier (id) and its timestamp (ts), $d_{\mathcal{M}} = \langle id, ts \rangle$. The pairs of all valid $\langle id, ts \rangle$ pairs in the document collection (D) are denoted by $\Sigma_{\mathcal{M}}$. Figure 3.1 illustrates how the different annotation layers for part-of-speech tags ($d_{\mathcal{P}}$), named entities ($d_{\mathcal{E}}$), temporal expressions ($d_{\mathcal{T}}$), and numerical values ($d_{\mathcal{N}}$) are overlayed over a sequence of words according to our data model.

3.3.2 Query Language

We now describe the language to express queries involving regular expressions over word sequences and annotations. The complete grammar containing production rules for query generation is given in Figure 3.2. In the following

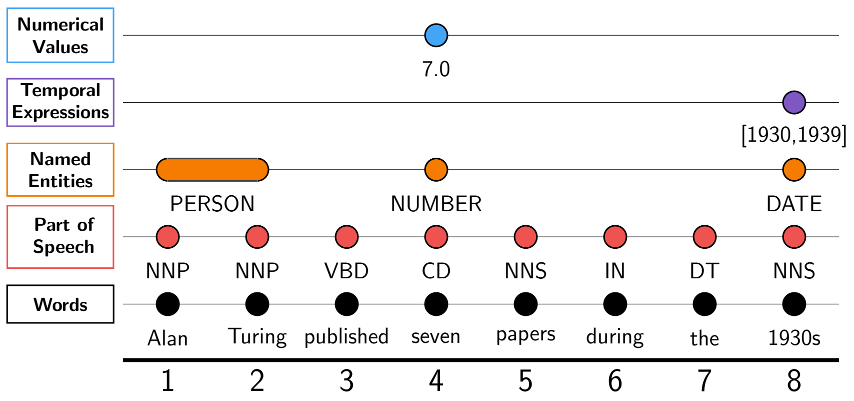


Figure 3.1: Data model showing the text & annotation layers.

paragraphs we discuss the semantics of the operators and how to form queries using this grammar.

To formalize the semantics of the operators, we use the function GYANI(Q) to represent the mapping between query Q and a set of documents $\{d_1, d_2, \ldots, d_k\}$ that satisfy the conditions of the query. A document is deemed a match if it contains a text region matching the query. A text region is considered a match to a query if and only if the sequence of elements $\ell_{\langle i:l \rangle}$ in the query occurs as a contiguous sequence in the appropriate layer of the document $d_{\mathcal{L}}$. Formally,

$$\ell_{\langle i:l \rangle} = \langle \ell_{[i,j]} \ldots \ell_{[k,l]} \rangle \sqsubset d_{\mathcal{L}}. \tag{3.4}$$

For instance, $\langle$alan turing$\rangle \sqsubset \langle$computer scientist alan turing$\rangle$, while $\langle$scientist turing$\rangle \not\sqsubset \langle$computer scientist alan turing$\rangle$.

Boolean Operators in our query language are: AND ($\boxed{\wedge}$), OR ($\boxed{\vee}$), and NEGATION ($\boxed{\neg}$). The binary operators AND and OR act on two sequences of words such that the resulting documents must contain both or either of the sequences of words respectively. The NEGATION operator, on the other hand, is a unary operator and acts on a sequence of words such that the resulting documents do not contain that sequence of words. For instance, the query: [$\langle$alan turing$\rangle$ $\boxed{\vee}$ $\langle$enigma machine$\rangle$] $\boxed{\wedge}$ $\langle$wins war$\rangle$ selects those documents that contain either of the sequences of words $\langle$alan turing$\rangle$ or $\langle$enigma machine$\rangle$ with the phrase $\langle$wins war$\rangle$.

Stack Operator. With the help of annotations, different interpretations can be stacked on top of sequences of words in the text layer. This is done with the help of the STACK ($\oplus$) operator. Thus, a particular interpretation can be attached to a sequence of words, for example: $\langle$last year$\rangle\oplus$[1918,1918], refers to those temporal expressions that resolve to the year 1918. The STACK operator is a unary operator, such that it results in only those documents that contain the sequences of words with that particular annotation attached to them. Formally, the semantics can be specified as,

$$\text{GYANI}\left(w_{\langle i:j \rangle} \oplus \ell_{[i,j]}\right) = \left\{ d \in D \,\middle|\, w_{\langle i:j \rangle} \sqsubset d_{\mathcal{V}} \wedge \ell_{[i,j]} \sqsubset d_{\mathcal{L}} \right\}.$$

For example, the query: ⟨paris hilton⟩⊕(PERSON) retrieves those documents that contain the sequence of words paris hilton annotated as a person by the named entity annotator.

Regular Expression Operators. We consider four basic regular expressions in our query language at word or annotation level. These are (based on [8]), STAR (*) that allows greater than zero repetition; PLUS (+) that allows greater than one repetition; QUES (?) that either matches a single word or annotation or nothing; and DOT (.) that acts as a placeholder for any word or annotation. We also provide the UNION (|) operator to group together results for two different word sequences. The semantics of the UNION operator is equivalent to that of the OR operator. We are particularly interested in combinations of those regular expressions that join two word or annotation sequences to match and retrieve text regions. The regular expression operators that we provide are: DOT STAR ($\boxed{.*}$), DOT PLUS ($\boxed{.+}$), DOT QUES ($\boxed{.?}$), and DOT ($\boxed{.}$) operator. We show the semantics of the DOT PLUS operator below:

$$\text{GYANI}\left(w_{\langle i:j\rangle}\boxed{.+}\,w_{\langle k:l\rangle}\right)=\left\{d\in D\;\middle|\;\begin{matrix}(w_{\langle i:j\rangle}\sqsubset d_{\mathcal{V}})\wedge\\(w_{\langle k:l\rangle}\sqsubset d_{\mathcal{V}})\wedge\\(k-j\geq 2)\end{matrix}\right\}. \tag{3.5}$$

Semantics for the other regular expression operators can be obtained similarly by adjusting the gaps between the two word or annotation sequences that are being joined (i.e., varying the distance $(k-j)$ in Equation 3.5). Concretely, for the DOT STAR $\boxed{.*}$ operator we have $(k-j\geq 1)$; for the DOT QUES $\boxed{.?}$ operator $(k-j\in[1,2])$; and for the DOT $\boxed{.}$ operator $(k-j=2)$. The operators $\boxed{.*}$ and $\boxed{.+}$ can be greedy in matching multiple sentences in a document that satisfy the positional constraints. In order, to keep only the shortest possible match, these can be turned lazy by using QUES as a flag. That is, the operators $\boxed{.*?}$ and $\boxed{.+?}$ match only the shortest positional difference.

Projection Operators. The regular expression operators, $\boxed{.*}$ and $\boxed{.+}$ will yield documents that contain text regions spanning multiple sentences. In many knowledge-centric applications it is desirable that the text regions lie within a context window of few sentences or be restricted to a single sentence. To support this operation, we propose the $k-$PROJECT operator $\boxed{\pi_k}$ where k specifies the number of sentences the positional intervals may span.

We further specifically instantiate operators that project the regular expression match for words or annotations within a sentence boundary. These operators consist of PHRASE STAR ($\boxed{\ell*}$), PHRASE PLUS ($\boxed{\ell+}$), PHRASE QUES ($\boxed{\ell?}$), and PHRASE DOT ($\boxed{\ell}$) operators. The PHRASE PROJECTION operators are obtained by binding the regular expression to a specific annotation type. The overriding constraint we impose on these operators is that the resulting positions lie within sentence boundaries in addition to the positional constraints of the regular expression. The semantics of one such instance (other instances can be derived in a similar manner) is stated below:

$$
\begin{array}{rl}
\text{OPERATOR} \rightarrow & \wedge \mid \vee \mid \neg \mid \oplus \\
\text{REG. EXPR.} \rightarrow & * \mid + \mid \ ? \mid . \mid \mid \\
\text{QUERY} \rightarrow & \mid \langle\text{SEQUENCE}\rangle \text{ OPERATOR } \langle\text{SEQUENCE}\rangle \text{ QUERY}^* \\
 & \mid \langle\text{SEQUENCE}\rangle \text{ REG. EXPR. } \langle\text{SEQUENCE}\rangle \text{ QUERY}^* \\
 & \mid \langle\text{SEQUENCE}\rangle \text{ REG. EXPR. QUERY}^* \\
 & \mid \text{REG. EXPR. } \langle\text{SEQUENCE}\rangle \text{ QUERY}^* \\
 & \mid [\,\text{QUERY}\,] \mid \langle\,\text{SEQUENCE}\,\rangle \mid \varepsilon \\
\text{SEQUENCE} \rightarrow & \Sigma^+_{\mathcal{V}} \mid \Sigma^+_{\mathcal{P}} \mid \Sigma^+_{\mathcal{E}} \mid \Sigma^+_{\mathcal{T}} \mid \Sigma^+_{\mathcal{N}}
\end{array}
$$

Figure 3.2: The query language.

$$\text{GYANI}\left(w_{\langle i:j\rangle}\boxed{\ell *}\right) = \left\{ d \in D \;\middle|\; \begin{array}{c} (w_{\langle i:j\rangle} \sqsubset d_{\mathcal{V}}) \wedge (l_{[p,q]} \sqsubset d_{\mathcal{L}}) \wedge \\ (j < p) \wedge ([i,q] \subseteq [m,n]) \end{array} \right\}.$$

where, the interval $[m, n]$ encompasses the sentence boundary containing the word sequence and the annotation in the document. We can further combine regular expressions and the projection operators to establish the following equivalence:

$$w_l \ \boxed{\ell +}\ w_r \equiv \boxed{\pi_1}\big(w_l \ \boxed{.*?}\ \ell \ \boxed{.*?}\ w_r\big). \tag{3.6}$$

3.3.3 Index Design

Next, we discuss the design and implementation aspects concerning the data model. While considering the implementation for GYANI we considered three key aspects: scalability, reliability, and compatibility. Prior work [170, 202] highlights the utility of using combinations of inverted indexes and augmented indexes (e.g., next word, phrase, or direct indexes) can provide in answering phrase queries. In particular, we base our index design on a combination of inverted and direct indexes. Since, these indexes are present in existing infrastructure [170], a complete overhaul of the indexes is not required thereby assuring compatibility of our implementation. Additionally, these indexes can be sharded (distributed) across a network of commodity hardware, making them extremely scalable and reliable.

Design Space. We now describe the design space to decide what are the appropriate indexing units for our data model. The basic queries in our language consist of word sequences, $w_{\langle i:j\rangle}$. The word sequences can be stacked with annotations, $w_{\langle i:j\rangle} \oplus \ell_{[i,j]}$, to specify semantics. Moreover, for queries involving regular expressions, we need to maintain positional constraints. Figure 3.3 summarizes the key choices. We now contemplate four different choices for designing our indexes to support the query language.

NAÏVE DESIGN. A naïve design choice is to implement GREP as-is. That is, given a query traverse the entire document collection to retrieve the required documents. This can also be achieved by a direct index, that stores the layers of annotations and word sequence against the document metadata.

INVERTED INDEX DESIGN. The naïve design can be improved by using an inverted index over elements to narrow down the search space. The inverted index structure will thus store individual elements with singleton positional offsets, to indicate unit intervals. As shown in Figure 3.3, this design choice corresponds to indexing units being unigrams $w_{[i,i]}$ or annotations $\ell_{[i,i]}$ spanning unit intervals. The types of queries that can be answered using an inverted index built on these units are Boolean and wildcard combinations of unigrams or single annotations. However, there are three shortcomings of adopting this design choice. First, this approach is *lossy*, as the positional information conveyed by intervals is lost. That is, a named entity with positional interval $[i, j]$ is not equivalent to $\{i\}, \{i+1\}, \ldots, \{j\}$. For instance, the annotation $(\text{PERSON})_{[1,2]}$ conveys that the words at positions $w_{[1,1]}$ and $w_{[2,2]}$ is one person as opposed to $(\text{PERSON})_{[1,1]}$ & $(\text{PERSON})_{[2,2]}$ indicative of two different persons. Second, retrieving word sequences incurs a high computational cost as the number of calls to the index is proportional to the size of the sequence. Third, word sequences stacked with annotations can not be answered in this design choice. These issues can be mitigated to some degree by combining the inverted index with the direct index. This combined design mimics the choice made by Cafarella and Etzioni [51] for their neighbor index. The neighbor index modeled an inverted index over unigrams and then wraps the annotation layer for the document containing the unigram as an additional payload to the posting list.

K-FRAGMENT DESIGN. Going beyond single elements as indexing units, we can decide on their size with respect to our annotation choices. Since, the annotations for PoS tags, named entities, temporal expressions, and numerical values incrementally build on each other, they share positional information based on the word sequences which they annotate. For example, in Figure 12.2, the word sequence alan turing is annotated with PoS tags $\{(\text{NNP})(\text{NNP})\}$ and named entity tag (PERSON). Thus, we can treat the layers of annotation that build on the PoS tag as a fragment or as an indivisible unit to index. We refer to this indivisible unit of variable word sequence attached to its $k-1$ annotation layers as a *k-fragment*. In this design space, semantic queries can be answered, if and only if the variable-length word sequence and all the attached annotations for it are known to the user. While, maintaining complete fragments in an inverted index is cheap, it poses little utility since the length of the word sequences accompanying a particular annotation is variable. Therefore, both these design choices are ill suited for implementing our data model. We thus look at the design space between these two extremes, which is discussed next.

GYANI DESIGN. We have already considered the extremes of indexing word sequences as single words to variable length fragments. A better design choice, is thus to consider word sequences as combinations of fixed size n-grams. The N-GRAM INDEX, consists of n-grams pointing to a list of postings that contain the metadata ($\langle id, ts \rangle$) and a list S of positional spans ($s = [i, j]$) of the document in which it occurs. Specifically, we construct unigram ($n = 1$), bigram ($n = 2$) and trigram ($n = 3$) indexes to retrieve word sequences. In addition, we construct ANNOTATION INDEXES to maintain the locations of semantic annotations in the document collection. To index the variable-length word sequences with their annotations, we consider *2-fragments* that are pairwise combinations of the word sequences with one of the attached annotations. These, binary fragments are the basic indexing units for the 2-FRAGMENT INDEXES. To locate sentence boundaries and maintaining positional constraints we index all layers in a DIRECT INDEX.

Knowledge-centric tasks such as information extraction rely on extraction templates leveraging regular expressions between annotations and word sequences. Processing regular expression queries involving word sequences and annotations can be extremely expensive if only ANNOTATION INDEXES are utilized, as the posting lists for each annotation can span the entire document collections (e.g., ⟨(LOCATION) city of (LOCATION)⟩). This is a problem similar to indexing stopwords in document collections. In order to execute such complex queries, we additionally index combinations of elements across layers that are shifted. Indexing units that arise from combination of word sequences and ordered co-occurring annotations from $k - 1$ different layers are termed as as *k-stitches*. For example by combining a word sequence and named entity we can create a 2-stitch: ⟨city of (LOCATION)⟩. Thus, we additionally create 2-STITCH INDEXES that record pair-wise ordered co-occurrences of unigrams, bigrams, and trigrams with annotations within sentence boundaries. These ordered co-occurring combinations of word sequences and annotations convey relations similar to those obtained by dependency parsing. A relation in a dependency parse tree connects within a sentence two words with a particular relationship which is a subset of the combinations modeled by our 2-stitch indexes.

We further need to store information regarding sentence boundaries in order to restrain the text regions obtained from the indexes. Sentence boundaries can be obtained by retrieving them from the direct index (where they are stored as separate annotations). Or they can be added as an additional payload to the N-GRAM INDEXES. For the latter case, we make the design choice to store sentence numbers that further allows us to easily compute relaxations to context windows of more than one sentence.

3.3.4 Query Processing

We now discuss how the indexes in GYANI are combined to retrieve documents for queries expressed in our language.

Retrieving Word & Annotated Sequences. We first illustrate how the basic tokens in our query language are retrieved. Word sequences are retrieved from an N-GRAM INDEX, which requires normalizing the requested word sequence into a list of n-grams and subsequently merging their positions.

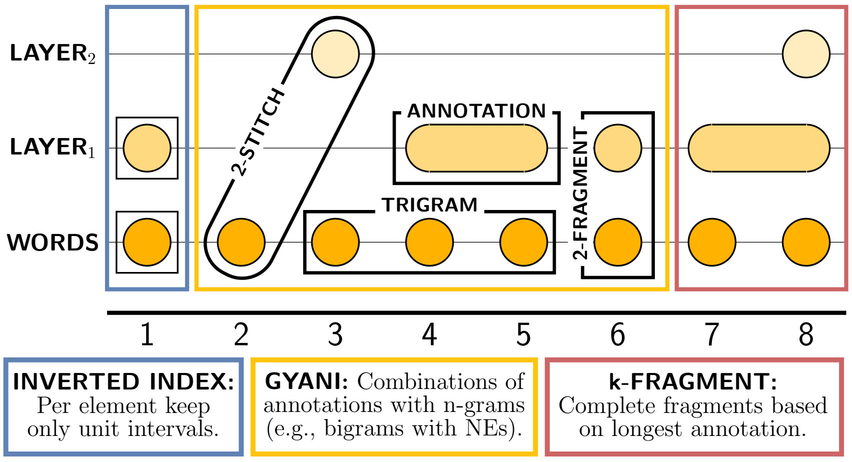

LAYER$_2$
LAYER$_1$
WORDS
2-STITCH
ANNOTATION
TRIGRAM
2-FRAGMENT
1
2
3
4
5
6
7
8
INVERTED INDEX: Per element keep only unit intervals.
GYANI: Combinations of annotations with n-grams (e.g., bigrams with NEs).
k-FRAGMENT: Complete fragments based on longest annotation.

Figure 3.3: Design space.

This method of retrieving any arbitrary length word sequence is shown in Algorithm 1. The retrieval of annotated word sequences using the STACK operator requires directly querying the 2-FRAGMENT INDEX, and no additional processing is required.

Processing Regular Expressions. We now consider how to process queries involving regular expressions operators. In particular, we consider the operator $\boxed{.+}$ in Algorithm 2. The operators $\boxed{.*}$, $\boxed{.?}$, and $\boxed{.}$ can be implemented similarly. The $\boxed{.+}$ operator is a binary operator that consumes as its left and right operand two posting lists. The output list contains postings indicating each occurrence of the element in the left operand is succeeded by the element in the right operand. This join operation is specified with the help of the operator $\bowtie_{\boxed{.+}}$. The $\bowtie_{\boxed{.+}}$ operator takes as input the positions of the elements in the left S_l and right operand S_r, and the gap constraint Δ. The gap constraint Δ indicates the permissible interval size between the elements of the left and

Algorithm 1: Processing word sequences.

```
    Input  : $Q = \langle w_1 \dots w_k \rangle$
    Output : Posting List, $L$, containing document metadata $\langle id, ts \rangle$ containing
             $Q$ and its positions $s = [i, j]$.
1   Function NGramQuery($Q$)
2       $N \leftarrow$ generate a list of n-grams from $Q$
3       $L \leftarrow$ retrieve posting list for $N[0]$ from n-GRAM INDEX
4       $L' \leftarrow \varnothing$                                  // temporary variable
5       for ($i \leftarrow 1; i < (k - n + 1); i++$) do
6           $L' \leftarrow$ retrieve posting list for $N[i]$
7           $L \leftarrow$ merge positions for postings in $L$ & $L'$ s.t. each position for a
              posting in $L$ is before the position for the same posting in $L'$
8       return $L$
```

right operand. For the operator $\boxed{.+}$, the gap constraint Δ is equal to 2. For the operators $\boxed{.*}$, $\boxed{.?}$, and $\boxed{.}$ the constraints are, $\Delta \geq 1$, $\Delta \in [1, 2]$, and $\Delta = 2$.

Processing Projection Operators. We now consider how to process queries involving PHRASE PROJECTION operators that project the regular expression match to within a sentence. Processing these operators can be done using a combination of different indexes.

First, we can process PHRASE PROJECTION operators using a combination of DIRECT and N-GRAM indexes. Here, the DIRECT index is used to identify sentence boundaries and to identify the positions of the annotations in its respective layer of a document. Since, these regular expressions with annotations or words return results at the sentence level, we instantiate each operator to expand the suffix ($\boxed{\blacktriangleright}$); or the prefix ($\boxed{\blacktriangleleft}$); or both suffix and prefix ($\boxed{\blacklozenge}$) of the attached word sequence. For instance, for the query ⟨war started on⟩(DATE), involves processing the word sequence ⟨war started on⟩ and selecting those sentences in which it occurs with a date as a suffix. While, the query (WORD)+ ⟨war started on⟩ (DATE) involves expanding both a prefix and suffix expansion. The processing for the suffix expansion of $\boxed{\ell+}$ operator is shown in Algorithm 3. The operators $\boxed{\ell*}$, $\boxed{\ell?}$, and $\boxed{\ell}$ and the associated type of expansion are implemented in a similar manner.

Second, we can implement the query processing for PHRASE PROJECTION operators using 2-STITCH and N-GRAM indexes. The processing of these operators is done by first looking up the ordered co-occurrences of the word sequences and annotations from the 2-STITCH indexes and then merging them based on overlaps in the text regions they share within the same sentence. For example, for the query ⟨ (ORGANIZATION) acquired the start-up for (MONEY) ⟩, we can lookup the 2-stitches ⟨ (ORGANIZATION) acquired the start-up for⟩ and ⟨ acquired the start-up for (MONEY) ⟩ and merge them based on overlapping text regions.

Third and finally, we can use a combination of ANNOTATION, N-GRAM, and DIRECT indexes for processing PHRASE PROJECTION operators. To do so, we can compute a regular expression join between the word sequence obtained from the N-GRAM index and annotation obtained from the ANNOTATION index. After the join, we can restrict the text region to within one sentence using the $\boxed{\pi_1}$ operator. The $\boxed{\pi_k}$ operator restricts the positional spans of a posting list that are input to the operator to lie within a span of k sentences. The implementation for this operator is given in Algorithm 4. For instance, the following query $\boxed{\pi_1}\big($(LOCATION) $\boxed{.*?}$⟨declared war on⟩$\big)$, yields documents where the resulting text regions after the $\boxed{.*?}$ operation lie within a sentence of the retrieved documents.

Algorithm 2: Processing the $\boxed{.+}$ operator.

```
   Input  : Posting lists $L_l$ and $L_r$ corresponding to the left and right
            operands of $\boxed{.+}$ operator, respectively.
   Output : Posting List, $L \leftarrow L_l \boxed{.+} L_r$.
1  Function $\boxed{.+}(L_l,\ L_r)$
2      $R \leftarrow$ find all common metadata for postings in $L_l$ & $L_r$
3      $L \leftarrow \varnothing$, $S \leftarrow \varnothing$
4      foreach $\langle id, ts \rangle \in R$ do
5          $S \leftarrow \bowtie_{\boxed{.+}}$(positions for $\langle id, ts \rangle$ in $L_l$, positions for $\langle id, ts \rangle$ in $L_r$, 2)
6          $L \leftarrow L$.append(new $\langle\ \langle id, ts \rangle, S\ \rangle$)
7      return $L$
8  Function $\bowtie_{\boxed{.+}}(S_l,\ S_r,\ \Delta)$
9      $S \leftarrow \varnothing$
10     if the last interval in $S_r$ lies before the first interval in $S_l$ then
11         return $S$
12     if first interval in $S_r$ is before the first interval in $S_l$ then
13         $S_r \leftarrow$ remove intervals from the front of $S_r$ until the first interval in
             it is after the first interval in $S_l$
14     for $(i \leftarrow 0; i < |S_l|; i++)$ do
15         for $(j \leftarrow 0; j < |S_r|; j++)$ do
16             if $(S_l[i]$ is before $S_r[j]) \wedge (S_r[j].end - S_l[i].begin \geq \Delta)$ then
17                 $S \leftarrow S$.append($[S_l[i]$.begin, $S_r[j]$.end$]$)
18     return $S$
```

Algorithm 3: Processing the $\boxed{\blacktriangleright +}$ operator.

```
Input  : Posting list $L$.
Output : Expanded Posting List, $L$, using suffix expansion.
1  Function $\boxed{\blacktriangleright +}(L)$
2      $S \leftarrow \varnothing$                                // holds the annotation layer
3      $B \leftarrow \varnothing$                                // holds the sentence boundaries
4      $N \leftarrow \varnothing$                                // holds the new positions after expansion
5      $count \leftarrow 0$                                      // holds the annotation count
6      foreach posting $P$ in list $L$ do
7          $S \leftarrow$ retrieve the annotation layer containing $\ell$ for the metadata
             $\langle id, ts \rangle$ using the DIRECT INDEX
8          $B \leftarrow$ retrieve the sentence boundaries for the metadata $\langle id, ts \rangle$ using
             the DIRECT INDEX
9          $count \leftarrow 0$, $N \leftarrow \varnothing$
10         foreach position interval $[x, y]$ in posting $P$ do
11             $count \leftarrow$ count the number of annotations that are equal to $\ell$
                 between the positional interval spanning $y$ and the end of the
                 sentence using $S$ and $B$
12             if $count > 1$ then
13                 $N \leftarrow N$.append(expand $[x, y]$ with position of the
                     annotations found between $y$ and the end of sentence)
14         replace positions of $P$ with expanded positions $N$
15     return $L$
```

Algorithm 4: Processing $\boxed{\pi_k}$ operator.

```
Input  : Posting List $L$ and sentence window $k$.
Output : Posting List $L'$ such that each position lies within a $k$ sentence
         window.
1  Function $\boxed{\pi_k}(L,\ k)$
2      foreach posting $P$ in list $L$ do
3          $S \leftarrow \varnothing$                          // modified positions for $P$
4          $B \leftarrow$ retrieve the sentence boundaries for the metadata $\langle id, ts\rangle$ using
            the DIRECT INDEX
5          if $k > 1$ then
6              $B \leftarrow$ coalesce $(B,\ k)$
7          foreach position interval $[x, y]$ in posting $P$ do
8              foreach position interval $[m, n]$ in $B$ do
9                  if $[m, n]$.contains($[x, y]$) then
10                     $S \leftarrow S$.append($[x, y]$)
11         replace positions of $P$ with modified positions $S$
12     return $L$
13 Function coalesce($B,\ k$)
14     $S \leftarrow \varnothing$                                  // coalesced positions
15     $b, e \leftarrow -1$                               // begin and end for intervals
16     for $(i \leftarrow 0; i < |B|.size - (k-1); i++)$ do
17         $b \leftarrow B[i].begin$
18         $e \leftarrow B[i+k-1].end$
19         $S \leftarrow S$.append($[b, e]$)
20     return $S$
```

COLLECTION	SIZE (GB)	$n_{\text{documents}}$	n_{words}	$n_{\text{sentences}}$
NEW YORK TIMES	49.7	1,855,623	1,058,949,098	54,024,146
WIKIPEDIA	156.0	5,327,767	2,807,776,276	192,925,710
GIGAWORD	193.6	9,870,655	3,988,683,648	181,386,746

COLLECTION	$n_{\text{part-of-speech}}$	$n_{\text{named entity}}$	n_{time}	n_{numbers}
NEW YORK TIMES	1,058,949,098	107,745,696	15,411,681	21,720,437
WIKIPEDIA	2,807,776,276	444,301,507	97,064,344	82,591,612
GIGAWORD	3,988,683,648	517,420,195	72,247,124	102,299,554

Table 3.1: Document collection statistics. The table shows the sizes of annotated collections as well as annotation statistics.

3.4 Evaluation

We now describe the evaluation setup of the experiments that includes a description of the document collections. We then show how the testbeds for the knowledge-centric tasks were constructed. Finally, we discuss the results obtained for the experiments.

3.4.1 Setup

Document Collections. We considered three large document collections to index with GYANI. Statistics regarding the collections are summarized in Table 3.1. The first document collection is the New York Times, which comprises of news articles published over a twenty year (1987-2007) time period [18]. The second document collection is the English Gigaword containing articles collected from seven distinct English news publishers over a sixteen year (1995-2010) time period [5]. The third and final document collection is the entire English Wikipedia [23] (we use the snapshot available on March 13th, 2017). An important aspect of all the aforementioned document collections is that, they are written in well poised grammar and language, so that the automated annotations obtained via NLP tools are of high quality.

Annotating Document Collections. Each document in the collections was annotated with four different types of semantic annotations. We utilized the Stanford Core NLP [151] toolkit to annotate the documents with part-of-speech, named entities, temporal expressions, and numerical quantities. The processing for the documents was done as follows. First, each document's text content is created by concatenating the headline, the article body, and other auxiliary keyword or classification terms provided as metadata into one long document string. Second, the publication date for the news article is obtained; for Wikipedia pages we used the document creation time. Third, the document string is fed into the annotation pipeline, which performs sentence boundary detection, tokenization, and tags each token with the aforementioned types of annotations. Fourth, for each token we analyze the type of annotation performed and subsequently create the layer elements. For the parts of speech tags we keep the tag as the annotation element and unit interval spans as positions. The Stanford Core NLP named entity annotator provides ten different classes of entities. We divide these classes as per our requirement. For the named entity layer, we consider all the class tags as annotation elements: PERSON, ORGANIZATION, LOCATION, DATE, TIME, DURATION, MONEY, PERCENT, NUMBER, and ORDINAL. For the temporal expressions, we consider for resolution the classes: DATE, TIME, and DURATION. For the numerical quantities we consider for resolution the classes: MONEY, PERCENT, NUMBER, and ORDINAL. Finally, the document string, along with the strings containing the annotation layers as per our data model are stored together with the metadata information containing the timestamp (determined using publication date or creation time) and its identifier (determined using the hash of the publisher supplied identifier string or the title string).

Implementation Details. The implementation of the entire infrastructure was done in Java. All document processing and indexing was done in a distributed manner over a cluster of twenty machines running the Cloudera CDH 5.90 distribution of Hadoop. All machines in the cluster were equipped with Intel Xeon CPUs with up to 24 cores and a clock speed of up to 3.50 GHz, up to 128 GB of primary memory, and up to eight 4 TB hard disks as secondary storage. We utilized the 1.2.0 CDH 5.9.0 version of HBase for implementing our indexes.

GYANI Indexes. We instantiated the index types discussed in Section 3.3.3 for each of the document collections. Posting lists are compressed using the PForDelta compression technique [11]. We summarize the index sizes for the various types in Table 3.2.

INDEX TYPE	NYT	WIKIPEDIA	GIGAWORD
DIRECT	18.80	44.80	52.40
N-GRAM	45.90	126.30	154.40
ANNOTATION	2.39	7.65	9.33
2-FRAGMENT	6.30	23.10	24.16
2-STITCH	141.00	473.00	542.40

Table 3.2: Index sizes in Gigabytes (GB).

3.4.2 Knowledge-Centric Tasks

We next describe the structure of the queries for the five knowledge-centric tasks used in our evaluation.

Information Extraction (IE) Task. To construct information extraction templates, we utilize the paraphrases of relations [162] present in the Yago knowledge graph. The information extraction templates are constructed as follows. First, for each of the Yago relations the domain of the subject and the range of the object is identified. For example, for the predicate wasBornIn the domain of the subject is PERSON and the range of the object is LOCATION. Second, for the given relation, we look up how the relation is expressed in text using a paraphrase dictionary [162]. For example, the paraphrases for wasBornIn are grew up, returned to, and raised in. Finally, we combine the subject, paraphrase of the relation, and the object to form the information extraction template. For instance, a template for the relation wasBornIn is: (PERSON) ⟨raised in⟩ (LOCATION).

Relation Extraction (RE) Task. The aim of the relation extraction task to identify the textual patterns of the predicate given its subject and object arguments. From the paraphrase dictionary [162], we also have concrete instances of subject-object pairs identified in the New York Times and Wikipedia. However, most of the named entities can be expressed in myriad surface forms. In order to capture the different surface forms for a given named entity we turn to the redirectedFrom relation in the Yago knowledge graph. To create queries for this task we proceed as follows. First, we distill the unique instances of the subject-object pairs identified by [162] in both news and encyclopedic sources. Second, we look up surface forms for the named entities contained in the subject and object arguments from Yago's redirectedFrom relation. Third, we combine the named entity and its various surface forms as UNION wildcard clause, e.g., [kennedy | jfk]. Finally, we construct the relation extraction template by combining the subject and objects arguments with $\boxed{\ell *}$ operator. For example, [john f. kennedy | jfk | john fitzgerald kennedy] (WORD)* [ronald reagan | 40th president of the united states | ronnie reagan].

TASK	n_{query}	μ_{word}	$\mu_{\text{annotation}}$
TASK-IE	56,261	2.50	2.22
TASK-RE-NEWS	116,157	83.77	0.00
TASK-QA-NEWS	828,208	37.68	1.60
TASK-FS-NEWS	1,618,377	126.61	0.83
TASK-RE-WIKI	861,235	67.89	0.00
TASK-QA-WIKI	18,151,907	19.47	1.48
TASK-FS-WIKI	26,164,545	81.32	0.69
TASK-SQ	4,589	6.15	2.53

Table 3.3: Testbed statistics.

Question Answering (QA) Task. The aim of the question answering task is to retrieve sentences as candidate answers in response to a query with annotation wildcards. For this task we consider the textual patterns of the predicates that occur between the subject-object instances, also available from the dataset in [162]. To create the queries for this task we carry out the following steps. First, we consider only the subject's named entity and its surface forms from the subject-object pairs. Second, we combine the textual pattern detected as a predicate for the given subject-object pair [162]. Finally, we replace the object with its appropriate range type with a $\boxed{\ell}$ annotation wildcard operator. For example, [microsoft | office corporation] to work closely with (ORGANIZATION).

Fact Spotting (FS) Task. The aim of the fact spotting task is to retrieve sentences that are evidences of facts from a knowledge graph. The process of creating queries for this task is similar to that of question answering task except that we keep the named entity and its surface forms for the object argument. An example of a query in this task is [microsoft | office corporation] to crush [netscape | devedge].

Semantic Search (SQ) Task. The aim of the semantic search task is to demonstrate the ability to express queries containing word sequences adorned with a semantic meaning. To create queries for this task, we turn to a compendium of important events compiled by the New York Times called "On this Day" events [19]. Each event in this compendium consists of a date and an accompanying textual description. The steps involved in creating the queries are as follows. First, each of the event descriptions are run through an annotation process similar to the one applied to the document collections. Second, we combine annotations and word sequences from the named entity, time, and numerical quantity layers to form stacked phrases. Finally, we combine them with the Boolean $\boxed{\wedge}$ operator to form the semantic query. For example, (mohandas k. gandhi)$\oplus$(PERSON) $\boxed{\wedge}$ (india)$\oplus$(LOCATION).

In Table 3.3 we summarize the entire testbed statistics. As can be noticed from query length in Table 3.3, the queries in our testbed are quite complex, lengthy, and verbose. The dataset is publicly available at the following URL:

`http://resources.mpi-inf.mpg.de/dhgupta/data/cikm2018/`.

3.4.3 Results

We evaluate GYANI for efficiency by measuring end-to-end query execution times. For each of the task we sampled 100 queries from the appropriate testbed for evaluation. Each sample is executed three times and the average execution time is reported. We execute each task under two settings: warm and cold caches. In the warm cache setting, each query is executed once to bring the relevant posting lists into the main memory of the HBase cluster and then executed three more times to measure its execution time. In the cold cache settings, the sample of queries is executed three times by shuffling the order of query execution between rounds. The time measured consists of retrieving the posting lists from HBase and further performing the necessary operations dictated by the tasks, which may further involve accessing the direct index. In order to minimize interruptions due to garbage collection we utilize the concurrent garbage-first garbage collector (`G1GC`). Experiments are run on two servers capable of handling high I/O bound jobs as our front-end and the Hadoop cluster acting as our back-end storage. Each server consists of up to two Intel Xeon processors with up to 96 cores, clocked at 2.66 GHz and up to 1.48 TB of primary memory.

Baselines. The baselines we evaluate are aligned with respect to the design choices explored in Section 3.3.3. We first measure the time to scan the entire document collection without any indexes. This NAÏVE DESIGN thus establishes a lower bound to execute a single query by finding the pattern in the entire document collection. This simple design thus imitates GREP in an embarrassingly parallel manner. The first baseline implements the INVERTED INDEX DESIGN and is denoted by TEXTI. This baseline considers only the word sequences that can be obtained by combining N-GRAM indexes and the DIRECT index. With TEXTI we test how efficiently an infrastructure can retrieve candidate documents relying only on text. To a certain extent, TEXTI simulates the "neighbor index" [51], where we are forced to access the DIRECT index to match the context around the words during query processing. To execute the queries with N-GRAM indexes and DIRECT index, we identify the sentences (using the DIRECT index) containing the text only arguments of the query and apply the AND operator between the obtained posting lists to obtain the final result. The second baseline considers N-GRAM indexes, ANNOTATION indexes, and DIRECT index to evaluate regular expression queries. We call this baseline ANNI. The ANNI baseline considers posting lists for annotations when evaluating regular expression queries for the knowledge-centric tasks. It additionally resorts to the DIRECT index for identifying sentence boundaries when restraining the results to within one sentence. We evaluate our infrastructure GYANI that leverages the complete set of indexes proposed. With GYANI, however, we leverage the sentence identifiers stored within the N-GRAM indexes to restrict the regular expression matches to within one sentence. In order to execute the same set of queries for all the three infrastructures, we choose those queries where the predicate does not contain any annotation wildcards and the subject and object regular expressions are either (PERSON), (ORGANIZATION), or (LOCATION). To execute the queries against TEXTI we replace (PERSON), (ORGANIZATION), or (LOCATION) operator with (WORD)+. For the semantic query task, only the TEXTI baseline is applicable where only the N-GRAM indexes are used to execute the word-only versions of the semantic queries.

Results. We now discuss the results for TEXTI, ANNI, and GYANI at the five different tasks over three document collections.

Grepping the Entire Document Collection. We first report the results for the baseline GREP that involves scanning (i.e., matching the $\boxed{.*}$ operator) the entire document collection on our Hadoop cluster. This is akin to running GREP as an embarrassingly parallel task per query over a large document collection. We report the time taken to scan the three different document collections for a single query in Table 3.4. The time required for scanning the document collections is proportional to the collection's size. The minimum amount of time was required for the New York Times which is the smallest collection amongst the three. While, Gigaword required the most time as it was the largest amongst the three collections.

NEW YORK TIMES	WIKIPEDIA	GIGAWORD
111.00	322.00	396.00

Table 3.4: GREP baseline times in seconds.

End-to-end Query Execution Times. We now discuss the results obtained for query execution times over the three different document collections. The results are reported in Table 3.5 are in seconds. Note that our system and the baselines shown in Table 3.5 retrieve equivalent sets of text regions as results. All values marked with △ and ▲ indicate statistically significant results ($p \leq 0.05$) with respect to TEXTI and ANNI respectively. The significance was measured using the paired t-test. For the information extraction task we can see that our proposed infrastructure GYANI drastically brings down

	TASK	TEXTI (COLD) (s)	ANNI (COLD) (s)	GYANI (COLD) (s)
NEW YORK TIMES	IE	8.38 ± 20.61	12.32 ± 12.41	△▲ 0.01 ± 0.02
	QA	9.68 ± 18.11	9.18 ± 0.82	△▲ 0.15 ± 0.16
	FS	7.10 ± 34.49	0.29 ± 0.57	△ 0.29 ± 0.58
	RE	41.92 ± 122.89	△ 2.75 ± 9.98	△ 2.41 ± 8.30
	SQ	1.22 ± 3.96	–	0.69 ± 2.98
	TASK	TEXTI (WARM) (s)	ANNI (WARM) (s)	GYANI (WARM) (s)
NEW YORK TIMES	IE	3.53 ± 10.97	11.55 ± 11.15	△▲ 0.01 ± 0.01
	QA	4.81 ± 9.70	9.13 ± 0.42	△▲ 0.09 ± 0.15
	FS	4.39 ± 21.79	0.30 ± 0.55	△ 0.29 ± 0.51
	RE	29.60 ± 111.56	△ 2.73 ± 9.90	△ 2.42 ± 8.25
	SQ	0.96 ± 2.98	–	0.86 ± 3.61
	TASK	TEXTI (COLD) (s)	ANNI (COLD) (s)	GYANI (COLD) (s)
WIKIPEDIA	IE	17.73 ± 35.35	51.52 ± 33.08	△▲ 0.11 ± 0.25
	QA	21.10 ± 73.18	28.30 ± 19.33	△▲ 0.21 ± 0.63
	FS	5.76 ± 38.32	0.46 ± 0.96	0.46 ± 0.95
	RE	105.31 ± 298.58	△ 2.50 ± 7.00	△ 2.16 ± 5.81
	SQ	2.64 ± 4.50	–	2.50 ± 6.61
	TASK	TEXTI (WARM) (s)	ANNI (WARM) (s)	GYANI (WARM) (s)
WIKIPEDIA	IE	7.18 ± 14.88	43.69 ± 19.37	△▲ 0.06 ± 0.16
	QA	8.25 ± 34.22	25.92 ± 1.55	△▲ 0.14 ± 0.55
	FS	2.49 ± 14.33	0.49 ± 0.99	0.46 ± 0.93
	RE	39.73 ± 113.50	△ 2.36 ± 6.42	△ 2.15 ± 5.81
	SQ	2.58 ± 4.31	–	△ 1.43 ± 2.70
	TASK	TEXTI (COLD) (s)	ANNI (COLD) (s)	GYANI (COLD) (s)
GIGAWORD	IE	36.69 ± 88.43	65.10 ± 51.88	△▲ 0.07 ± 0.10
	QA	57.78 ± 109.89	43.93 ± 2.99	△▲ 0.39 ± 0.54
	FS	52.41 ± 212.42	△ 1.31 ± 2.58	△▲ 1.25 ± 2.47
	RE	316.52 ± 1048.42	△ 19.33 ± 83.11	△ 15.69 ± 61.45
	SQ	5.25 ± 7.82	–	△ 3.65 ± 5.96
	TASK	TEXTI (WARM) (s)	ANNI (WARM) (s)	GYANI (WARM) (s)
GIGAWORD	IE	12.44 ± 33.78	61.88 ± 33.04	△▲ 0.13 ± 0.96
	QA	19.15 ± 38.10	43.11 ± 2.43	△▲ 0.31 ± 0.50
	FS	32.75 ± 172.51	1.27 ± 2.51	1.26 ± 2.46
	RE	256.10 ± 1047.16	△ 18.11 ± 75.08	△ 15.67 ± 61.13
	SQ	5.21 ± 7.23	–	△ 3.48 ± 5.56

Table 3.5: Query execution times in seconds.

execution times from several seconds (several minutes in case of Gigaword) to within milliseconds per query. The drastic decrease in execution time can be attributed to the observation that GYANI relies on the 2-STITCH indexes and does not resort to ANNOTATION indexes (which ANNI does) and DIRECT index (which both ANNI and TEXTI do). For the question answering task, our proposed approach again delivers results within milliseconds as compared to the other two baselines. The gains again can be attributed to the same observation as with the IE task. For the relationship extraction and fact spotting task the performance of GYANI is better or at par with ANNI. However, compared to TEXTI the query execution costs are brought down from several minutes to few seconds. The gain that GYANI attains over the other baselines is due to the fact that it does not resort to the DIRECT index for identifying sentence boundaries (it uses the sentence numbers available within the N-GRAM indexes) when evaluating the regular expressions between the query arguments. For the semantic query task, we can see that by directly leveraging the 2-fragment indexes GYANI identifies the result more quickly than the TEXTI baseline, which uses only N-GRAM indexes, as it can not disambiguate their semantics.

Summary. Summing up our experimental results across tasks and document collections, we observe that the indexes that constitute GYANI consume at most 5.62× the space required by the uncompressed semantically annotated document collections. SQ and FS are the tasks that profit least with speed ups in response time of 1.12× and 5.41× respectively. For IE, QA, and RE, as more complex tasks, GYANI achieves impressive speed ups of at least 95.69×, 53.44×, and 12.23×, respectively. We designed GYANI as a versatile tool supporting various knowledge-centric tasks. We point out that, should only specific tasks need to be supported, a subset of indexes would suffice.

3.5 GYANI Demonstration

The graphical user interface (GUI) for GYANI is shown in Figure 3.4. The GUI features a search bar in which the user can enter the query using the operators described in Section 3.3.2. The user can then select from the different document collections (see Table 3.1) indexed with GYANI to retrieve the relevant text regions. The retrieved text regions are shown in the results page on the right-hand side of the GUI. For each text region we show the named entity annotation layer by highlighting the word sequences which have been annotated by the Stanford's CoreNLP named entity recognizer.

Target Users for our demonstration are linguists, journalists, and scholars in humanities. Often for them structured search capabilities are not offered by the most accessible commercial search engines. Structured search in large document collections is crucial for them. For instance, when performing fact checking, journalists often need to verify claims involving named entities, temporal expressions, and numerical values. In such instances, the knowledge-centric tasks discussed in Section 3.1 naturally arise.

The prototype system allows the user to interactively formulate queries involving regular expressions, word sequences, and semantic annotations for the five knowledge-centric tasks (i.e., IE, QA, RE, FS, and SQ) and obtain text regions from the indexed document collections. As an example scenario, a user can query GYANI to retrieve all evidences that detail acquisitions by Google:

$$[\,\mathtt{google}\,|\,\mathtt{search\ giant}\,]\boxed{.*}[\,\mathtt{invested\ in}\,|\,\mathtt{acquired}\,]\boxed{.*}\textsf{ORG}.$$

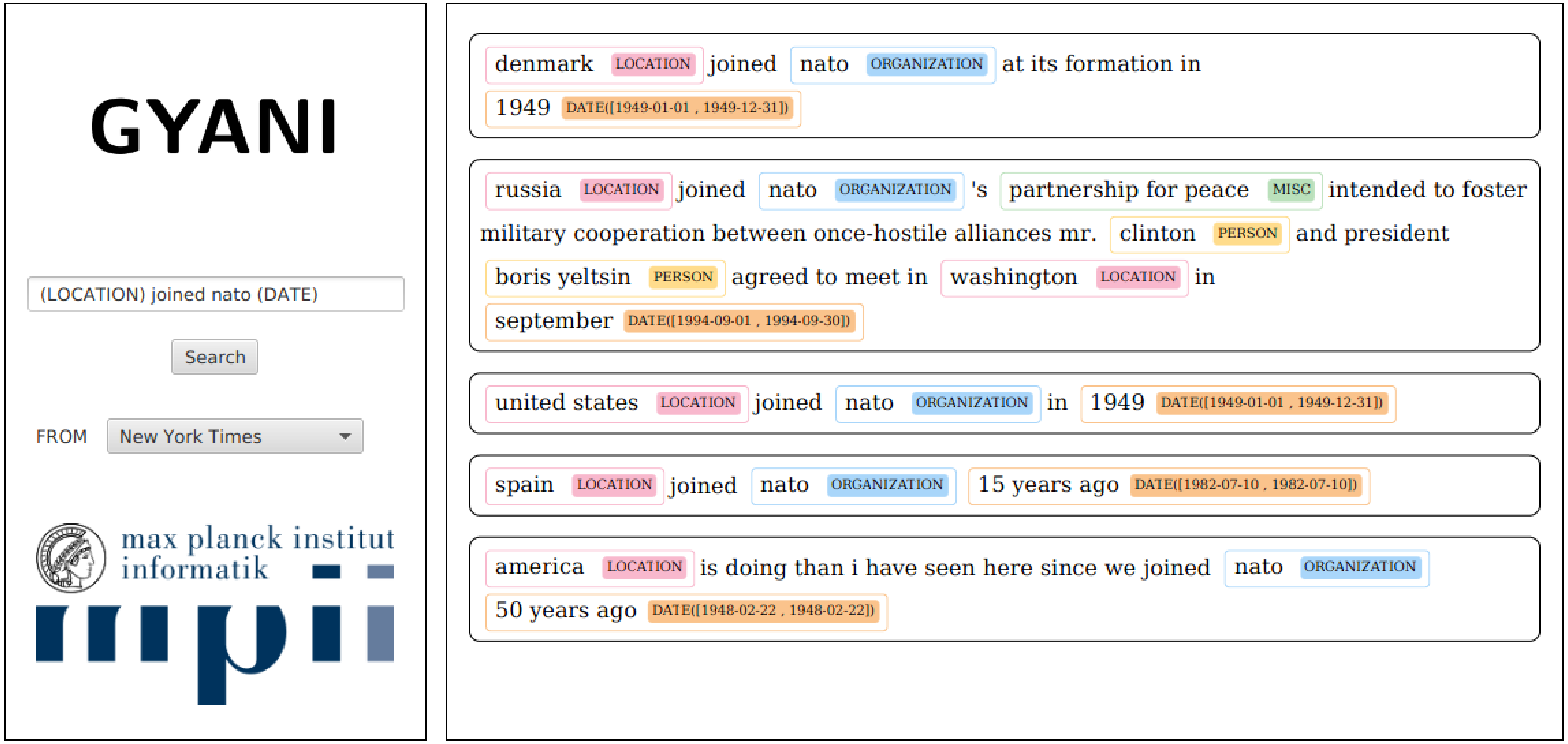
Furthermore, to identify a chronology of important events mentioned in news regarding Silicon Valley (the location and not the television series) startups with monetary values, the attendee can formulate the following query:

$$\textsf{DATE}\quad\boxed{.*}\quad\langle\mathtt{silicon\ valley}\rangle\oplus\textsf{LOCATION}\quad\boxed{.*}\quad\textsf{MONEY}.$$

Query examples with retrieved text regions are shown in Figures 3.4, 3.5, 3.6, 3.7, and 3.8. Using our prototype implementation, we show that knowledge acquisition using GYANI's structured search capabilities can indeed be done interactively and at scale across millions of documents.

3.6 Conclusion

In this chapter, we described GYANI, an infrastructure for supporting sophisticated knowledge-centric tasks at scale. We first proposed a novel data model that accommodates word sequences and layers of semantic annotations associated with them. We then proposed a novel language that allows the user to express queries consisting of regular expressions over word sequences and annotations. To allow for fast query execution times, we further described the appropriate indexes to support our query language in a complex design space. Finally, our experimental results over five knowledge-centric tasks show the ability of GYANI to efficiently support search and analysis of large semantically annotated document collections for knowledge acquisition at scale.



denmark LOCATION joined nato ORGANIZATION at its formation in 1949 DATE([1949-01-01 , 1949-12-31])

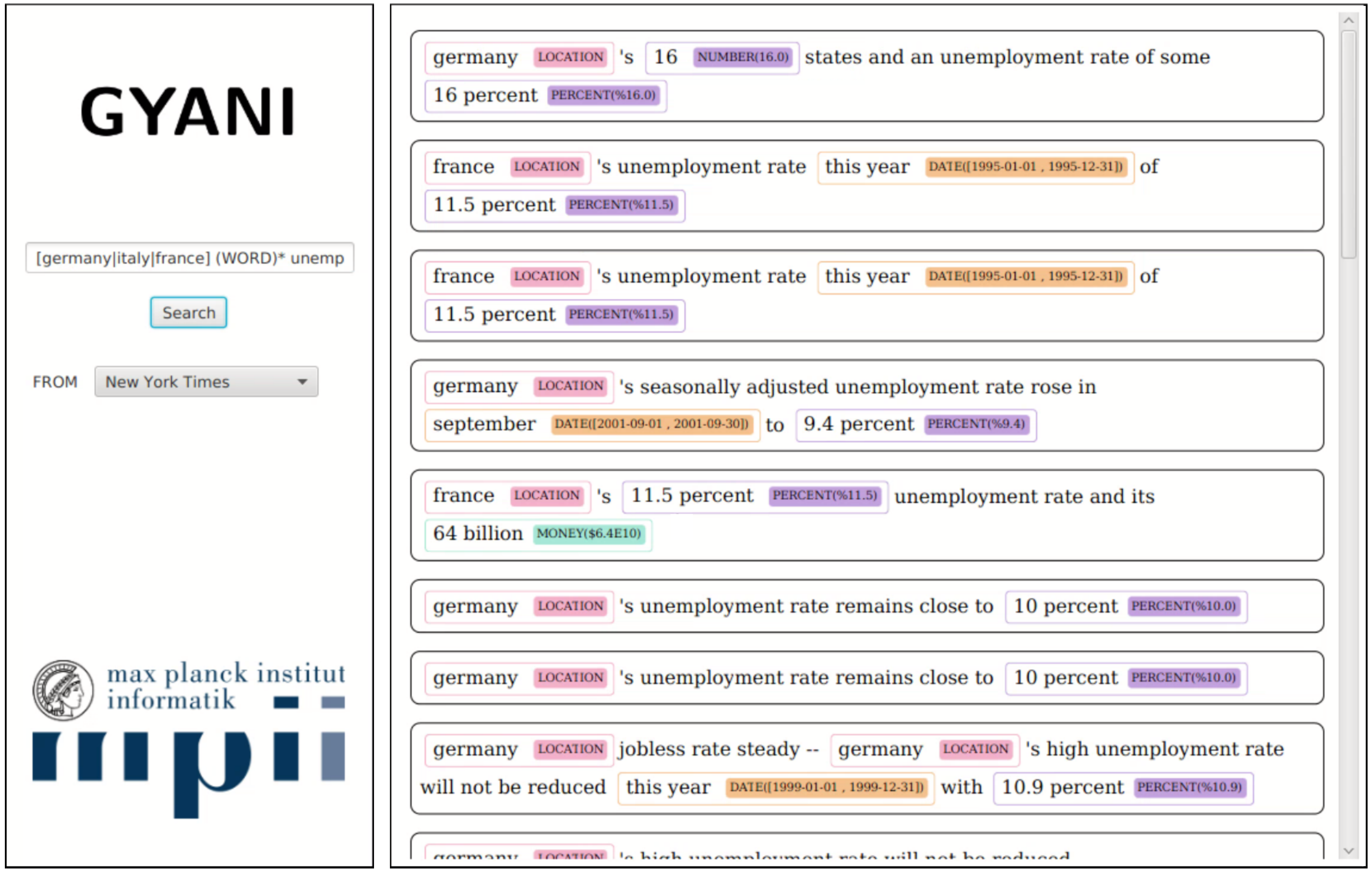
russia LOCATION joined nato ORGANIZATION 's partnership for peace MISC intended to foster military cooperation between once-hostile alliances mr. clinton PERSON and president boris yeltsin PERSON agreed to meet in washington LOCATION in september DATE([1994-09-01 , 1994-09-30])

united states LOCATION joined nato ORGANIZATION in 1949 DATE([1949-01-01 , 1949-12-31])

spain LOCATION joined nato ORGANIZATION 15 years ago DATE([1982-07-10 , 1982-07-10])

america LOCATION is doing than i have seen here since we joined nato ORGANIZATION 50 years ago DATE([1948-02-22 , 1948-02-22])

Figure 3.4: GYANI's GUI. The query, "LOCATION $\langle \mathit{joined\ nato} \rangle$ DATE", uses the projection operator $\boxed{\ell}$ to retrieve sentences as evidences.


GYANI
[germany|italy|france] (WORD)* unemp
Search
FROM
New York Times
max planck institut informatik

germany LOCATION 's 16 NUMBER(16.0) states and an unemployment rate of some 16 percent PERCENT(%16.0)

france LOCATION 's unemployment rate this year DATE([1995-01-01 , 1995-12-31]) of 11.5 percent PERCENT(%11.5)

france LOCATION 's unemployment rate this year DATE([1995-01-01 , 1995-12-31]) of 11.5 percent PERCENT(%11.5)

germany LOCATION 's seasonally adjusted unemployment rate rose in september DATE([2001-09-01 , 2001-09-30]) to 9.4 percent PERCENT(%9.4)

france LOCATION 's 11.5 percent PERCENT(%11.5) unemployment rate and its 64 billion MONEY($6.4E10)

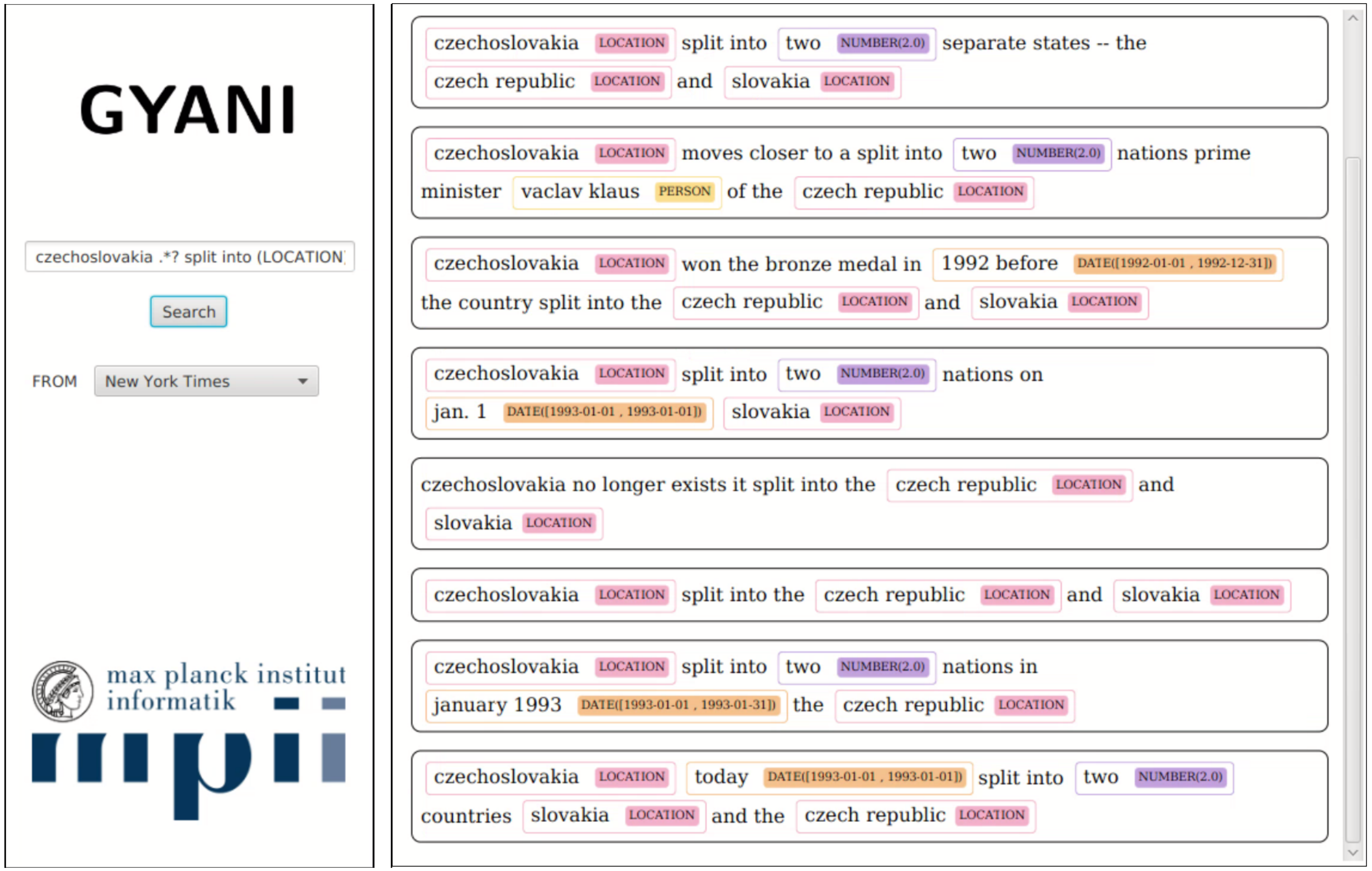
germany LOCATION 's unemployment rate remains close to 10 percent PERCENT(%10.0)

germany LOCATION 's unemployment rate remains close to 10 percent PERCENT(%10.0)

germany LOCATION jobless rate steady -- germany LOCATION 's high unemployment rate will not be reduced this year DATE([1999-01-01 , 1999-12-31]) with 10.9 percent PERCENT(%10.9)

germany LOCATION 's high unemployment rate will not be reduced

Figure 3.5: Retrieved text regions for the structured query "[*germany* | *italy* | *france*] (WORD)∗ unemployment rate (NUMBER)".



czechoslovakia .*? split into (LOCATION

czechoslovakia LOCATION split into two NUMBER(2.0) separate states -- the czech republic LOCATION and slovakia LOCATION

czechoslovakia LOCATION moves closer to a split into two NUMBER(2.0) nations prime minister vaclav klaus PERSON of the czech republic LOCATION

czechoslovakia LOCATION won the bronze medal in 1992 before DATE([1992-01-01 , 1992-12-31]) the country split into the czech republic LOCATION and slovakia LOCATION

czechoslovakia LOCATION split into two NUMBER(2.0) nations on jan. 1 DATE([1993-01-01 , 1993-01-01]) slovakia LOCATION

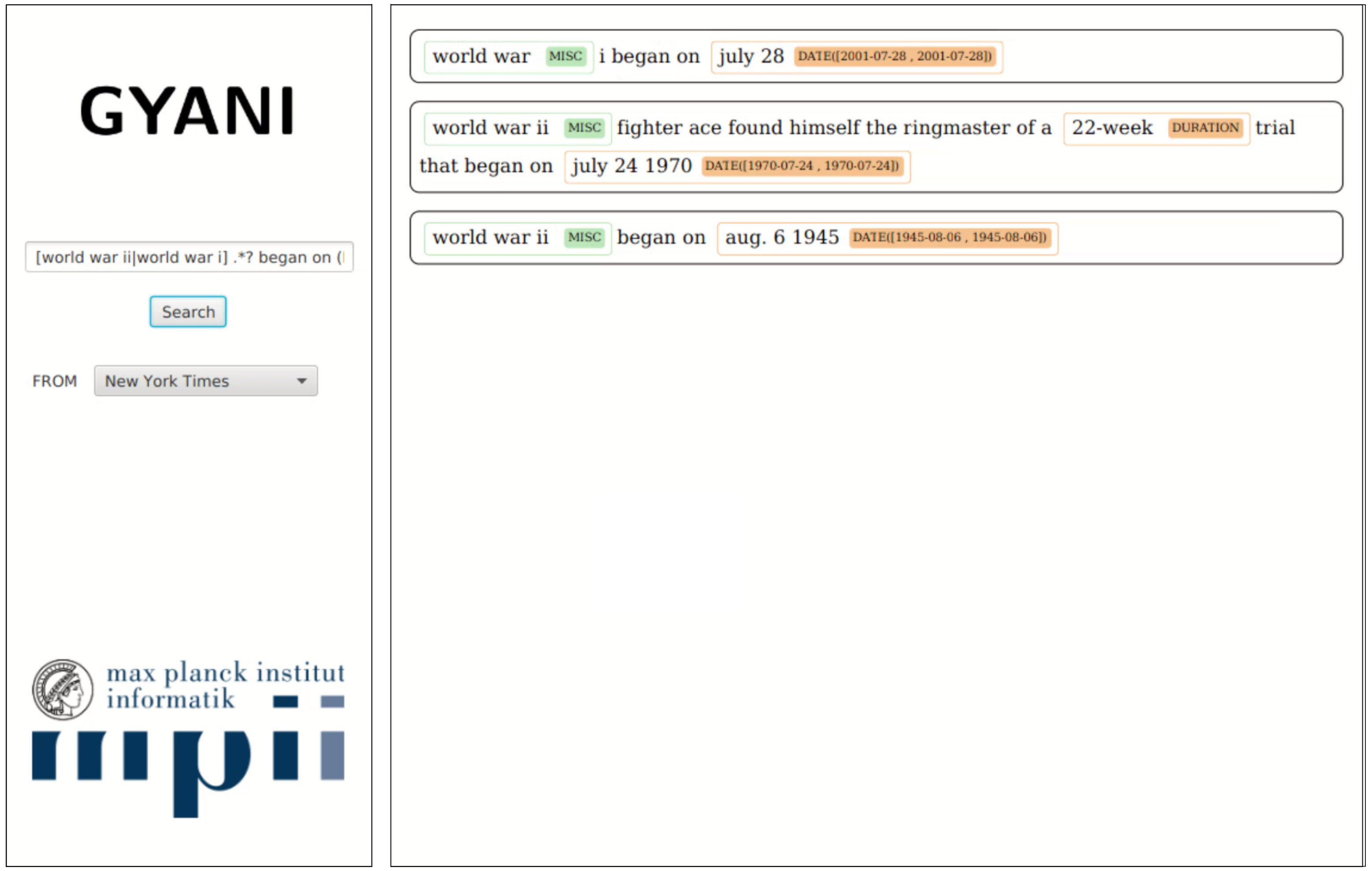
czechoslovakia no longer exists it split into the czech republic LOCATION and slovakia LOCATION

czechoslovakia LOCATION split into the czech republic LOCATION and slovakia LOCATION

czechoslovakia LOCATION split into two NUMBER(2.0) nations in january 1993 DATE([1993-01-01 , 1993-01-31]) the czech republic LOCATION

czechoslovakia LOCATION today DATE([1993-01-01 , 1993-01-01]) split into two NUMBER(2.0) countries slovakia LOCATION and the czech republic LOCATION

Figure 3.6: Retrieved text regions for the structured query "*czechoslovakia* $\boxed{.*?}$ *split into* (LOCATION)+".

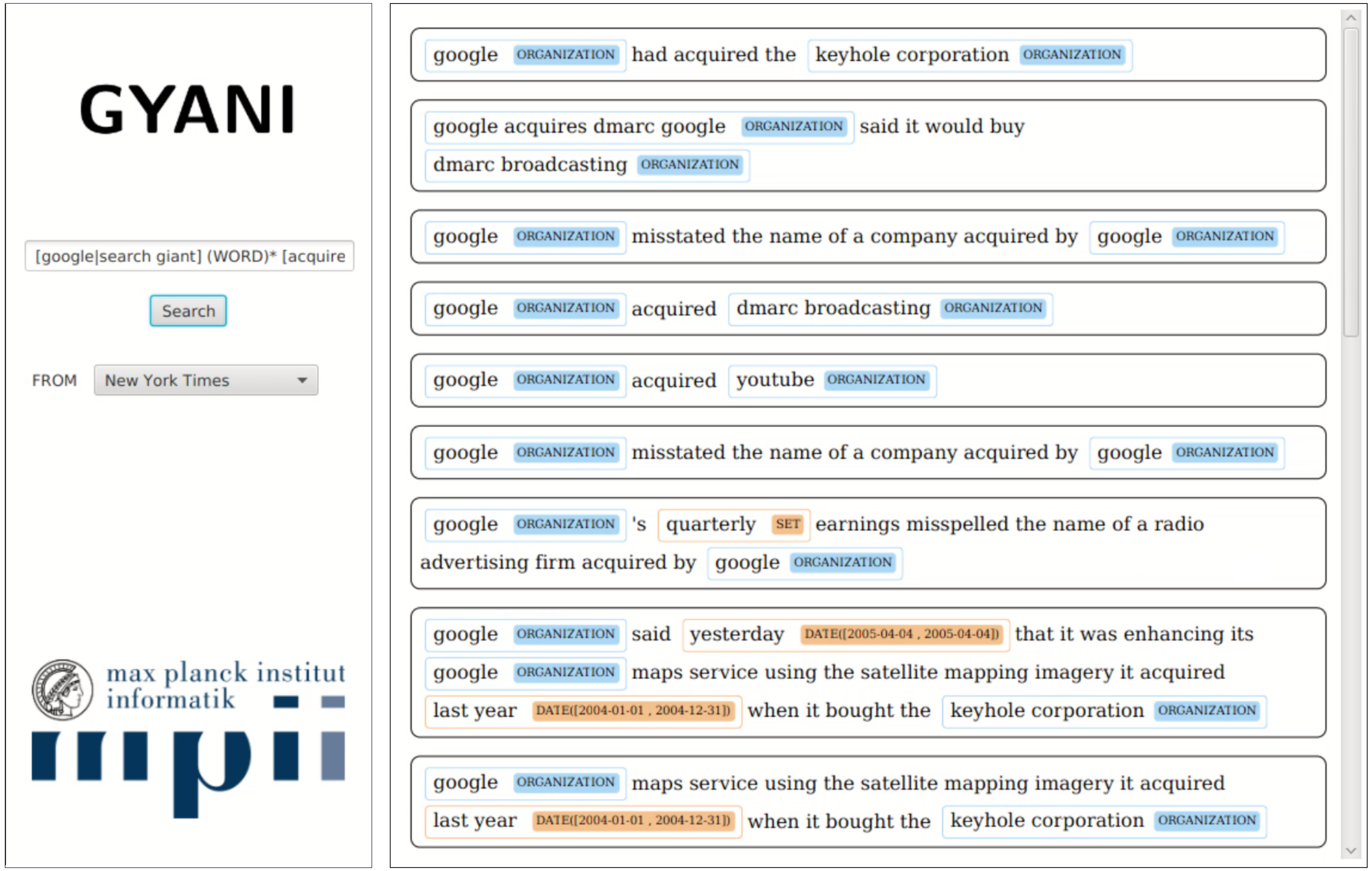

GYANI
[world war ii|world war i] .*? began on (
Search
FROM
New York Times
max planck institut informatik

world war MISC i began on july 28 DATE([2001-07-28 , 2001-07-28])

world war ii MISC fighter ace found himself the ringmaster of a 22-week DURATION trial that began on july 24 1970 DATE([1970-07-24 , 1970-07-24])

world war ii MISC began on aug. 6 1945 DATE([1945-08-06 , 1945-08-06])

Figure 3.7: Retrieved text regions for the structured query “[*world war ii* | *world war i*] $\boxed{.*?}$ *began on* (DATE)”.


GYANI
[google|search giant] (WORD)* [acquire
Search
FROM
New York Times
max planck institut informatik

google ORGANIZATION had acquired the keyhole corporation ORGANIZATION

google acquires dmarc google ORGANIZATION said it would buy
dmarc broadcasting ORGANIZATION

google ORGANIZATION misstated the name of a company acquired by google ORGANIZATION

google ORGANIZATION acquired dmarc broadcasting ORGANIZATION

google ORGANIZATION acquired youtube ORGANIZATION

google ORGANIZATION misstated the name of a company acquired by google ORGANIZATION

google ORGANIZATION 's quarterly SET earnings misspelled the name of a radio
advertising firm acquired by google ORGANIZATION

google ORGANIZATION said yesterday DATE([2005-04-04 , 2005-04-04]) that it was enhancing its
google ORGANIZATION maps service using the satellite mapping imagery it acquired
last year DATE([2004-01-01 , 2004-12-31]) when it bought the keyhole corporation ORGANIZATION

google ORGANIZATION maps service using the satellite mapping imagery it acquired
last year DATE([2004-01-01 , 2004-12-31]) when it bought the keyhole corporation ORGANIZATION

Figure 3.8: Retrieved text regions for the structured query “[*google* | *search giant*] (WORD)∗ [*acquired* | *acquires*] (ORGANIZATION)”.

CHAPTER 4

OPTIMIZING HYPER-PHRASE QUERIES

- Dhruv Gupta and Klaus Berberich. Efficient Retrieval of Knowledge Graph Fact Evidences. In ESWC 2019, pages 90–94.
- Dhruv Gupta and Klaus Berberich. Optimizing Hyper-Phrase Queries. Under Submission.

4.1 Introduction

In this chapter, we study how to reduce the time to process a special class of queries that arise in information extraction (IE) and retrieval (IR) — *hyper-phrase queries* (HPQs). A HPQ consists of a sequence of phrase sets that are needed to be matched against a document collection. A relevant match requires that it contain at least one phrase from each phrase set. Furthermore, to retrieve only semantically meaningful textual evidences, each match of the HPQ must respect the sequential ordering of the spotted phrases from each phrase set. As we show, processing a HPQ in a naïve manner, using inverted indexes over words, will be too expensive, as we can not leverage common phrases and word co-occurrences among the phrase-sets for its execution.

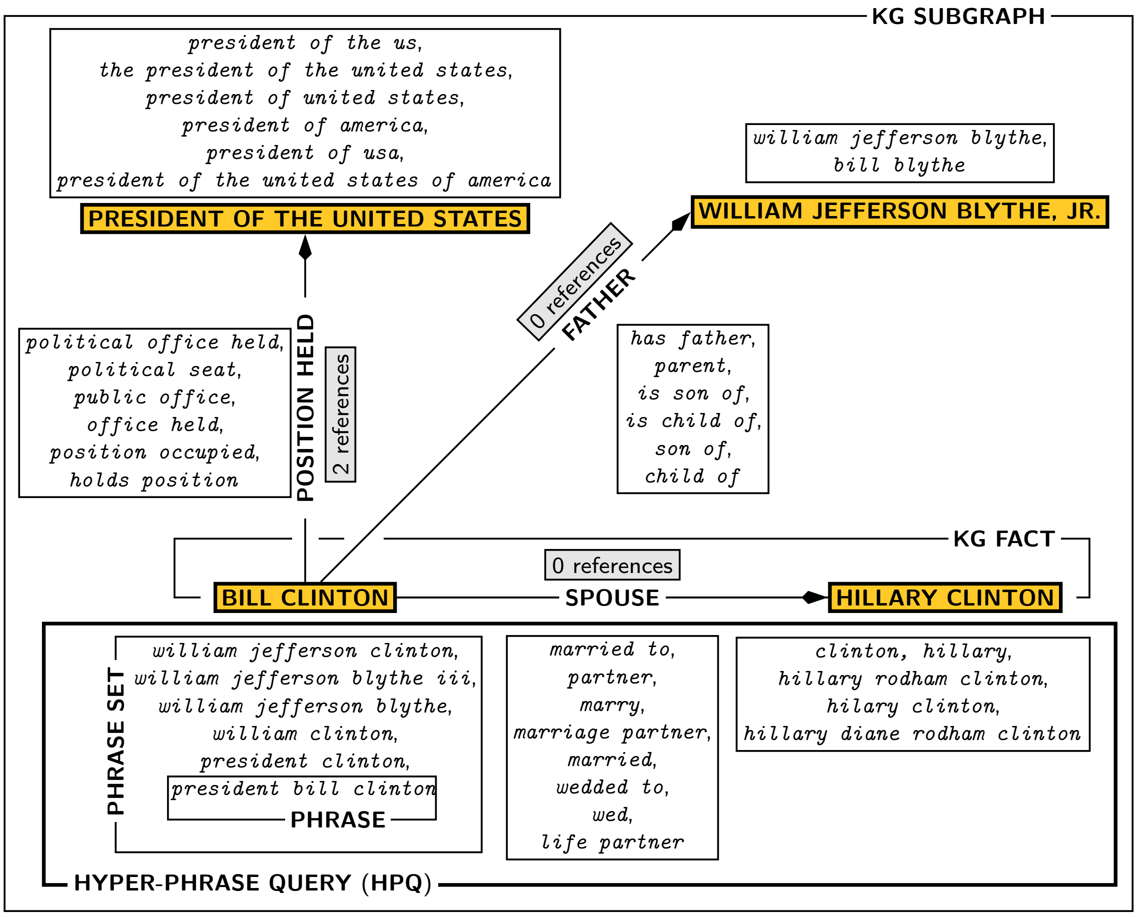


Figure 4.1: KG subgraph containing relationships of BILL CLINTON to other entities in Wikidata. A fact connects two entities via a relation in the KG. A hyper-phrase query for a KG fact corresponds to sequence of phrase sets that contain surface forms of the subject, predicate, and object. Note that there are no references establishing the provenance of two facts in Wikidata.

An important application of HPQs that we study in this chapter is that of spotting provenance for facts in knowledge graphs (KGs). Journalists and scholars in humanities are often required to verify and validate claims involving persons and organizations by identifying textual evidence in large document collections or on the Web [63]. In such scenarios, the journalists require a search system that helps them spot multiple evidences for known and emerging named entities as well as their relations. This however is not easy. Due to the inherent dynamics of natural language, named entities (e.g., persons, organizations, and locations) can be referred by many aliases. Thus, to retrieve textual evidence for an entity (e.g., BILL CLINTON), a phrase set query (e.g., {`bill clinton` , `william clinton` , `president clinton` ...}) is required.

To assist journalists and scholars in humanities, large KGs such as Wikidata have started to substantiate facts concerning named entities with references to news articles or scientific reports available on the Web. However, there still exist many facts in Wikidata that are entered manually without any references

$$\left\langle \left\{ \begin{array}{c} \textit{bill clinton} \\ \textit{william jefferson clinton} \\ \textit{william jefferson blythe iii} \\ \textit{william jefferson blythe} \\ \textit{william j. clinton} \\ \textit{clinton} \\ \textit{william jefferson "bill" clinton} \\ \textit{william clinton} \\ \textit{president clinton} \\ \textit{president bill clinton} \end{array} \right\}, \left\{ \begin{array}{c} \textit{position held} \\ \textit{political office held} \\ \textit{political seat} \\ \textit{public office} \\ \textit{office held} \\ \textit{position occupied} \\ \textit{holds position} \end{array} \right\}, \left\{ \begin{array}{c} \textit{president of the united states of america} \\ \textit{mr. president} \\ \textit{us president} \\ \textit{president of the us} \\ \textit{president of the united states} \\ \textit{the president of the united states} \\ \textit{president of united states} \\ \textit{potus} \\ \textit{president of america} \\ \textit{president of the u.s.} \\ \textit{president of usa} \end{array} \right\} \right\rangle$$

Figure 4.2: An example HPQ. It is constructed from the Wikidata KG [22] and concerns the US president BILL CLINTON.

or provenance. It has been shown that a Wikidata item has considerably less references than a Wikipedia article on average: 3.4 versus 7.5 [173]. In order to establish the origin of a KG fact (e.g., ⟨BILL CLINTON, SPOUSE, HILLARY CLINTON⟩), we need to retrieve textual evidences using multiple phrase sets (e.g., ⟨{*bill clinton* , *william clinton* , ...}, {*married* , *partner* , ...}, {*hillary clinton* , *hillary rodham clinton* , ...}⟩). Naïvely scanning large document collections for all surface forms underlying the subject, predicate, and object can be extremely time consuming to retrieve their evidences.

To validate KG facts, current approaches either rely on KG refinement techniques [81, 171] or spotting their evidences in external document collections [84, 85, 141]. To retrieve evidences for a KG fact, current approaches utilize commercial search engine APIs that only offer Boolean and phrase query operators. Using the limited expressibility of such operators, a user has to tediously issue multiple phrase queries for each alias underlying the KG fact arguments. Moreover, results returned by such systems may not be semantically relevant as the aliases of the fact arguments may not be ordered or occur far apart from each other in the document.

On the Web, hyper-phrase queries arise when answering phrase queries that can be formulated in many ways (e.g., regarding entities [183, 205]). For instance, the phrase query *icc wc 1990s* can be expanded using dictionaries to sequence of multi-phrase sets ⟨{*icc*, *international cricket council*}, {*wc*, *world cup*}, {*1992*, *1996*, *1999*}⟩. To increase recall, it is necessary that we can at scale retrieve sentences that contain their mentions to answer users' queries. Hyper-phrase queries also arise in the field of *Phramacovigilance* [140, 182], where adverse drug reactions to medicines need to be retrieved from large amounts of text present in scientific literature, medical records, and social media posts on the Web.

To the best of our knowledge there exists no prior work on optimizing HPQs in large-scale IR systems. Prior studies in the string processing community [32, 45, 67, 82, 159] have studied the problem of *variable-length phrase matching* however using only small in-memory indexes to support the matching algorithms. The key challenges that we overcome for optimizing HPQs are as follows. First, we need to come up with a data model for phrases and word co-occurrences along with syntactical information such that we can represent the combinatorial space for optimizing a HPQ. Second, we need to construct query operators that can retrieve text regions corresponding to a HPQ. Third and finally, we need to design methods that order the query operators in an efficient query plan to be executed over our data model.

Outline for the remainder of the chapter is as follows. In Section 4.2, we discuss prior art with respect to our problem setting. In Section 4.3, we formally describe a HPQ and what its corresponding match constitutes. In Section 4.4, we discuss the data model that assists in retrieving text regions for a given HPQ. In Section 4.5, we derive the indexing units used to construct our dictionaries and indexes. In Section 4.6, we discuss the design of operators needed to execute a HPQ. In Section 4.7, we describe the algorithms to identify the optimal sequence of query operators to process a HPQ. In Section 4.8, we discuss an in-depth evaluation of our proposed methods and discuss the results thus obtained. Finally, we present the concluding remarks of our research in Section 4.9.

4.2 Related Work

We now discuss prior studies related to variable length pattern matching in text documents, spotting KG facts using indexes over annotated document collections, and query optimization in large-scale IR systems.

Variable Length Pattern Matching is an allied area with respect to our problem setting. Prior works [32, 45, 67, 82, 159] have studied how in-memory data structures can help in the design of efficient matching algorithms. For instance, [159] considered "matching-lookup table" while [32] considered a "wavelet tree" as an in-memory index to to speed up the matching process. Cho and Rajagopalan [60] and Li et al. [145] presented algorithms to execute character-level regular expressions to match text regions for many information extraction tasks. Our work in contrast, leverages large-scale inverted indexes that are part of modern IR systems to efficiently execute a more difficult problem.

Indexing Annotated Text. A straight-forward approach to spotting evidences for KG facts is to index document collections annotated with named entities linked to KGs. However, using such an approach we can not spot facts for out-of-KG entities or their emerging relations. An early work for indexing annotated text was [157]. In that work, text regions were represented in a two dimensional Cartesian co-ordinate plane. With this data model, an efficient index was created using R*-trees and region overlap operations were specified. Cafarella and Etzioni [51] investigated how to speed up IE tasks by indexing annotations surrounding words in an inverted index. Bast and Buchhold [36] studied how to jointly index KG entities that are spotted in text documents and the KG relations for semantic search. More recently, there has been work on spotting KG facts using regular expression based operators at word-level [98, 103]. However, all these approaches disregard any optimization for efficient execution of hyper-phrase queries.

Fact Spotting. Elbassuoni et al. [75] and Metzger et al. [154] proposed a system that retrieved witness documents given a KG fact as a query. However, a limitation of the system was that documents need to be processed apriori and linked to KG facts for their retrieval. Put another way, out-of-KG facts or entities can not be processed with their system. Tylenda et al. [199] attempted to overcome the above limitation by spotting KG facts using surface forms of named entities and paraphrase dictionaries. However, just like the string-matching approaches their approaches are not scalable to large document collections. This is because they make many simplifying assumptions (e.g., knowing before hand the relevance of a document for a KG fact) and heuristics (e.g., specified character distance between arguments of KG fact) for their methods. Our approach solves these issues by relying on a data model that can represent n-grams, skip-grams, and sentence boundaries. Relying on our data model, we can then retrieve text regions as evidences for KG facts.

Query Optimization. Ipeirotis et al. [122] investigated how to model query execution plans with respect to recall of relevant documents and the query's execution time. Their approach contrasted between two models: inverted index based approach versus scanning the entire document collection. Agrawal et al. [27] described an algorithm that identifies relevant sets of document for named entities by finding a "token-set-cover" for various surface forms of the named entity and computing a join of the retrieved documents. Williams et al. [202] and Panev and Berberich [170] described approaches to query phrases using combinations of inverted, phrase, nextword, and direct indexes. Our work in contrast explores ways to compute an optimal plan of hyper-phrase query execution using dictionaries and indexes over n-grams and skip-grams.

4.3 Problem Definition

We now define the basic elements underlying the problem of executing hyper-phrase queries. Key concepts related to the definitions are illustrated in Figure 4.1. Consider a large document collection, $\mathcal{D} = \{d_1, \ldots, d_{|D|}\}$. Each document in the collection $d \in \mathcal{D}$ is a sequence of sentences $d = \langle s_1, s_2, \ldots, s_{|d|} \rangle$. Each sentence further consists of a sequence of words drawn from the vocabulary Σ associated with the collection.

Phrase Query. A basic *phrase query* is a sequence of words that must be matched contiguously in a document. For example, consider the phrase query: $\langle$*bill clinton is son of bill blythe*$\rangle$. The entire phrase with each word occurring in that sequence must be matched in the retrieved document. Formally, a phrase query p can be defined as:

$$p \in \Sigma^+. \tag{4.1}$$

A document is a match for a phrase query p if it contains a sentence in which the words of the phrase occur contiguously. Formally, a sentence s matches the phrase p if

$$p \sqsubset s \;\equiv\; \exists 1 \leq i \leq |s| : \forall 1 \leq l \leq |p| : s[i+l-1] = p[l], \tag{4.2}$$

and a document d matches a phrase if

$$p \sqsubset d \;\equiv\; \exists s \in d : p \sqsubset s. \tag{4.3}$$

Phrase-Set Query. A *phrase-set* query is one that combines multiple phrases, for example, consisting of different paraphrases to increase recall in document retrieval. Consider, as a concrete example, the phrase set query

$\{\langle\texttt{alumni of}\rangle\langle\texttt{attended college}\rangle\}$. Formally, a phrase set query P can be defined as a subset of all possible phrases that can be generated from the vocabulary:

$$P = \{p_1, p_2, \ldots, p_{|P|}\} \subseteq \Sigma^{+}. \tag{4.4}$$

A document is considered a match for a phrase-set query if at least one of the phrases in the set is found in the document according to Equation 4.3. Concretely, this can be put as follows:

$$P \sqsubset d \equiv \exists p \in P : p \sqsubset d. \tag{4.5}$$

Hyper-Phrase Query (HPQ) is defined to be a sequence of phrase sets. An example of a HPQ corresponding to a KG fact is shown in Figure 4.2. Formally, a HPQ is a sequence of phrase sets:

$$\mathcal{P} = \langle P_1, P_2, \ldots, P_{|\mathcal{P}|}\rangle. \tag{4.6}$$

A document is said to match a HPQ if one phrase from each phrase set is matched, and matches for adjacent phrase sets occur within k sentences from each other in the document. Formally:

$$\begin{aligned} \mathcal{P} \sqsubset d \;\; \equiv \;\; & \exists 1 \leq i_1 \leq \ldots \leq i_{|\mathcal{P}|} : \\ & \forall 1 \leq j \leq |\mathcal{P}| : \exists p \in P_j : p \sqsubset s_{i_j} \wedge \\ & \forall 1 < j \leq |\mathcal{P}| : (s_{i_{j+1}} - s_{i_j}) \leq k \end{aligned} \tag{4.7}$$

The definition ensures that phrase matches across different sentences have to occur in the order specified by the HPQ. Should more than one phrase set be matched by the same sentence, we additionally ensure that their order is respected within the sentence. To reduce formalism, we omit this detail from the above definition.

Establishing Provenance for KG Facts

Consider the problem of spotting knowledge graph (KG) facts on the Web or in large document collections [154] as a concrete use case of matching hyper-phrase queries. A KG consists of facts. Each fact consists of vertices that are either named entities or literals and edges that define relationships between them. The facts are usually encoded in the form of a triple, which consists of two vertices (either named entities or literals) and an edge (predicate or relationship). The triples can be succinctly represented as ⟨(S)UBJECT, (P)REDICATE, (O)BJECT⟩. Each component of the triple is a canonical representation of its various surface forms. Let these surface forms of S, P, and O be denoted by: $\{\mathsf{s}_1, \mathsf{s}_2, \ldots, \mathsf{s}_u\}, \{\mathsf{p}_1, \mathsf{p}_2, \ldots, \mathsf{p}_v\}$, and $\{\mathsf{o}_1, \mathsf{o}_2, \ldots, \mathsf{o}_w\}$ respectively. A **text region** is considered an **evidence** (thereby establishing provenance for the fact) if it contains at least one phrase from each of the phrase sets within a distance of k sentences and with the particular order as expressed in the fact.

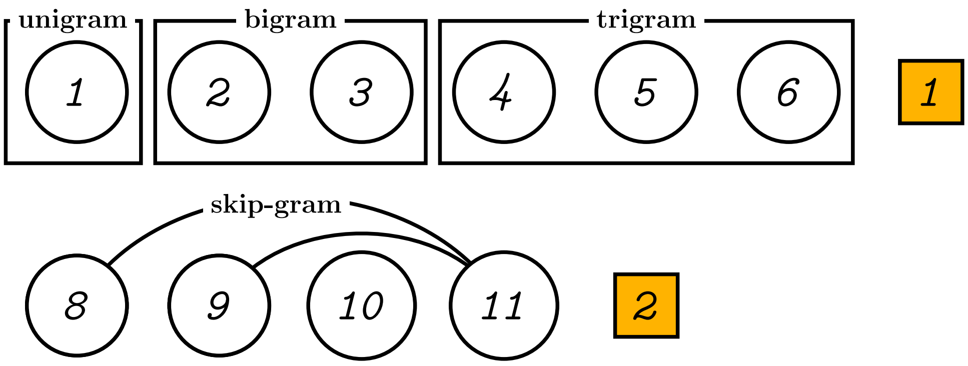


Figure 4.3: Indexing units can be obtained by considering ordered contiguous sequences of words (n-grams) or ordered non-contiguous combinations of words (skip-grams). Additionally, to maintain context windows and semantically meaningful text regions we keep track of sentence boundaries. Circular nodes represent words and the numbers their positions. While the square nodes represent sentence boundaries and the numbers represent sentence identifiers.

Problem Complexity

Consider the problem of retrieving documents for a HPQ $\mathcal{P} = \langle P_1, P_2, \ldots, P_{|\mathcal{P}|} \rangle$, where each phrase set P can contain at most m surface forms, over a document collection $\mathcal{D}$. Consider that we have access to a standard dictionary and inverted index over words in the document collection. From them we can assemble the text regions for the evidences by looking up the word and its offsets within the documents. A naïve approach is: first retrieve those documents that contain the words from each of the m phrases in $|\mathcal{P}|$ phrase sets. As a second step, we can intersect and pool those documents in which at least one phrase from each phrase set is present. Finally, with this pool of documents we can then scan each document for potential matches using the string matching algorithms from [32, 45, 67, 82, 159]. However, this simple approach is inefficient as we do not leverage common sub-phrases among the phrases in each phrase set for retrieval of posting lists. Furthermore, we also do not leverage any co-occurrence of words among phrase sets that can significantly bring down the cost of processing a HPQ.

4.4 Data Model

There are three key challenges that need to be overcome in order to reduce the cost of processing a HPQ. The first challenge is to capture phrases as simpler combinations of n-grams and skip-grams. Capturing skip-grams is hard as their number grows polynomially with the sentence length. The second challenge is coming up with novel ways of maintaining sentence boundaries such that we can quickly identify the match of two phrases to lie within a distance of k sentences. Third and finally, the data model must allow us to compute different ways to represent combinations of phrases. We can then design algorithms to optimize the order of processing such operators. Figure 4.3 summarizes the key elements of our data model. First, we must provide a data model that is capable of representing text regions within documents.

Modeling Text Regions. A *phrase* in our data model is defined as a contiguous sequence of words with their positional span as:

$$\mathbb{N} \times \mathbb{N} \times \Sigma^+. \tag{4.8}$$

In Equation 4.8, the Cartesian product of natural numbers $\mathbb{N} \times \mathbb{N}$ represents the *text region* $[i, j]$ of the phrase $\langle w_i, \ldots, w_j \rangle \in \Sigma^+$. First, by restricting the size of the spans to a constant n we can model n-grams. Thus, it allows us to represent arbitrary-length phrases as combinations of fixed-size n-grams. Second, we can also capture skip-grams of bounded gap size with the same representation. Skip-grams are important as they capture the co-occurrence statistics needed to maintain long-range sequence information for ordering phrases.

Incorporating Sentence Boundaries. A text document is explicitly structured using syntactical structures such as sentences, paragraphs, pages, sections, chapters etc. In our work, we consider *sentences* as the maximal unit for imposing structure on text. Sentence boundaries can be reliably detected using natural language processing (NLP) tools (e.g., Stanford's CoreNLP toolkit [151]).

We now describe design choices on encoding the sentence structure in our data model. The first design choice corresponds to encoding the *sentence spans* in the model for the text regions (Equation 4.8) as follows:

$$\overbrace{\mathbb{N} \times \mathbb{N}}^{\text{sentence span}} \times \overbrace{\mathbb{N} \times \mathbb{N} \times \Sigma^+}^{\text{phrase}}. \tag{4.9}$$

In Equation 4.9, the first Cartesian product of natural numbers records the sentence boundaries as positions of the outermost words that mark-off the sentence in which the phrase spanned by the positions $[i, j]$ is contained. For example, to represent the bigram in Figure 4.3, we can write the four-tuple as: $(1, 6, 2, 3)$. The are two disadvantages with this design choice. First, we double the size of the positions needed to be recorded in the indexes thereby increasing the sizes of our indexes. Second, we can not compute distance in terms of number of sentences when relaxing the match of the HPQ to lie within a distance of k sentences.

The second way to incorporate sentence boundaries is to consider punctuation (e.g., period) as an element of the vocabulary. With sentence boundaries available as words we can then model their association with other words as skip-grams. Thus, Equation 4.8 need not be modified further. The disadvantage with this design choice is that we require another fetch request from the indexes whenever matching the HPQ to lie within a distance of k sentences.

The third and final way to record sentence information is to only encode the sentence numbers as identifiers along with phrases. The data model with this augmentation can be represented as:

$$\overbrace{\mathbb{N}}^{\text{sentence id}} \times \overbrace{\mathbb{N} \times \mathbb{N} \times \Sigma^{+}}^{\text{phrase}} . \tag{4.10}$$

For instance, with this approach, we can represent the bigram and the sentence information in Figure 4.3 as: $(1, 2, 3)$. With this design choice, we need to keep track of only an additional identifier, as opposed to two positions in the first design choice, thereby reducing the storage cost. Furthermore, with this representation, we can easily compute relaxations of phrase matches to lie within a distance of k sentences. Our implementation utilizes this model of sentence identifiers.

4.5 Indexing Documents

Documents can be indexed by considering contiguous and non-contiguous combinations of words in our data model as indexing units. The key indexing units we consider are n-grams and skip-grams (shown in Figure 4.3).

4.5.1 Indexing Units

n-grams. By considering the contiguous sequence of words of fixed length we can arrive at unigrams, bigrams, and trigrams as indexing units. These n-grams can immediately help us spot phrases by decomposing them as an overlap of two or more n-grams. For example, to retrieve documents for the phrase "*physics nobel prize*", we can decompose it to be a overlap of bigrams "*physics nobel*" and "*nobel prize*".

skip-grams. To spot a phrase or sequence of phrases it shall be helpful if in advance we know whether pair of words co-occurs or not. Examples of skip-grams are also shown in Figure 4.3. However, recording all skip-grams contained within a sentence can lead to index blowup. To curtail the selection of skip-grams to only those which are highly discriminative, we note three choices for their generation. First, we can leverage the concept of *point-wise mutual information* from information theory. To record a skip-gram $\langle a, b \rangle$, we first measure their document frequencies:

$$Pr(a) = \frac{\text{df}(a)}{|D|}, \tag{4.11}$$

$$Pr(b) = \frac{\text{df}(b)}{|D|}, \tag{4.12}$$

$$Pr(\langle a, b \rangle) = \frac{\text{df}(\langle a, b \rangle)}{|D|} \tag{4.13}$$

where, df(a), df(b), and df($\langle a, b\rangle$) denote document frequency of a, b, and $\langle a, b\rangle$ while $|D|$ denotes the collection size. The mutual information is then computed by:

$$MI(\langle a, b\rangle) = \frac{Pr(\langle a, b\rangle)}{Pr(a) \cdot Pr(b)}. \tag{4.14}$$

By selecting a suitable threshold for discriminative skip-grams we can reduce the number of indexing units kept in the index. With this choice for skip-gram generation, we can lower the size of skip-gram indexes, as we only keep those skip-grams that are significant. However, to spot out-of-KG entities, we may require combinations of words that are infrequent (e.g., 50 CENT). In other words, this scheme of skip-gram generation is highly restrictive for spotting mentions of entities.

Second, we can leverage the presence of noun phrases, verbal phrases, and more complex relations between part-of-speech tags obtained via dependency parsing. Such choices can be justified by restricting ourselves to the application at hand (e.g., relation extraction). We can therefore record skip-grams that are ordered pair of words between noun phrases or between two part-of-speech tags.

Third and finally, we can leverage the syntactical structure of sentences within documents to record skip-grams. With this scheme we keep track of only those ordered co-occurrences of words that are within a sentence and within fixed separation of ℓ words. To instantiate our indexes we consider skip-grams where the word separation is at most ten words (i.e., $\ell = 10$). By limiting the separation between the skip-grams we also discard those combinations of words that may be part of different compound sentences and not semantically related to each other. Since, we keep track of infrequent skip-grams also, we can detect mentions of emerging entities as well. This however leads to large inverted index sizes. Since, establishing provenance for KG facts is a recall-oriented task, we consider this scheme for recording the ordered co-occurrence of words.

4.5.2 Dictionaries and Indexes

Distributed Indexing Infrastructure. Our dictionaries and indexes are stored in HBase, a modern state-of-the-art distributed extensible record store. HBase is an open-source variant of the BigTable [59] distributed extensible record store that forms the backbone of many Web-scale commercial services. Distributed record stores provide us the advantage of quickly creating our infrastructure as index build times scale linearly with the number of machines in the cluster. Furthermore, distributed record stores are fault tolerant and allow us to maintain multiple copies of indexes. Our indexes in the HBase record store comprise of tables, where the records contain key-value pairs. In each key-value pair, the key of a record in the table encodes the indexing unit while the value stores the compressed payload for the posting list.

Dictionaries. For each n-gram and skip-gram we record their collection statistics in a dictionary. Each entry in the dictionary stores for each indexing

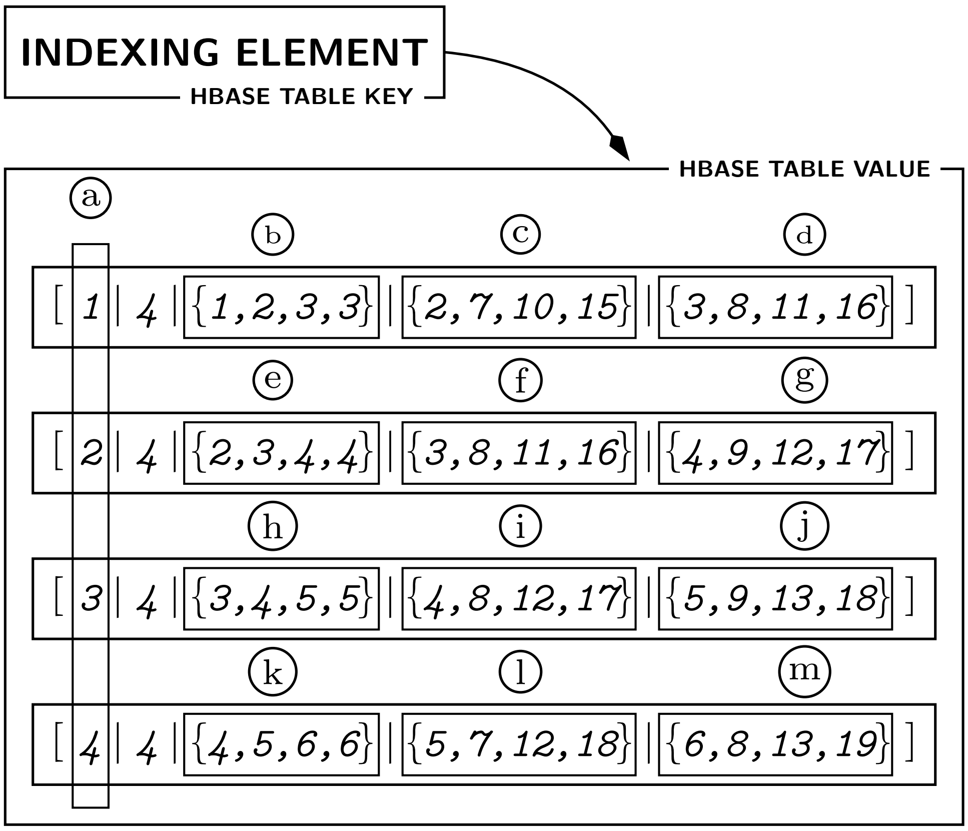


Figure 4.4: Inverted index structure. Block marked as (a) denotes the document identifiers which are used for ordering the payloads in the posting lists and are compressed as a separate blocks. Next each positional payload is stored by recording the number of occurrences of the indexing element in the document followed by compressed payloads of sentence identifiers (i.e., b, e, h, & k), corresponding compressed begin (i.e., c, f, i, & l), and end block (i.e., d, g, j, & m) payloads.

unit p: the document frequency df(p) – number of documents containing p and the collection frequency cf(p) – number of times p was observed in the entire collection.

Inverted Indexes. The inverted index structure we implement is shown in Figure 4.4. Each posting consists of compressed lists corresponding to sorted document identifiers, and sorted lists for sentence identifiers, begin and end positions of the indexing elements. For the n-gram indexes we can omit payload corresponding to the end positions as they can be inferred by adding the n-gram length to the begin positions. The compression technique utilized is the patched frame of reference [11, 216].

4.6 Query Processing and Operators

We now discuss the design of operators over text regions that we use to represent the combinatorial space of a hyper-phrase query.

Algorithm 5: Computing a variable-length gap match.

```
Input  : Posting lists $L_l$ and $L_r$ corresponding to the left and right
         operands of the variable length match operator and $k$ indicating
         the number of sentences the match may span.
Output : Posting List containing the resultant text regions containing the
         variable length match.
Function match($L_l$, $L_r$, $k$)
    $R \leftarrow$ find all common documents for postings in $L_l$ & $L_r$
    $L \leftarrow \varnothing$, $S \leftarrow \varnothing$
    foreach doc-id $\in R$ do
        $S \leftarrow$ join(payload for doc-id in $L_l$, payload for doc-id in $L_r$, $k$)
        $L \leftarrow L$.append(new⟨doc-id, $S$⟩)
    return $L$
Function join($S_l$, $S_r$, $k$)
    $S \leftarrow \varnothing$
    if the last position span in $S_r$ lies before the first position span in $S_l$ then
        return $S$
    if first position span in $S_r$ is before the first position span in $S_l$ then
        $S_r \leftarrow$ remove position spans from the front of $S_r$ until the first
          position spans in it is after the first position spans in $S_l$
    for ($i \leftarrow 1; i \leq |S_l|; i++$) do
        for ($j \leftarrow 1; j \leq |S_r|; j++$) do
            if |sentence id for $S_r -$ sentence id for $S_l| \leq k$ then
                if ($S_l[i]$ is before $S_r[j]$) then
                    $S \leftarrow S$.append([$S_l[i]$.begin, $S_r[j]$.end])
    return $S$
```

4.6.1 Basic Query Processing

First and foremost, we discuss how to process a HPQ to obtain the resulting text regions. Assume that for a HPQ $\mathcal{P}$ we have obtained the posting lists corresponding to each of its constituent phrase sets P. To find the resulting text regions that contain the evidences for the HPQ we apply the binary variable-length match operator by processing two phrase sets at a time, from left to right, in $\mathcal{P}$. Algorithm 5 shows how to process a variable-length match between two posting lists. With Algorithm 5 acting as a general framework for query processing, we note two avenues for optimization. First, we must minimize the time spent for retrieving the postings corresponding to each phrase set P in HPQ $\mathcal{P}$. Which in turn implies presenting the variable-length match operator with a minimum number of documents to process (line 4 in Algorithm 5). Second, we must minimize the time spent for joining the positions corresponding to each common document (lines 8-19 of Algorithm 5).

Naïve Optimization. A naïve strategy to optimize the execution of the HPQ is to identify a common pool of documents for *all* the phrase sets P

in HPQ $\mathcal{P}$ before processing them for a variable-length match. Clearly, this strategy is expensive as explained in Section 4.3. An improvement for matching phrase sets was proposed by Agarwal et al. [27] for named entity extraction from text documents. Their method relies on first computing a set cover over the surface forms of named entities and then utilize Boolean operators over a word index (i.e., no positional information is used) to obtain a final superset of potentially matching documents. However, there is much room for improvement on executing a HPQ by exploiting the order of phrases and optimizing them in our proposed data model.

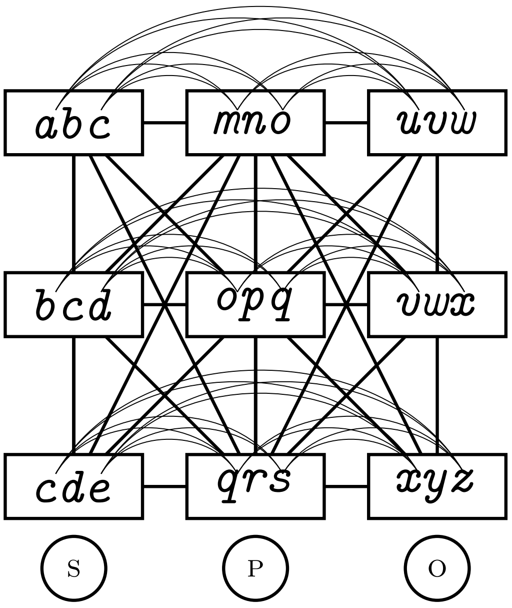


Figure 4.5: Types of operators for matching a hyper-phrase query. A vertical cover partitions a phrase set in the query by matching common n-grams or skip-grams. A horizontal order models co-occurrence of words across phrase sets in the query using skip-grams.

4.6.2 Query Operators

Vertical Cover Operator (◫). Given our data model, we can leverage n-grams and skip-grams to reduce the time spent for retrieving postings for each phrase. We now describe a *vertical cover* operator that does this. Let P be the phrase set and $\{p_1, p_2, \ldots, p_n\}$ the constituent phrases. We can induce a partitioning for each phrase set using n-grams and skip-grams. The partition that has minimal cost (see Section 4.7.1) is then used for retrieval of the posting lists.

Using n-grams, we can decompose each phrase in the phrase set using unigrams, bigrams, and trigrams. For instance, for the phrase-set $\{abc, bcd\}$ the common unigrams are $\{b, c\}$, while the common bigrams is $\{bc\}$. Using the common n-grams we can induce partitions over the phrase-set. For example, the partition induced using unigrams is: $\{a, b, c, d\}$; using bigrams the partition is: $\{ab, bc, cd\}$; and using trigrams it is: $\{abc, bcd\}$.

Using skip-grams, we can induce the partitions over phrase sets by computing ordered co-occurrences with respect to an anchor word. To compute skip-grams, we fix the first word of the phrase as an anchor word and then derive all the skip-grams with respect to it. For instance, for the phrase set $\{abc, adc\}$ the skip-grams induce the partitioning: $\{\langle a, b\rangle, \langle a, c\rangle, \langle a, d\rangle\}$. Algorithm 6 shows how to compute skip-grams that can then be used to retrieve posting lists.

Algorithm 6: Vertical cover operator using skip-grams.

```
   Input  : HPQ $\mathcal{P} = \langle P_1, P_2, \ldots, P_n \rangle$.
   Output : Set of skip-grams that cover the phrase sets $P$ in HPQ $\mathcal{P}$.
1  Function cover($\mathcal{P} \leftarrow \langle P_1, P_2, \ldots, P_n \rangle$)
2      $S \leftarrow \varnothing$                          // Resulting skip-gram cover.
       /* Compute the set of skip-grams needed to retrieve postings
          for $\mathcal{P}$ by keeping all the phrases across phrase sets in one
          set.                                                          */
3      skipGrams[] $\leftarrow$ generateSkipGrams(put all phrases of $\mathcal{P}$ in an array)
4      return skipGrams
5  Function generateSkipGrams($p[] \leftarrow \{p_1, p_2, \ldots, p_m\}$)
6      $S \leftarrow \varnothing$                          // Resulting set of skip-grams.
7      words[] $\leftarrow \varnothing$ // Holds the words for the phrase being processed.
8      foreach $p \in p[]$ do
9          words $\leftarrow p$.split()
           /* If the phrase is one-word only, keep it as is.  It will
              be retrieved using the unigram index.                    */
10         if (words.length $= 1$) then
11             $S$.put($p[i]$)
12             continue
13         i $\leftarrow 1$
14         for (j $\leftarrow$ i $+ 1$; j $\leq$ words.length; j $++$) do
15             $S$.put($\langle$words[i], words[j]$\rangle$)
16     return $S$
```

Horizontal Order Operator ($\boxminus$). We now discuss how to minimize the time for performing the joins (lines 8-19 of Algorithm 5). When computing the variable-length match for a HPQ we shall like to process those combinations that are highly selective, i.e., possess the least number of positions first. In our data model, we can compute the cost of these join by using skip-grams between words from phrases belonging to different phrase sets. Concretely, we select that skip-gram for phrases belonging to different phrase sets that contains the least number of positions. By summing up all the cardinalities indicated by the set of skip-grams, output by Algorithm 7, we can determine the cost of joining two phrase sets. Using the vertical cover and horizontal

Algorithm 7: Horizontal order operator using skip-grams.

```
Input  : Sets of n-grams that cover the phrase sets corresponding to the
         left and right operand of the variable-length phrase match.
Output : Set of selective skip-grams between phrase sets that are indicative
         of the number of positions needed to be merged for the join.
Function order($S_l, S_r$)
    $S \leftarrow \varnothing$                    // Resulting skip-gram join cover.
    words$_l$, words$_r$ $\leftarrow \varnothing$
    minCostSkipGram, skipGram $\leftarrow \varnothing$
    double minCost, cost $\leftarrow -\infty$
    foreach ($p_l \in S_l$) do
        words$_l$ $\leftarrow p_l$.split()
        minCost, cost $\leftarrow 0$
        foreach ($p_r \in S_r$) do
            words$_r$ $\leftarrow p_l$.split()
            for (i $\leftarrow$ 1; i $\leq$ words$_l$.length; i++) do
                for (j $\leftarrow$ 1; j $\leq$ words$_r$.length; j++) do
                    skipGram $\leftarrow \langle$words$_l$[i], words$_r$[j]$\rangle$
                    cost $\leftarrow$ cost(skipGram)
                    if (minCost $\equiv -\infty$) $\vee$ (minCost $>$ cost) then
                        minCost $\leftarrow$ cost
                        minCostSkipGram $\leftarrow$ skipGram
            /* Add the min.  cost skip gram.                        */
            $S$.add(minCostSkipGram)
    return $S$
```

order operators we can now model the execution of hyper-phrase queries using the n-gram and skip-gram indexes. Referring to Figure 4.5, we can represent the query execution as follows:

$$(ab \boxbar bc \boxbar cd \boxbar de) \boxminus (mno \boxbar opq \boxbar qrs) \boxminus (uv \boxbar vw \boxbar wx \boxbar xy \boxbar yz).$$

Operator Properties. We note the following operator properties that are helpful in optimizing their order. The vertical cover operator is both associative ($(a \boxbar b) \boxbar c = a \boxbar (b \boxbar c)$) and commutative ($a \boxbar b = b \boxbar a$). However, the horizontal order operator is only associative ($(a \boxminus b) \boxminus c = a \boxminus (b \boxminus c)$). Furthermore, the vertical cover operator is only left distributive over the horizontal order operator ($(a \boxbar b) \boxminus (a \boxbar c) = a \boxbar (b \boxminus c)$).

4.7 Query Optimization

We now discuss the strategy to select the optimal operator sequence to process a hyper-phrase query.

4.7.1 Cost Model

Our cost model depends on two factors. First, we account for the number of postings to be retrieved from the inverted indexes for a given indexing unit based on document frequency – C_{join}. Second, we account for the number of positions in each posting list based on collection frequency – C_{merge}. The first cost C_{join} gives us an estimate on the cardinality of the document sets we must join for the query operators. The second cost C_{merge} gives us an estimate of the number of positions that we must merge to compute the resulting posting for the query operators.

Cost of Vertical Cover Operator ($\boxbar$). The cardinality of the result of the vertical cover operator is directly proportional to the union of the number of documents associated with the left and right operand. Formally, it can be expressed in terms of document frequency (df(•)) as follows:

$$|a \boxbar b| \propto \Big(\text{df}(a) + \text{df}(b) - \text{df}(a \cap b)\Big). \tag{4.15}$$

The cost associated with the vertical cover operator can be determined using the number of common n-grams (skip-grams) (see cover(•) in Algorithm 6) that we can uncover from each phrase set of the HPQ. In case of no overlap, cost degenerates to querying the n-gram (skip-gram) inverted indexes for each n-gram (skip-gram) decomposition of the phrases in each phrase set. We can put this formally as:

$$\text{cost}(a \boxbar b) \propto \sum_{x \in \text{cover}(a,b)} \Big(C_{\text{join}} \cdot \text{df}(x) + C_{\text{merge}} \cdot \text{cf}(x)\Big).$$

Cost of Horizontal Order Operator ($\boxminus$). The cardinality of the result of the horizontal order operator is proportional to the intersection of the number of documents associated with the left and right operand. Formally, this can be expressed in the number of documents (df(•)) as follows:

$$|a \boxminus b| \propto \Big(\text{df}(a) + \text{df}(b) - \text{df}(a \cup b)\Big). \tag{4.16}$$

The cost associated with the horizontal order operator depends on the number of skip-gram combinations generated across the phrase sets. To obtain the ordering using skip-grams for the horizontal order operator, we need to generate skip-grams between each two consecutive phrase sets in the HPQ (see order(•) in Algorithm 7). We can model the cost in two ways. First, if it is required that we optimize both for the number of documents and positions, we can write the cost as:

$$\text{cost}(a \boxminus b) \propto \operatorname*{arg\,min}_{x \in \text{order}(\langle a,b \rangle)} \Big(C_{\text{join}} \cdot \text{df}(x) + C_{\text{merge}} \cdot \text{cf}(x)\Big).$$

	P_1	P_2	P_3	P_4	P_5
P_1	100	1400	2125	3400	3790
P_2	×	750	1750	3275	4465
P_3	×	×	175	1975	3415
P_4	×	×	×	725	3975
P_5	×	×	×	×	390

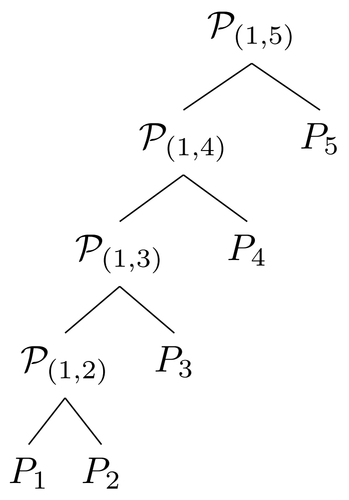


Figure 4.6: An illustrative example of generating the optimal execution plan. For the example, consider the cost constant ratio to be $C_{\text{join}}/C_{\text{merge}} = 10$ with following cardinalities of the $\boxminus$ operator $|P_1 \boxminus P_2| = 50$, $|P_2 \boxminus P_3| = 75$, $|P_3 \boxminus P_4| = 100$, and $|P_4 \boxminus P_5| = 300$. Assume that each resulting text regions has only 50 positional spans.

However, if the number of documents are equal in both the operands, then we are interested in minimizing the number of positions to be merged:

$$\text{cost}(a \boxminus b) \propto \underset{x \in \text{order}(\langle a,b \rangle)}{\arg\min} \left(\frac{C_{\text{merge}} \cdot \text{cf}(x)}{C_{\text{join}} \cdot \text{df}(x)} \right). \tag{4.17}$$

Cost Comparison between $\boxbar$ and $\boxminus$. In general, computing the intersection over large sets of documents is more expensive than computing the union. The common document identifiers can be identified via three strategies. The first strategy uses binary search for looking up an identifier from the first list in the second list of ordered document identifiers. The second strategy is to use galloping (interpolation) search in order to speed up the lookups in ordered lists. The third and final strategy is to use hash-set-based intersections to reduce the time for intersection at the cost of performance degradation due to collisions. The first two strategies benefit from data locality however, on modern hardware architectures, all perform comparably well [69]. Our implementation uses the third strategy. Furthermore, the horizontal order operator leads to a Cartesian product of sets if distributed over the vertical cover operator. That is, $(a \boxbar b) \boxminus (c \boxbar d) = (a \boxminus c) \boxbar (a \boxminus d) \boxbar (b \boxminus c) \boxbar (b \boxminus d)$. Given these two characteristics of the horizontal order operator it is more expensive to compute than the vertical cover operator. Formally,

$$\text{cost}(\boxbar) \ll \text{cost}(\boxminus). \tag{4.18}$$

4.7.2 Optimization

Let $\mathcal{P} = \langle P_1, P_2, \ldots, P_n \rangle$ be a hyper-phrase query. Where each phrase set $P_i = \{p^i_1, p^i_2, \ldots, p^i_m\}$ has at most m phrases. Using the aforementioned query operators we can specify the following operator sequence for executing the query $\mathcal{P}$:

$$(p^1_1 \boxbar p^1_2 \ldots p^1_m) \boxminus (p^2_1 \boxbar p^2_2 \ldots p^2_m) \boxminus \ldots \boxminus (p^n_1 \boxbar p^n_2 \ldots p^n_m). \tag{4.19}$$

We can rephrase the above formulation as follows:

$$\mathop{\Large\boxminus}_{j=1}^{n} P_j = \mathop{\Large\boxminus}_{j=1}^{n} \mathop{\Large\boxbar}_{i=1}^{m} p^j_i \tag{4.20}$$

where, P represents the result of the vertical cover operator. At a high level, it may seem trivial to execute the vertical cover operators ($\boxbar$) first to obtain the union of the documents representing the phrase sets (P_j) and subsequently the horizontal order operators ($\boxminus$) to obtain the result of sequences of one phrase set following the other ($P_{j-1} \boxminus P_j$). However, naïvely executing this query plan may be expensive. This is because, we may end up intersecting two potentially large posting lists. This can be avoided if we chose to perform a more selective operand (i.e., operand with fewer postings) with an operand with a large posting list in order to eliminate those documents that will not end up in the final result.

To understand this better, consider spotting the following KG fact: ⟨ BILL CLINTON, EDUCATED AT, USA ⟩. Let the hyper-phrase query associated with this be: ⟨{*bill clinton*, *william jefferson clinton*, *clinton*}, {*educated at*, *studied at*}, {*united states america*, *united states*}⟩. To show the difference between the number of comparisons incurred when executing different order of operator sequences consider the following example (where the words in the phrases are represented by their first letter, subscripts denote the number of documents, and number of positions per document):

$$(b \boxbar w \boxbar j \boxbar c)_{10^2 \times 10} \boxminus (e \boxbar s \boxbar a)_{10^5 \times 10^2} \boxminus (us \boxbar a')_{10 \times 10^5} .$$

If we evaluate the horizontal order between the subject and predicate first we incur a cost of $O(10 \times 10^2)$ comparisons. However, if we perform the horizontal order between the predicate and object first we incur a cost of $O(10^2 \times 10^5)$ comparisons. There is an order of 10^4 extra comparisons incurred when executing the second operator first.

Optimizing Vertical Cover Operators. We now describe the initial optimizations that we can perform by leveraging the associativity and commutativity of the vertical cover operator. The first optimization corresponds to decomposing the phrases using the most cost-effective cover of the phrase-sets using n-grams or skip-grams. This can be done in a greedy manner by leveraging the dictionaries.

The second optimization that we can make is to keep track of common phrases across phrase-sets. We therefore reduce the cost of the redundant retrieval of same posting lists for phrases. This optimization leverages the left distributive property of the vertical cover operator over the horizontal order operator.

Optimizing Horizontal Order Operators. The third and final optimization we can make concerns the horizontal order operator. To optimize the sequence of horizontal order operator $\boxminus$, we first need to uncover the **optimal substructure property** underlying the hyper-phrase queries. It is important to note that a subsequence of phrase sets in a HPQ $\mathcal{P} = \langle P_1, P_2, \ldots, P_n \rangle$ is yet another HPQ. To optimize the original HPQ, we must first identify the optimal sequence of performing the horizontal order operators for sub-hyper-phrase queries that it contains. This problem is similar to that of identifying the optimal order of multiplying a sequence of compatible matrices [65] and optimizing the join order for SQL queries in relational databases [185]. A naïve method of computing such an optimal solution is lower bounded by the *Catalan Numbers* – $\Omega(4^n/3^{3/2})$ [65]. Let a sub-hyper-phrase query be denoted by $\mathcal{P}_{(i,j)}$ where the subscripts denote the subsequence (without omissions) of phrase sets from the original hyper-phrase query $\mathcal{P}$. The optimal join order sequence can then be stated as:

$$\text{opt}(\mathcal{P}_{(i,j)}) = \begin{cases} \text{cost}(\boxbar_{l=1}^{m} p_l^i), & \text{if } i = j \\ \min_{i \le k < j} \Big[\text{opt}(\mathcal{P}_{(i,k)}) + \text{opt}(\mathcal{P}_{(k+1,j)}) + \\ \qquad \text{cost}\big(\mathcal{P}_{(i,k)} \boxminus \mathcal{P}_{(k+1,j)}\big) \Big], & \text{if } i < j. \end{cases}$$

The equation above decomposes the optimization of a HPQ $\mathcal{P}_{(1,n)}$ into smaller hyper-phrase queries. The base case of the inductive hypothesis corresponds to computing the cost of the vertical cover operator over each individual phrase set. The induction step builds up the dynamic programming table by first computing the join between a hyper-phrase query with only two phrase sets. Thereafter, we seek to intersect the solutions to those hyper-phrase queries that consist of the least number of documents. Figure 4.6 shows an example of computing an optimal join order sequence.

4.8 Evaluation

We describe the document collections used to test our approach, how we generated a testbed of hyper-phrase queries using knowledge graphs, and finally the baselines and methods put under test. We end the section by discussing the experimental results obtained.

COLLECTION	$n_{\text{documents}}$	n_{words}	$n_{\text{sentences}}$
NEW YORK TIMES	1,855,623	1,058,949,098	54,024,146
WIKIPEDIA	5,327,767	2,807,776,276	192,925,710
GIGAWORD	9,870,655	3,988,683,648	181,386,746
GDELT	14,320,457	6,371,451,092	297,861,511

Table 4.1: Document collection statistics.

TYPE	NYT	WIKIPEDIA	GIGAWORD	GDELT
UNIGRAM	5.83	22.22	31.10	50.37
BIGRAM	16.27	46.72	68.15	99.05
TRIGRAM	39.98	104.77	94.87	147.68
SKIP-GRAM	79.62	222.52	191.43	259.10

Table 4.2: Inverted Index build times in minutes.

	NYT	WIKIPEDIA	GIGAWORD	GDELT
COLLECTION SIZE	3.00	21.89	9.10	77.44
INDEX TYPE	NYT	WIKIPEDIA	GIGAWORD	GDELT
WORD DICTIONARY	0.04	0.30	0.10	0.14
N-GRAM DICTIONARIES	4.54	19.80	10.50	19.04
SKIP-GRAM DICTIONARY	14.40	25.70	21.30	29.30
WORD INDEX	5.80	18.00	22.30	35.90
N-GRAM INDEXES	45.90	126.30	154.40	234.80
SKIP-GRAM INDEX	56.10	180.80	203.60	289.00

Table 4.3: Dictionary and Index sizes in Gigabytes (GB).

4.8.1 Document Collections

We build our proposed indexes over four large document collections. The first document collection consists of news articles published in the New York Times and is publicly available [18]. The NYT archive consists of approximately two million documents published between 1997 and 2007. The second document collection we utilize is Wikipedia [23]. Wikipedia comprises of around five million documents and is also publicly available. The third document collection consists of news articles published by seven major newswire organizations including the Washington Post, the Associated Press, and the Xinhua News Agency during the 1995-2010 reporting period [5]. This news archive is publicly available as the fifth edition of the English Gigaword. It consists of around ten million documents. The fourth and final document collection is the largest in our evaluation setup. It comprises of a set of news articles obtained by crawling the links available on the real-world event repository GDelt [7]. The GDelt document collection amounts to a total of approximately fourteen million documents. In total, the four document collections amount to more than thirty million documents. A summary of the document collection statistics is given in Table 4.1.

4.8.2 Indexes

For each document collection, we created indexes based on the data model discussed in Section 4.5. For n-grams, we instantiated indexes for unigrams, bigrams, and trigrams. For skip-grams, we instantiated indexes that record skip-grams with word separation of up to ten words (i.e., $\ell = 10$). Corresponding to each collection the dictionary and index sizes in GB are reported in Table 4.3. We also show the sizes of the word dictionary and inverted index for each of the collection in Table 4.3. The index build times are shown in Table 4.2. It can be observed by storing higher length n-grams we blow up our indexes by a factor of at most 7.91×. While, the blow up for the skip-gram index is at most 10.04× compared to a standard word (unigram) index. Despite the large index sizes, we observe that by lowering the word separation ℓ between skip-grams we can reduce the skip-gram index sizes.

4.8.3 Hyper-Phrase Queries

In order to test the efficiency of our proposed approach we utilize the Wikidata KG [22] to construct a testbed of hyper-phrase queries. The translation of facts in the KG to hyper-phrase queries are done via the following scheme: $\mathcal{P} \equiv \langle \mathsf{S}, \mathsf{P}, \mathsf{O} \rangle$. We evaluate our proposed method and baselines against two kinds of tasks: KG fact and KG subgraph spotting task.

Queries for KG Fact Spotting (KG-F) Task

For this task, each test instance consists of a HPQ corresponding to a single KG fact with multiple object arguments. Each query consists of three phrase sets where each phrase is an alias for the subject, predicate, or object. To instantiate the instances for this task, we pursued those entities where multiple objects for the same predicate could be associated. Concretely, we constructed instances for categories and predicates in Table 4.4. For each instance in this task we record the time to retrieve the text regions as evidences for each HPQ. An example of an instance in the KG fact spotting task is shown in Figure 4.2.

Queries for KG Subgraph Spotting (KG-S) Task

Journalists and scholars in humanities are seldom interested in retrieving evidences for a single KG fact. It is often required that one can query for relationships between multiple entities in a single query. In order to simulate hyper-phrase queries with more than three phrase sets we consider KG subgraphs. In particular, we restrict ourselves to subgraphs with a star topology. To construct queries for the task of KG-S, we take each fact concerning an entity (subject) as a constituent HPQ. Thus, each instance in KG-S task is a list of hyper-phrase queries, where each HPQ represents a fact concerning a common entity. For

CATEGORY	PREDICATES	n_{queries}	μ_{words}
WRITERS	AWARD RECEIVED, NOTABLE WORK	694	57.95
MEDICINE LAUREATES	AWARD RECEIVED, EMPLOYER	410	55.02
PHYSICS LAUREATES	AWARD RECEIVED, EMPLOYER	406	58.48
CHEMISTRY LAUREATES	AWARD RECEIVED, EMPLOYER	348	56.04
MOVIES	CAST MEMBER, FILMING LOCATION	114	75.51
ALL US ELECTIONS	CANDIDATE	1	1081.00
ALL WORLD WAR I BATTLES	LOCATION	1	1669.00
ALL WORLD WAR II BATTLES	LOCATION	1	2563.00
ALL SUMMER OLYMPICS	LOCATION	1	717.00
ALL WINTER OLYMPICS	LOCATION	1	407.00

Table 4.4: Statistics regarding the testbed for KG-F task.

each instance in this task, we retrieve the text regions as evidences for each HPQ in the list and record the total time to execute all of the constituent HPQ.

To materialize the instances for the KG-S task we focus on prominent named entities. The prominence is chosen by restricting ourselves to famous scientists, artists, and athletes. To select these named entities we looked at the prestigious awards won by scientists (e.g., the Nobel Prize), artists (e.g., the Grammy), and athletes (e.g., medal at the Summer Olympics). The concrete Wikidata object identifiers corresponding to these awards is shown in Table 4.5. By obtaining a set of entities where the award occurs as an object, we then focused on common and selective key predicates to further narrow down the facts concerning these entities. The key predicates that we utilized were: *P19* (PLACE OF BIRTH), *P10* (OCCUPATION), *P166* (AWARDS RECEIVED), *P69* (EDUCATED AT), *P800* (NOTABLE WORK), *P22* (FATHER), *P26* (SPOUSE), *P361* (PART OF), *P39* (POSITION HELD), and *P102* (MEMBER OF POLITICAL PARTY). The number of facts thus created are reported in Table 4.5. An illustrative example of an instance of the KG subgraph spotting task is shown in Figure 4.1.

It is important to note that queries in both KG fact and subgraph spotting tasks are highly verbose. For instance, the most complex query comprising of all US elections and their candidates comprises of at least a thousand words (see Table 4.4). The testbed for the KG-F and KG-S tasks is available at the following URL:

`http://resources.mpi-inf.mpg.de/dhgupta/data/hpq/`.

CATEGORY	KG OBJECT ID	n_{queries}	μ_{words}
ACTRESSES (838)	*Q103618*	1,434	32.72
ACTORS (840)	*Q103916*	1,547	35.00
SINGERS (293)	*Q41254*	327	33.86
WRITERS (743)	*Q37922*	2,316	33.69
PRESIDENT OF THE US (807)	*Q11696*	367	47.22
PHYSICISTS (654)	*Q38104*	2,056	34.12
CHEMISTS (634)	*Q44585*	1,788	33.99
MATHEMATICIANS (58)	*Q28835*	189	33.19
ECONOMISTS (84)	*Q47170*	274	37.57
PACIFISTS (593)	*Q35637*	1,945	36.28
OLYMPIANS (27)	*Q15243387* *Q15889641* *Q15889643*	144	31.79

Table 4.5: Statistics regarding the testbed for KG-S task.

4.8.4 Setup

Implementation and Hardware Details. The entire indexing infrastructure has been built from scratch in Java with indexes stored in a cluster of machines running Cloudera CDH 5.90 version of Hadoop and HBase. Our Hadoop cluster consists of twenty machines in which all machines have up to 24 core Intel Xeon CPUs at 3.50 GHz, up to 128 GB of primary memory, and up to eight 4 TB HDDs. With the Hadoop cluster acting as our storage backend, we evaluate our queries on a high-memory compute node equipped with 1.48 TB of primary memory and a 96 core Intel Xeon CPU with processing speed of 2.66 GHz.

Baselines and Systems. We evaluate three baselines against our proposed approach. We first establish a lower bound on how much time a HPQ should take to answer by scanning the entire document collection on our Hadoop cluster in an embarrassingly parallel manner B_{SCAN}. As a second naïve baseline $\mathsf{B}_{\text{BIGRAM}}$ we retrieve the results corresponding to each individual phrase set in the HPQ using unigram and bigram decomposition only. We use bigrams to construct the result set for phrases as we do not index stopwords in the word index. Subsequent to this we compute the result set only for those document identifiers that are present in all the phrase sets of the HPQ. As a third baseline $\mathsf{B}_{\text{NGRAM}}$ we consider the longest possible n-gram decomposition possible with our indexes. The result is computed in a similar manner as in $\mathsf{B}_{\text{BIGRAM}}$. The $\mathsf{B}_{\text{BIGRAM}}$ and $\mathsf{B}_{\text{NGRAM}}$ baselines are in accordance with the discussion presented in Section 4.3. As our first system, $\mathsf{A}_{◫}$ we test our approach that searches for hyper-phrase queries by only leveraging the optimized vertical cover operator across phrase sets. As our second system $\mathsf{A}_{◫⊟}$ we execute each HPQ using the optimized vertical cover and horizontal order operators.

SYSTEM	N-GRAM DICTIONARY	N-GRAM INDEX	SKIP-GRAM DICTIONARY	SKIP-GRAM INDEX
$\mathsf{B}_{\mathsf{SCAN}}$	×	×	×	×
$\mathsf{B}_{\mathsf{BIGRAM}}$	×	●	×	×
$\mathsf{B}_{\mathsf{NGRAM}}$	×	●	×	×
$\mathsf{A}_{\text{◫}}$	●	●	×	×
$\mathsf{A}_{\text{◫⊟}}$	●	●	●	●

Table 4.6: Baselines and Systems.

SYSTEM	NYT	WIKIPEDIA	GIGAWORD	GDELT
$\mathsf{B}_{\mathsf{SCAN}}$	111.00	322.00	396.00	604.00

Table 4.7: Results for Baseline $\mathsf{B}_{\mathsf{SCAN}}$ (secs).

Parameters and Setup. For both the KG-F and KG-S task, we sample 100 queries to execute for the two baselines and our systems. For the remaining baseline $\mathsf{B}_{\mathsf{SCAN}}$ we measure the time required to scan the each document in the entire document collection once to establish a lower bound. We evaluate the sampled queries for three rounds with cold cache settings. To simulate the cold cache setting: we shuffle the order of queries in between rounds. We only show cold cache run times as they are similar to the warm cache setting results. Furthermore, for computing the optimal plan of execution we set the constants ratio $({}^{C_{\text{join}}}/_{C_{\text{merge}}})$ to 1 and we use the cost for horizontal-order operator in accordance with to Equation 4.17. The rationale for choosing this ratio was to consider the cost for merging positions same as documents as we have applied the naïve query optimization to all baselines and systems. We additionally vary the match between phrase sets to lie within a distance of $k \in \{0, 2, 5\}$ sentences of the document containing the evidence.

4.8.5 Results

A summary of the dictionaries and indexes used by the baselines and systems is shown in Table 4.6. We have further computed the statistical significance of the results using the Student's paired t-test at significance level $\alpha = 0.05$. The systems that produce statistically significant results over the $\mathsf{B}_{\mathsf{BIGRAM}}$ baseline are marked with $\triangle$ and over the $\mathsf{B}_{\mathsf{NGRAM}}$ are marked with $\blacktriangle$. The results for the first baseline $\mathsf{B}_{\mathsf{SCAN}}$ are shown in Table 4.7. As expected the time to scan the document collection directly depends on the collection size. The NYT being the smallest takes the least time while GDelt takes the most time on our Hadoop cluster to scan.

Results of KG Fact Spotting (KG-F) Task

We first discuss the efficiency results of the KG fact spotting task. In this task, we are required to retrieve text regions for a sequence of phrase sets that correspond to a KG fact. The results for the baselines and systems are displayed in Table 4.8. When restricting ourselves to matching phrase sets within a sentence we notice a speed up of at least 8.97$\times$ using bigrams $\mathsf{B}_{\text{BIGRAM}}$ over the simple B_{SCAN} baseline. Further using trigrams to spot phrases for matching a HPQ brings about a speed up of at least 42.42$\times$ over B_{SCAN}. Using our proposed optimization we can achieve a speed up of at least 1.22$\times$ over $\mathsf{B}_{\text{NGRAM}}$, 4.04$\times$ over $\mathsf{B}_{\text{BIGRAM}}$, and 53.31$\times$ over B_{SCAN}. As we increase the sentence relaxation k size for matching a HPQ we see a speed up of at least 1.15$\times$ over $\mathsf{B}_{\text{NGRAM}}$ and 3.72$\times$ over $\mathsf{B}_{\text{BIGRAM}}$. It is also important to observe that with increasing collection sizes the difference between end-to-end runtime results between the best baseline $\mathsf{B}_{\text{NGRAM}}$ and our system $\mathsf{A}_{\boxbar\boxminus}$ increases significantly.

Results of KG Subgraph Spotting (KG-S) Task

We now discuss the efficiency results of the KG subgraph spotting task. In this task, we are required to retrieve text regions for HPQs that correspond to multiple facts concerning an entity. The results for the baselines and systems are displayed in Table 4.9. For the KG subgraph spotting task, when restricting ourselves to matching each constituent fact of a subgraph to within a sentence, the baseline $\mathsf{B}_{\text{BIGRAM}}$ achieves a speed up of at least 9.00$\times$ over B_{SCAN}. While, the baseline $\mathsf{B}_{\text{NGRAM}}$ achieves a speed up of at least 17.85$\times$ over B_{SCAN}. The improvements offered by our proposed optimizations add up to reflect a speed up of at least 1.21$\times$ and at most 1.71$\times$ over $\mathsf{B}_{\text{NGRAM}}$. As we increase

RUNTIME RESULTS FOR KG-F TASK (SECS) FOR $k = 0$.

SYSTEM	NYT	WIKIPEDIA	GIGAWORD	GDELT
$\mathsf{B}_{\text{BIGRAM}}$	7.77 $\pm$ 12.91	24.41 $\pm$ 29.43	41.24 $\pm$ 62.94	67.34 $\pm$ 164.75
$\mathsf{B}_{\text{NGRAM}}$	1.80 $\pm$ 2.82	7.59 $\pm$ 6.37	7.89 $\pm$ 7.16	12.47 $\pm$ 17.98
$\mathsf{A}_{\boxbar}$	$^{\vartriangle}$1.92 $\pm$ 3.63	$^{\vartriangle}$7.21 $\pm$ 5.63	$^{\vartriangle}$7.74 $\pm$ 7.30	$^{\vartriangle}$11.57 $\pm$ 16.36
$\mathsf{A}_{\boxbar\boxminus}$	$^{\vartriangle\blacktriangle}$1.41 $\pm$ 2.42	$^{\vartriangle\blacktriangle}$6.04 $\pm$ 5.57	$^{\vartriangle\blacktriangle}$6.45 $\pm$ 6.50	$^{\vartriangle\blacktriangle}$9.78 $\pm$ 15.35

RUNTIME RESULTS FOR KG-F TASK (SECS) FOR $k = 2$.

SYSTEM	NYT	WIKIPEDIA	GIGAWORD	GDELT
$\mathsf{B}_{\text{BIGRAM}}$	8.61 $\pm$ 18.71	21.8 $\pm$ 28.25	34.77 $\pm$ 48.91	42.19 $\pm$ 68.97
$\mathsf{B}_{\text{NGRAM}}$	1.91 $\pm$ 3.33	7.11 $\pm$ 5.87	8.46 $\pm$ 8.66	10.50 $\pm$ 11.08
$\mathsf{A}_{\boxbar}$	$^{\vartriangle}$1.82 $\pm$ 2.87	$^{\vartriangle}$6.97 $\pm$ 6.06	$^{\vartriangle}$8.21 $\pm$ 7.63	$^{\vartriangle}$9.90 $\pm$ 10.31
$\mathsf{A}_{\boxbar\boxminus}$	$^{\vartriangle\blacktriangle}$1.52 $\pm$ 2.82	$^{\vartriangle\blacktriangle}$5.86 $\pm$ 5.44	$^{\vartriangle\blacktriangle}$6.92 $\pm$ 6.81	$^{\vartriangle\blacktriangle}$7.65 $\pm$ 8.97

RUNTIME RESULTS FOR KG-F TASK (SECS) FOR $k = 5$.

SYSTEM	NYT	WIKIPEDIA	GIGAWORD	GDELT
$\mathsf{B}_{\text{BIGRAM}}$	10.45 $\pm$ 30.12	35.18 $\pm$ 120.41	39.93 $\pm$ 59.29	71.81 $\pm$ 233.71
$\mathsf{B}_{\text{NGRAM}}$	2.08 $\pm$ 4.64	10.02 $\pm$ 32.68	8.41 $\pm$ 7.42	12.92 $\pm$ 22.46
$\mathsf{A}_{\boxbar}$	$^{\vartriangle}$2.56 $\pm$ 6.06	$^{\vartriangle}$9.47 $\pm$ 30.57	$^{\vartriangle}$10.17 $\pm$ 11.39	$^{\vartriangle}$13.71 $\pm$ 25.07
$\mathsf{A}_{\boxbar\boxminus}$	$^{\vartriangle\blacktriangle}$1.64 $\pm$ 4.35	$^{\vartriangle\blacktriangle}$8.26 $\pm$ 28.87	$^{\vartriangle\blacktriangle}$7.07 $\pm$ 7.13	$^{\vartriangle\blacktriangle}$11.26 $\pm$ 23.68

Table 4.8: Results for KG-F Task (secs).

RUNTIME RESULTS FOR KG-S TASK (SECS) FOR $k = 0$.

SYSTEM	NYT	WIKIPEDIA	GIGAWORD	GDELT
B_{BIGRAM}	7.50 ± 8.67	35.77 ± 53.78	37.66 ± 76.66	49.70 ± 66.84
B_{NGRAM}	4.07 ± 4.86	18.04 ± 9.90	15.61 ± 15.92	21.18 ± 17.35
$A_{◫}$	$^{\triangle}3.96 \pm 4.17$	$^{\triangle}17.26 \pm 10.35$	$^{\triangle}14.71 \pm 11.81$	$^{\triangle}20.52 \pm 17.29$
$A_{◫⊟}$	$^{\triangle}2.91 \pm 4.22$	$^{\triangle\blacktriangle}10.53 \pm 7.92$	$^{\triangle}12.05 \pm 10.96$	$^{\triangle}17.45 \pm 14.83$

RUNTIME RESULTS FOR KG-S TASK (SECS) FOR $k = 2$.

SYSTEM	NYT	WIKIPEDIA	GIGAWORD	GDELT
B_{BIGRAM}	6.20 ± 7.40	32.76 ± 31.72	46.56 ± 107.71	57.10 ± 67.78
B_{NGRAM}	3.54 ± 4.11	17.79 ± 11.76	15.20 ± 15.74	21.38 ± 17.27
$A_{◫}$	$^{\triangle}3.32 \pm 4.28$	$^{\triangle}16.97 \pm 11.11$	$^{\triangle}14.39 \pm 12.80$	$^{\triangle}20.00 \pm 16.27$
$A_{◫⊟}$	$^{\triangle}2.98 \pm 5.58$	$^{\triangle\blacktriangle}10.20 \pm 7.74$	$^{\triangle}12.17 \pm 13.71$	$^{\triangle\blacktriangle}15.38 \pm 13.60$

RUNTIME RESULTS FOR KG-S TASK (SECS) FOR $k = 5$.

SYSTEM	NYT	WIKIPEDIA	GIGAWORD	GDELT
B_{BIGRAM}	9.33 ± 10.95	31.19 ± 34.11	38.09 ± 48.21	63.94 ± 72.30
B_{NGRAM}	4.47 ± 4.29	17.36 ± 12.28	15.12 ± 12.11	24.47 ± 18.47
$A_{◫}$	$^{\triangle}4.22 \pm 4.99$	$^{\triangle}16.86 \pm 11.04$	$^{\triangle}14.76 \pm 12.43$	$^{\triangle}23.36 \pm 18.83$
$A_{◫⊟}$	$^{\triangle\blacktriangle}3.06 \pm 4.06$	$^{\triangle\blacktriangle}9.91 \pm 7.81$	$^{\triangle\blacktriangle}11.66 \pm 9.89$	$^{\triangle\blacktriangle}18.47 \pm 16.06$

Table 4.9: Results for KG-S Task (secs).

the sentence relaxation k size for matching phrase sets in a KG subgraph we observe speed ups of at least 1.19× over B_{NGRAM}, 2.08× over B_{BIGRAM}, and 31.57× over B_{SCAN}. Just like the KG fact spotting task, we observe that with increasing collection size the benefit offered by proposed optimization is significant over the baselines B_{BIGRAM} and B_{NGRAM}.

Summary

At the task of spotting evidences for KG facts, our proposed optimizations show an improvement of at least 1.15× and at most 1.37× over B_{NGRAM} across different levels of sentence separations. At the task of spotting evidences for KG subgraphs, our proposed optimizations have shown an improvement of at least 1.19× and at most a speed up of 1.75× over B_{NGRAM}. We also observe that as we move across increasing collection sizes the benefit of optimization also becomes apparent. This speed up however, comes at a cost of maintaining n-gram indexes and skip-gram indexes that are a factor of at most 7.91× and 10.04× more larger than keeping only a word index. Despite this, we note that depending upon the application domain, selective choices regarding what kind of skip-grams to index and n-grams can further bring down the storage cost and at the same time offer speed ups based on our optimization for executing hyper-phrase queries.

4.9 Conclusion

We have shown how to speed up the processing of verbose hyper-phrase queries that help in establishing provenance for knowledge graph facts and subgraphs. Additionally, our approach shall find applications in knowledge acquisition for relationships and facts regarding out-of-knowledge-graph entities. For instance, when extracting sentences for adverse drug reactions in the field of biomedicine. Our solution consists of a data model for text that indexes n-grams and skip-grams along with their sentence identifiers. Furthermore, we presented operators to express the complete combinatorial space for optimizing hyper-phrase queries. We then described a dynamic programming based algorithm to generate an optimal query plan using the proposed vertical cover and horizontal order query operators. We showed the efficiency of our system in spotting evidences for knowledge graph facts and subgraphs in document collections amounting to more than thirty million documents.

PART II

QUERYING ANNOTATED DOCUMENT COLLECTIONS

CHAPTER 5

IDENTIFYING TIME INTERVALS OF INTEREST FOR QUERIES

• Dhruv Gupta and Klaus Berberich. Identifying Time Intervals of Interest to Queries. In CIKM 2014, pages 1835–1838.

5.1 Introduction

Time has been recognized as an important dimension in Information Retrieval [29], and recent years have seen an increased interest in making use of temporal information associated with documents or information needs. Tasks that have been tackled include retrieving recent relevant documents [144] as well as documents relevant to implicitly [155] or explicitly [41, 70] temporal queries. Beyond that, also web search engines have meanwhile deployed features to keep up with the changing Web, indexing recently published documents, and filtering results based on their publication dates.

In this chapter, we address the problem of automatically identifying time intervals of interest to a given keyword query. For instance, when presented with the keyword query *bill clinton presidency*, a good time interval to determine would be [1993, 2001], which covers the years of Clinton's presidency. This is a useful building block in temporal information retrieval with applications such as (i) temporal query reformulation and expansion – by adding time intervals of interest to the query, (ii) temporal diversification of search results – by making sure that the result covers diverse time intervals of interest to the query, and (iii) providing more structured query results to users – organized by important time intervals they refer to.

While ours is not the first effort in this direction, it differs from previous ones [70, 132] in several important aspects. First, our approach generates time intervals (e.g., [1993, 2001]) as opposed to generating time points at a fixed temporal granularity (e.g., the years 1993 and 2001) for keyword queries. Second, we make use of both documents' publication dates, as part of their metadata, as well as temporal expressions from their contents. Third, our approach is not restricted to a fixed temporal granularity but can determine time intervals of interest at different temporal granularities (e.g., day, month, and year). Finally, we also consider temporally ambiguous queries for which more than one time interval is of interest – say *george bush presidency* or *san francisco earthquake*.

The work in this chapter builds on prior research [41], which aims at improving retrieval effectiveness for explicitly temporal queries such as *summer olympics 2004*. Borrowing their formal model for representing temporal expressions contained in documents (e.g., in the summer of 2004) and capturing their inherent uncertainty, we put forward a generative model for identifying time intervals of interest to a given keyword query. Our model is based on the intuition that a time interval of interest should be often referred to in relevant documents. More specifically, it considers the top-k documents retrieved by a unigram language model, treating them as pseudo-relevant, and analyzes their contents, specifically the temporal expressions therein, for often referred to time intervals. We describe the design space and consider different concrete instantiations of our model. To evaluate their performance, we compile two novel testbeds, consisting of temporally unambiguous and temporally ambiguous queries obtained from high-quality web sources.

Contributions described in this chapter are:

1. a novel approach to identify time intervals of interest to a given keyword query;
2. two testbeds consisting of temporally (un)ambiguous queries which are made publicly available;
3. an experimental evaluation of our approach on The New York Times Corpus [18], as a publicly-available document collection, on the aforementioned query testbeds.

Organization. The rest of this chapter is organized as follows. We put our work in context with prior research in Section 5.2. Section 5.3 then describes our approach, including a discussion of the design space and details on our concrete instantiation. Following that, we describe our experimental evaluation in Section 5.4, before concluding in Section 5.5.

5.2 Related Work

In this section, we put our work in context with existing prior work. Kanhabua et al. [132] is the work closest to ours. In contrast to the approach put forward in this chapter, their method focuses on identifying years of interest to a keyword query and does so only based on documents' publication dates. Their method is thus restricted to time points at year granularity and cannot identify time intervals at other granularities. Dakka et al. [70] as well as Diaz and Jones [127], as one building block in their respective research, describe methods that identify time points of interest to a query. Their methods, though, are solely based on the publication dates associated with documents and do not consider temporal expressions from their contents. Again, no time intervals are considered and the granularity is limited to that of documents' publication dates. Strötgen et al. [186] look into the related problem of identifying salient temporal expressions from a document. Other work has looked into improving the result quality of implicitly or explicitly temporal queries. For the former, this includes Metzler et al. [155], who identify implicitly temporal queries within the query log of a web search engine, and Dakka et al. [70], who analyze the distribution of publication dates to identify implicitly temporal queries. Peetz et al. [172] is a related work that leverages bursts in the temporal distribution of publication dates to improve retrieval effectiveness. Berberich et al. [41], as the work mentioned in the introduction, targets explicitly temporal queries and leverages both documents' publication dates and temporal expressions. Our work is orthogonal and the time intervals that we identify can be used to augment the query and obtain better results with one of the aforementioned approaches. Finally, there has been work on attaching a time point or time interval to an entire document. Thus, de Jong et al. [72] determine the likely publication time of a document based on its language; Kanhabua et al. [133] make use of temporal expressions from documents' contents to the same end. Jatowt et al. [123], even in the absence of any temporal expressions, determine a so-called focus time for a document, which delimits the time period the document predominantly refers to. For all of these approaches, the focus is on identifying a single time point or time interval (as opposed to possibly more than one) for a given document (as opposed to a query in our case).

5.3 Identifying Interesting Time Intervals

In this section, we describe our approach for identifying interesting time intervals for a given keyword query.

5.3.1 Document Model

We largely adopt the formal model and notation introduced by [41]. Our document collection is denoted by $\mathcal{D}$. A document $d \in \mathcal{D}$ consists of a multiset of keywords d_{text} and a multiset of temporal expressions d_{time}. We let $\text{tf}(v, d)$ and $\text{tf}(T, d)$ denote the term frequency of the keyword v and the temporal expression T in document d, respectively. We use $|d_{\text{text}}|$ and $|d_{\text{time}}|$ to denote the multiset cardinalities of the textual and temporal part, respectively. In the remainder, when it is clear from the context, we simply write d to refer to either of them. Keywords are drawn from a vocabulary $\mathcal{V}$.

5.3.2 Time Model

To incorporate temporal uncertainty we adopt the time model from [41]. A temporal expression is a four-tuple, $T = \langle b_l, b_u, e_l, e_u \rangle$. Where, $[b_l, b_u]$ and $[e_l, e_u]$, represent the lower and upper bounds on beginning of time interval, b, and its end, e, respectively. Each component of T is drawn from a time domain $\mathcal{T}$ (usually $\mathbb{Z}$). A temporal expression T may refer to any time interval $[b, e] \in \mathcal{T} \times \mathcal{T}$ with $b_l \leq b \leq b_u$, $e_l \leq e \leq e_u$, and $b \leq e$. For example the temporal expression *"in the 1960s"* would be represented as $T = \langle 1960 - 01 - 01, 1969 - 12 - 31, 1960 - 01 - 01, 1969 - 12 - 31 \rangle$ and time interval such as $[1965 - 05 - 10, 1966 - 04 - 09]$ can be generated from T. We treat temporal expressions as a set of time intervals and let $|T|$ denote the number of time intervals that $|T|$ may refer to. Figure 5.1 illustrates the time model with uncertainty.

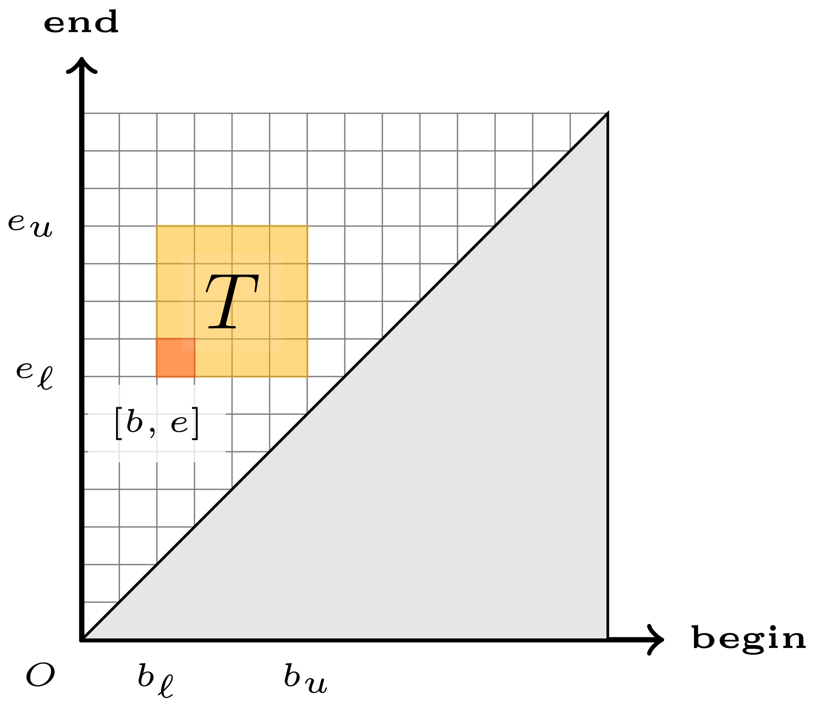


Figure 5.1: Graphical representation illustrating how a time interval $[b, e]$ is generated from T [41].

5.3.3 Retrieval Model

As mentioned above, our approach determines time intervals of interest to a query based on pseudo-relevant documents. To determine those, we use a unigram language model with Dirichlet smoothing and thus estimate the query likelihood of a given keyword query q as

$$P(\, q \mid d\,) = \prod_{v \in q} \frac{\mathrm{tf}(v,d) + \mu \cdot \frac{\mathrm{tf}(v,\mathcal{D})}{|\mathcal{D}|}}{|d| + \mu} \; . \tag{5.1}$$

Here, $\mathcal{D}$ is the document collection, treated as a single document, for the purpose of smoothing probability estimates.

5.3.4 Time Intervals of Interest

Having identified documents believed to be relevant to the keyword query q, our approach analyzes their contents to determine time intervals of interest. We next describe the high-level components of our approach, before discussing possible instantiations.

Intuitively, a time interval $[tb,\, te]$ is considered interesting for a keyword query q, if it is frequently referred to by highly relevant documents. We cast this intuition into the following generative model:

$$P(\,[tb,\, te] \mid q\,) \;=\; \sum_{d \,\in\, \mathrm{top}(q,k)} P(\,[tb,\, te] \mid d\,)\, P(\, d \mid q\,) \tag{5.2}$$

According to this model, first a document d is selected from $\mathrm{top}(q,k)$ as the set of k documents having highest likelihood of generating the keyword query q. Second, a time interval $[tb,\, te]$ is generated from the temporal expressions contained in document d. For each of the two steps, we consider different design alternatives. To obtain time intervals of interest at different granularities, the generative model is applied recursively. By doing so, we obtain time intervals of interest at year, month, and day granularity. This is illustrated in Figure 6.1.

Generating Documents

In the simplest case, in the first step, a document is selected at uniform random among the top-k results, yielding

$$P(\, d \mid q\,) = 1/k \; . \tag{5.3}$$

Here, the query likelihood $P(\, q \mid d\,)$ is thus not taken into account. While this may not be a problem for small choices of k, we expect it to deteriorate performance for larger choices. As an alternative, we consider

$$P(\, d \mid q\,) = \frac{P(\, q \mid d\,)}{\sum_{d' \,\in\, \mathrm{top}(q,k)} P(\, q \mid d'\,)} \;, \tag{5.4}$$

which estimates the probability of selecting a document in the first step as proportional to its query likelihood estimated according to Equation 5.1.

Generating Time Intervals

For the second step, we can estimate the probability of generating the time interval $[tb,\, te]$ from document d as

$$P(\,[tb,\, te] \mid d\,) = \frac{1}{|d_{\mathrm{time}}|} \sum_{T \,\in\, d_{\mathrm{time}}} \mathbb{1}(\,[tb,\, tb,\, te,\, te] = T\,)\;. \tag{5.5}$$

The time interval $[tb,\, te]$ can thus only be generated from documents containing temporal expressions that exactly map to it. To illustrate this, the time interval [1992, 1998] can only be generated from documents that contain from 1992 until 1998 but not from documents containing only in the 1990s. As a more relaxed advanced alternative, building on the generative model introduced in [41], we also consider

$$P(\,[tb,\, te] \mid d\,) = \frac{1}{|d_{\mathrm{time}}|} \sum_{T \,\in\, d_{\mathrm{time}}} \frac{\mathbb{1}(\,[tb,\, te] \in T\,)}{|T|} \;, \tag{5.6}$$

which takes into account the uncertainty inherent to temporal expressions. With this model, also a document containing 1990s, formally represented as $\langle 1990,\, 1999,\, 1990,\, 1999 \rangle$, could generate the time interval [1992, 1998].

Query Processing

At query time, our method first determines the set $\mathrm{top}(q,k)$ of documents having highest query likelihoods. It then analyzes the temporal expressions therein, determining t_{min} and t_{max} corresponding, respectively, to the earliest and latest time mentioned in any of the result documents. Following that, it enumerates all valid time intervals $[tb,\, te] \subseteq [t_{\mathrm{min}},\, t_{\mathrm{max}}]$ and determines their probability $P(\,[tb,\, te] \mid d\,)$. For this last step, combining the two design alternatives for each of the two steps of our generative model, we obtain four possible instantiations, which we experimentally evaluate in the following section. We will use **N** and **A** to refer to the *naïve* and *advanced* design alternative for each of the two steps. The method combining Equation 5.4 and Equation 5.5, for example, will be referred to as **AN**.

5.4 Evaluation

We now describe the setup of our experiments.

5.4.1 Setup and Datasets

Document Collection. As a document collection, we use The New York Times Annotated Corpus [18], which consists of about 2 million news articles published between 1987 and 2007. Publication dates are readily available. Temporal expressions are obtained from the data provided by [41] – they used TARSQI [200] to annotate temporal expressions augmented by a handful of handcrafted regular expressions to go after range expressions (e.g., from 1980 until 1984). Publication dates of documents are taken into account as additional temporal expressions – thus a document published on March 13, 1988 virtually contains the temporal expression on March 13, 1988.

Sports	*commonwealth games* (21) | *asian games* (18) | *summer olympics* (34) | *winter olympics* (26) | *super bowl winners* (48)
Music	*u2 album* (13) | *nirvana album* (4) | *beatles album* (52) | *red hot chilli peppers album* (11) | *michael jackson album* (11)
Movies	*harry potter movie* (6) | *oscar academy awards* (88) | *lord of the rings movie* (3)
Politics	*german federal elections* (19) | *us presidential elections* (58) | *australia federal elections* (45)
History	*iraq war* (2) | *world trade center bombing* (2) | *madrid bombing* (9) | *earthquake united states of america* (73)

Table 5.1: Temporally ambiguous queries

Queries. We use two sets of test cases: **(i)** *temporally unambiguous queries* obtained from the "On this Day" website of The New York Times[1]. For each day of the year, this website lists an event of historic significance, including a concise description. For example, for July 1st, the event is described as *"In 1997, Hong Kong reverted to Chinese rule after 156 years as a British colony."*. We extract the indicated year (here: 1997) for each date to obtain a precise date at day granularity and keep the rest of the description as a query. This leaves us with a total of 366 temporally unambiguous queries; **(ii)** *temporally ambiguous queries* from the domains of Sports, Music, Movies, Politics, and History, which we compiled manually. For each of them, we consult Wikipedia to find out the associated time intervals at day granularity. The obtained set of 20 queries is given in Table 5.1. Here, the number of associated time intervals is given in parentheses, indicating the degree of ambiguity of each

[1]http://learning.blogs.nytimes.com/on-this-day/

query. In the interest of repeatability, both query sets, including associated time intervals are made available at the following URL:

`http://www.mpi-inf.mpg.de/~kberberi/data/cikm2014` .

Methods under comparison are the four combinations of the naïve and advanced models delineated in Section 5.3, referred to as **NN**, **AN**, **NA**, and **AA**. We can not sensibly compare against [132] as a baseline, since their method is based on publication dates and year granularity. For each of the methods under comparison, we set the smoothing parameter of the unigram language model as $\mu = 1000$ and vary the number of pseudo-relevant documents retrieved as $k = \{\, 25,\ 50,\ 100 \,\}$. We consider three different temporal granularities (day, month, year) in our experiments. When going for a coarser granularity (e.g., year), temporal expressions, which are natively stored at day granularity, are systematically coarsened. As a concrete example, the temporal expression $\langle\, 1998-01-01,\ 1998-12-31,\ 1998-01-01,\ 1998-12-31 \,\rangle$ would be converted into $\langle\, 1998,\ 1998,\ 1998,\ 1998 \,\rangle$ at year granularity. The same procedure is applied to the ground-truth time intervals of our query test cases.

Measures. We use Precision@k (P@k) as a measure of retrieval effectiveness. For the sake of comparability, we report P@1 and P@5 for both the unambiguous and ambiguous queries – instead of using mean reciprocal rank (MRR) for the unambiguous case.

5.4.2 Results

Table 5.2 shows values of P@1 and P@5 obtained for unambiguous queries. We observe relatively higher precision values for **NA** and **AA**, which rely on the advanced approach to estimate $P(\,[tb,\, te] \mid d\,)$. Both achieve similar performance, indicating that our advanced method to estimate $P(\,d|q\,)$, taking into account query likelihoods, is not effective. This is substantiated by the performance of **NN** and **AN** – while the latter uses the advanced method to estimate $P(\,d \mid q\,)$, its precision values are as low as those obtained by the completely naïve **NN**. It can also be seen that methods' performance varies with temporal granularity, peaking at month granularity. Finally, we observe that considering more pseudo-relevant documents only pays off to a point – for none of the methods performance increases consistently as we go beyond $k = 50$.

Results for ambiguous queries are shown in Table 5.3. All four methods consistently achieve higher values of P@1 and P@5 than for the unambiguous case. Comparing **NN** and **AN**, we again observe that the advanced method of estimating $P(\,d|q\,)$ is not very effective. In contrast, we see good improvements for **NA** and **AA**, indicating that the more advanced handling of temporal expressions pays off. For ambiguous queries, as a difference from the unambiguous case, we observe that all methods achieve their best performance for year granularity. However, again we do not see consistent improvements as more pseudo-relevant documents are considered for larger choices of k.

	Day						**Month**						**Year**					
P@K	**P@1**			**P@5**			**P@1**			**P@5**			**P@1**			**P@5**		
k	**25**	**50**	**100**	**25**	**50**	**100**	**25**	**50**	**100**	**25**	**50**	**100**	**25**	**50**	**100**	**25**	**50**	**100**
NN	.03	.04	.04	.02	.03	.03	.06	.06	.03	.06	.08	.01	.03	.04	.04	.02	.03	.03
AN	.03	.03	.04	.02	.03	.03	.06	.05	.03	.06	.08	.01	.02	.01	.01	.01	.01	.01
NA	.07	.06	.09	.04	.04	.04	.18	.18	.18	.10	.10	.05	.14	.17	.10	.11	.11	.08
AA	.06	.06	.09	.04	.04	.04	.19	.17	.20	.09	.10	.07	.14	.17	.10	.11	.11	.08

Table 5.2: Temporally unambiguous queries

	Day						**Month**						**Year**					
P@K	**P@1**			**P@5**			**P@1**			**P@5**			**P@1**			**P@5**		
k	**25**	**50**	**100**	**25**	**50**	**100**	**25**	**50**	**100**	**25**	**50**	**100**	**25**	**50**	**100**	**25**	**50**	**100**
NN	.05	.00	.00	.01	.01	.01	.10	.10	.16	.05	.04	.05	.55	.55	.26	.31	.36	.33
AN	.05	.05	.05	.01	.01	.02	.10	.15	.16	.05	.05	.05	.60	.60	.32	.35	.34	.34
NA	.10	.10	.16	.05	.10	.10	.35	.50	.42	.25	.26	.31	.75	.75	.74	.59	.58	.54
AA	.10	.10	.16	.04	.10	.10	.35	.50	.42	.25	.26	.31	.75	.75	.74	.59	.58	.54

Table 5.3: Temporally ambiguous queries

Summary

Our experiments, using temporally unambiguous and temporally ambiguous queries as test cases, have shown that **NA** and **AA** perform similarly and are ahead of the other two configurations. Thus, the advanced method to handle temporal expressions and estimate $P(\,[tb,\, te] \mid d\,)$ is effective; the advanced method to estimate $P(\,d \mid q\,)$, on the other hand, has no effect.

5.5 Conclusion

We have proposed a novel approach to identify time intervals of interest for a given keyword query. Our approach is based on a generative model and we considered four possible instantiations of it. Experiments on temporally unambiguous queries and temporally ambiguous queries as test cases showed that there are effective instantiations of our approach – considering temporal expressions and their inherent uncertainty pays off; factoring in query likelihoods does not. As part of our future research, we plan to investigate (i) how users perceive the interestingness of the determined time intervals and (ii) how retrieval effectiveness is affected when using the determined time intervals in query expansion.

CHAPTER 6

TEMPORAL QUERY CLASSIFICATION AT DIFFERENT GRANULARITIES

• Dhruv Gupta and Klaus Berberich. Temporal Query Classification at Different Granularities. In SPIRE 2015, pages 156–164.

6.1 Introduction

The information need conveyed in a time-sensitive query can only be served properly if the temporal class (e.g., temporally ambiguous or unambiguous) associated with it can be determined. Determining the temporal class of a query is an important stepping stone to larger components in a time-sensitive information retrieval system. For instance, selection of retrieval model or deciding whether to diversify documents along time. Existing work in this direction has only relied on publication dates while ignoring temporal expressions in document content. Temporal expressions allow us to analyze events in web collections which may not have reliable publication dates associated with them. This alleviates the problem of being restricted to the time period covered by the publication dates of the document collection. Analyzing the temporal class based on temporal expressions is challenging as they are highly uncertain (e.g., early 1990's, during last century) and are present at multiple granularities (e.g., day, month, and year).

Earlier approaches [70, 125, 127] to determine the temporal class of a query have three major drawbacks. First, all approaches only use publication dates associated with documents. This may serve the purpose well for time-sensitive queries regarding current events covered in the news. But it may be inadequate for queries covering historic events. Second, prior approaches ignore the fact that certain events described in a query may be periodically recurring in nature (e.g., *summer olympics* or *nobel prize physics*) or they may be aperiodically recurring (e.g., *economic depression*). Third, temporal ambiguity is considered only at a single level of granularity. However, temporal ambiguity may vary according to granularity. Consider, as concrete example, the query *summer olympics tokyo athletics*. Relying only on publication dates this query would be incorrectly classified as temporally unambiguous; whereas it is temporally ambiguous at day granularity. Such an information need would be best served if these shortcomings can be overcome.

Approach. By addressing the aforementioned problems, we hypothesize we can improve upon the classification of time-sensitive queries containing: (i) historical events and entities; (ii) periodic events; and (iii) temporal ambiguity at a particular granularity. We build on our earlier work [93] which suggests interesting time intervals using temporal expressions. For classifying queries we identify multiple features from Bayesian analysis of the time intervals of interest for a given keyword query. We show the effectiveness of our proposed approach over prior work on a large testbed of time-sensitive queries.

Contributions described in this chapter are:

1. temporal class taxonomy taking into account multiple granularities and (a)periodicity of events;

2. determining time intervals as intents for temporally ambiguous queries;

3. effective features that outperform prior approaches; and

4. a large testbed of time-sensitive queries assimilated from previously available resources such as TREC time-sensitive queries [70], NTCIR Geo-Time queries [86] and other resources available on the Web, which is made publicly available for future research.

Organization. We next point to the sections that outline the solution to the problem of temporal query classification at different granularities. First prior approaches to the problem of temporal query classification are discussed in Section 6.2. We then cover the necessary preliminaries for explaining our approach in Section 6.3. We present our temporal class taxonomy in Section 6.4. We outline the approach for determining the temporal class given only a keyword query in Sections 6.5 and 6.6. Therein, we cover the Bayesian analysis utilized and the features derived from the analysis. In Section 6.7, we describe the datasets, query workloads, baseline, and our experimental setup. In Section 6.8, we conclude with a discussion of the results obtained.

6.2 Related Work

In this section, we describe prior approaches that address the task of temporal query classification. The work by Jones and Diaz [127] describes a taxonomy of classes for classification of time-sensitive queries. They discuss various features derived from probability distribution of document publication dates, e.g., temporal clarity, kurtosis, and auto-correlation. The Temporalia project described by Joho et al. [125] considers temporal query classification with a different taxonomy of temporal classes. The temporal classes they target are qualitatively labeled as past, recency, and future. This has two major caveats. First, the qualitative categories leave room for ambiguity in temporal intents. For example, for the query *nba playoffs last week* the temporal class can either be past or recent. Second, quantitatively no information can be discerned about the exact time intervals the temporal class refers to. Both these problems are addressed in our work. Detecting seasonality and periodicity associated with web queries has recently been explored by Kanhabua et al. [131]. They propose to use features acquired from web query logs. Additionally, akin to existing approaches, they rely on features derived from signal processing on time series of publication dates from an external document collection. These may not be adequate to detect the temporal class at different granularities, as shown in our experiments.

A large chunk of work has also been carried out with regard to incorporating the temporal class associated with the query during retrieval. In the approach presented by Berberich et al. [41] the authors leverage temporal expressions and document publication dates for improving search result quality for explicit temporal queries. Improving on this approach Kanhabua and Nørvåg [132] look at automatically suggesting years of interest to implicit temporal queries. To this end they utilize publication dates. Work by Dakka et al. [70] relies on publication dates of documents to improve the retrieval effectiveness by analyzing query-frequency histograms.

6.3 Preliminaries

We now introduce the notation used throughout the chapter and the approach for identifying time intervals of interest.

Notation. Consider a document collection $\mathcal{D}$. Each document $d \in \mathcal{D}$ consists of a bag of keywords d_{text} and a bag of temporal expressions d_{time}. We let $|d_{\text{text}}|$ and $|d_{\text{time}}|$ denote the cardinalities of these bags. We use the time model as described in Section 5.3.2.

Time Intervals of Interest to the given keyword query q are identified using the approach proposed in [93]. In a nutshell, with R as the set of pseudo-relevant documents, the approach assigns the probability:

$$P(\,[b,e] \mid q\,) = \sum_{d \in R} P(\,[b,e] \mid d\,)P(\,d \mid q\,),$$

to time interval $[b, e]$. The first probability is estimated as

$$P(\,[b,e] \mid d_{\text{time}}\,) = \frac{1}{|d_{\text{time}}|} \sum_{T \in d_{\text{time}}} \frac{\mathbb{1}([b,e] \in T)}{|T|},$$

following [41]. The second probability is estimated from the query likelihoods $P(q|d)$ under a unigram language model with Dirichlet smoothing, that is:

$$P(\,d \mid q\,) = \frac{P(\,q \mid d\,)}{\sum_{d' \in R} P(\,q \mid d'\,)} \; .$$

6.4 Temporal Class Taxonomy

We propose a new taxonomy taking into account additional classes for periodicity, aperiodicity, and multiple granularities (day, month, and year). It builds on the existing taxonomy proposed by Jones and Diaz [127]. The taxonomy, depicted in Figure 6.2, is arrived at by taking into account the following observations.

Atemporal queries as per [127] are time-invariant in nature. Thus, an atemporal query at year granularity also implies that it is atemporal at finer level of granularity (day and month) and vice versa. Hence, we only need to ascertain if a query is atemporal at *any one* of the granularities. This is illustrated by the distribution of time intervals in Figure 6.3a.

Temporally unambiguous queries are those with a unique time interval of interest associated with them. If a given query is identified to be unambiguous at day level of granularity then it will also be unambiguous at any coarser level of granularity (month & year). For instance, an unambiguous query at day level `concorde crash` is also unambiguous at year level. However, this does *not* imply that a unambiguous query at year level may necessarily be unambiguous at a month or day level. This is described in Figure 6.3b.

Temporally ambiguous queries are those which may have multiple time intervals of interest associated with them. Ambiguity associated with a query may lie at different granularities as illustrated in Figure 6.3c, 6.3d, and 6.3e. A temporally ambiguous query at a finer granularity may be unambiguous at coarser granularity. However, we make the distinction that a query ambiguous at *any* granularity be deemed temporally ambiguous at that level of granularity. For example the query `summer olympics 2000 rowing` is temporally ambiguous at day level granularity. Another aspect that we investigate is the periodicity and aperiodicity as a temporal intent. For example the query `summer olympics` should be classified as a periodic temporally ambiguous query. In this work, we focus only on periodicity at the year level.

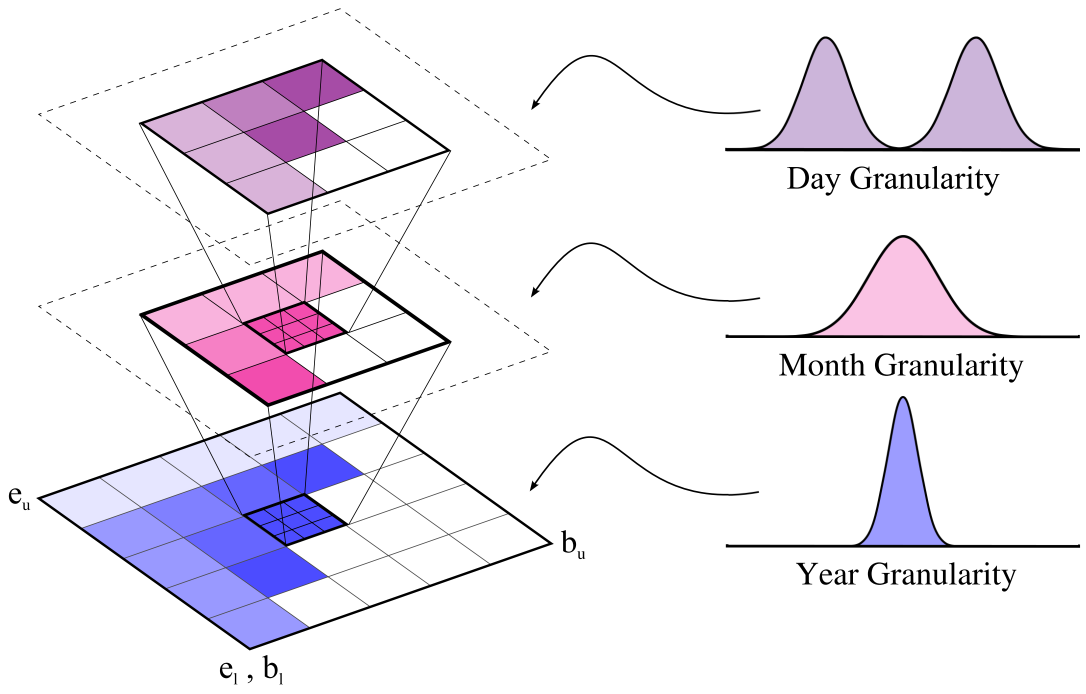


Figure 6.1: Distribution of time intervals at multiple granularities

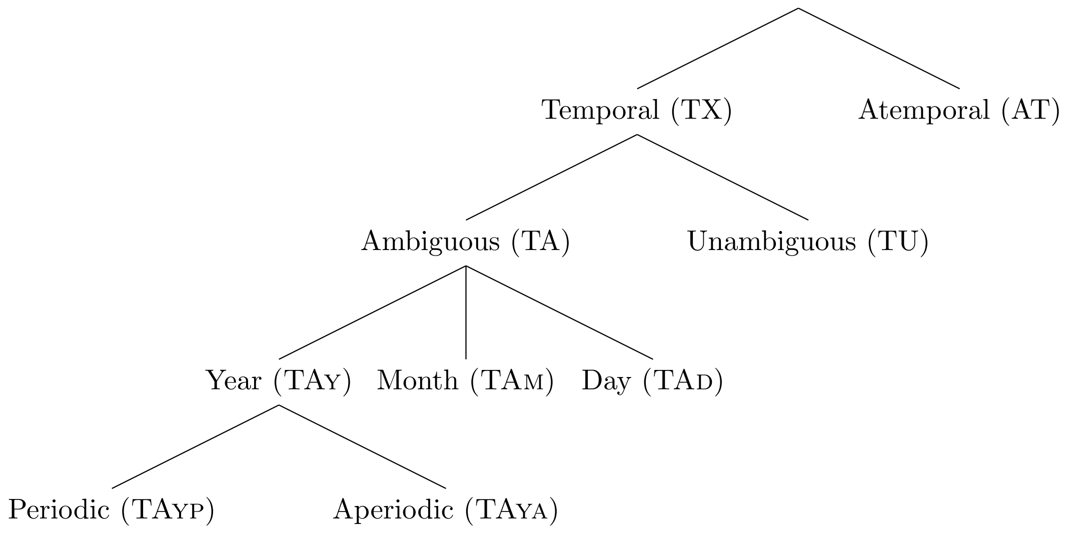
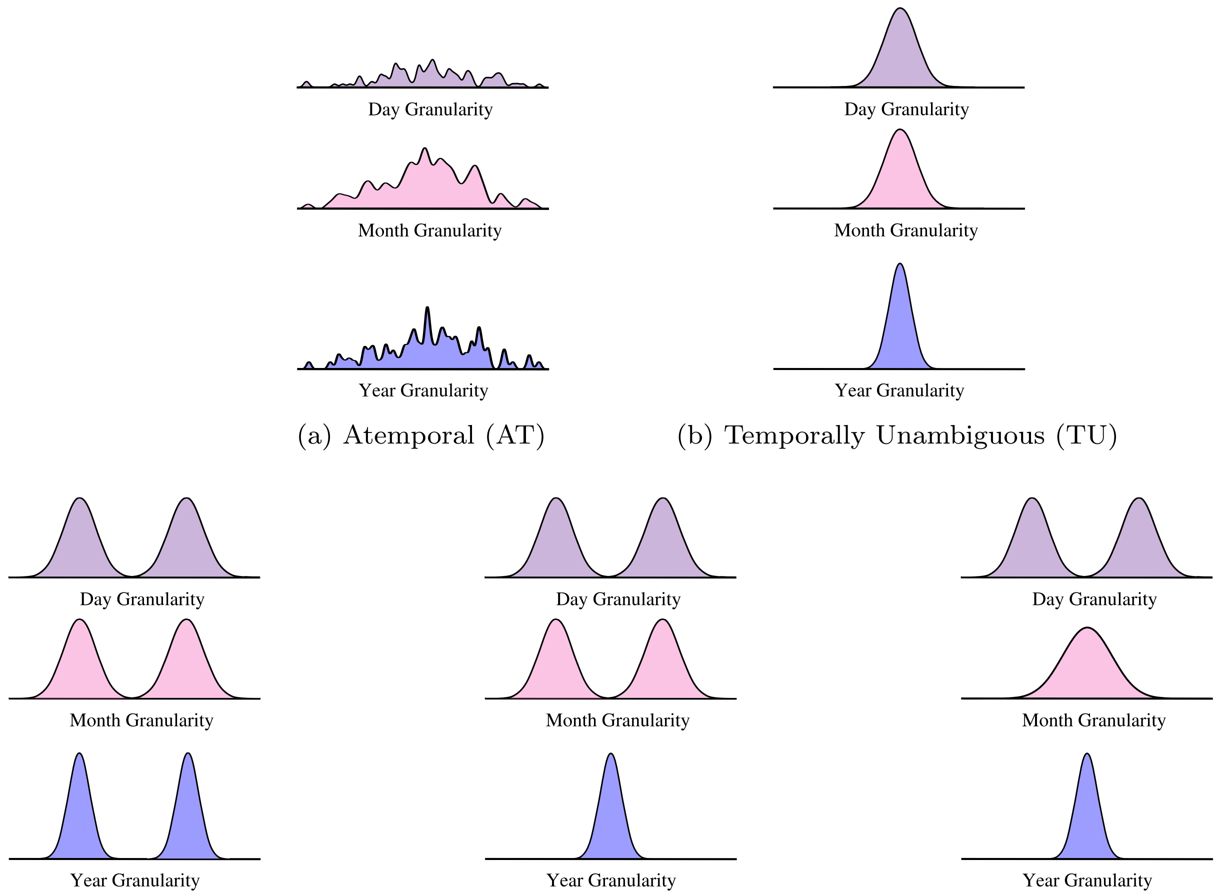


Figure 6.2: Temporal intent taxonomy with (a)periodicity & multiple granularity



(a) Atemporal (AT)

(b) Temporally Unambiguous (TU)

(c) Temporally Ambiguous at Year Granularity (TAy)

(d) Temporally Ambiguous at Month Granularity (TAm)

(e) Temporally Ambiguous at Day Granularity (TAd)

Figure 6.3: Distribution of time intervals at multiple granularities for different intents

6.5 Bayesian Analysis

In order to temporally profile a keyword query q we first obtain the time intervals of interest at all three temporal granularities and the associated probabilities of relevance $P([b,e]|q)$ (see Figure 6.1). We only consider time intervals of size equal to granularity under consideration (e.g., for year granularity $[b,e]$ spans one year). We further smooth $P([b,e]|q)$ with time intervals from the entire document collection $\mathcal{D}$ to obtain consistent analysis for all queries,

$$\hat{P}([b,e]|q) = \lambda \cdot P([b,e]|q) + (1-\lambda) \cdot P([b,e]|\mathcal{D}),$$

where,

$$P([b,e]|\mathcal{D}) = \frac{1}{|\mathcal{D}_{\text{time}}|} \sum_{T \in \mathcal{D}_{\text{time}}} \frac{\mathbb{1}([b,e] \in T)}{|T|}.$$

Detecting Multiple Modes. The generated distribution $\hat{P}([b,e]|q)$ is next analyzed for multi-modality. For this we utilize a *Bayesian Mixture Model* fitted using *reversible jump Markov Chain Monte Carlo* (MCMC) procedure outlined by Xu et al. [204]. The approach fits an unknown probability distribution by approximating it as mixture of Gaussian distributions. With this approach we have the advantage of performing both model selection and model fitting at the same time. That is, with this approach the number of components k for the mixture model is determined automatically. The mixture model is described as follows:

$$\hat{P}([b,e]|q) = \sum_{i=1}^{k} w_i \cdot \mathcal{N}(\mu_i, \sigma_i),$$

such that $\sum_i^k w_i = 1$; μ_i and σ_i characterize the mean and standard deviation of the normal distribution $\mathcal{N}(\mu_i, \sigma_i)$. To assess confidence of our hypothesis whether $\hat{P}([b,e]|q)$ is multi-modal, we take Bayes factor as an objective. Bayes factor is the ratio of the posterior to prior odds. If the Bayes factor exceeds 100, we consider the hypothesis, that the probability distribution under observation has multiple modes, correct. The time intervals with the means μ_i of the components of the mixture model are the temporal categories ($S^i_{[b,e]}$) of q:

$$S = \langle S^1_{[b,e]}, S^2_{[b,e]}, \ldots, S^k_{[b,e]} \rangle.$$

The temporal categories for q can further be utilized for diversifying search results along the temporal dimension. This is discussed in detail in Chapter 7.

6.6 Feature Design

After having determined the number of modes and the temporal categories from $\hat{P}([b,e]|q)$, we need to identify the temporal class it belongs to. This is done by encoding features derived from the Bayesian mixture model. The features are: (i) modality, (ii) fuzzy feature, and (iii) p-value of randomness test. Subsequently we train a decision tree (CART algorithm [49]), a supervised machine learning algorithm, on the identified features. In this section, we discuss the features and the motivation behind them.

Modality feature describes the number of modes identified by the Bayesian mixture model. The intuition is if $\hat{P}([b,e]|q)$ is unimodal ($|S| = 1$), then the temporal intent should be temporally unambiguous. If the distribution $\hat{P}([b,e]|q)$ is multi-modal ($|S| > 1$), then it should be temporally ambiguous.

Fuzzy Feature. To analyze the temporally ambiguous query for periodicity we use the concept of fuzzy numbers. Fuzzy logic is used here to account for outlier cases in periodic events, e.g., in case of `summer olympics` anomalous years would be [1936, 1936] and [1948, 1948]. Specifically, we capture the membership value of the time lags between the time intervals associated with different modes against a fuzzy number around the mean of the time lags ($\hat{\Phi}$). We explain the computational steps next.

We first identify the time lags between the ordered set of temporal categories:

$$\Phi^i_{[b,e]} = \langle t | t \in S^{i+1}_{[b,e]} - S^i_{[b,e]} \rangle,$$

with,

$$\hat{\Phi} = \frac{\sum_{i=1}^{n} \Phi^i_{[b,e]}}{n}.$$

Difference between intervals is calculated by subtracting corresponding b and e in each $S^i_{[b,e]}$. Next we construct a triangular fuzzy number with $\hat{\Phi}$ whose membership function is given by:

$$\mu(x) = \begin{cases} \frac{1}{1+x^2} & \text{if } x \neq \hat{\Phi} \\ 1 & \text{if } x = \hat{\Phi} \end{cases}$$

The motivation is: if $\mu(x) \neq 0 \ \forall x \in \Phi$, then issued query is a periodic query with period approximately equal to that of $\hat{\Phi}$. Otherwise if $\exists x \in \Phi$ for which $\mu(x) = 0$ then query could potentially be aperiodic.

Randomness Test. For atemporal queries we check $\hat{P}([b,e]|q)$ for randomness. For this, we perform a two-tailed runs up and down test for randomness [201] on time lags. We next note the p-value of this test as a feature. This feature captures if the time lags are randomly generated or not. For a given query, we construct the feature vector at day, month, and year granularity. The feature data then used for classification via the decision tree.

6.7 Evaluation

6.7.1 Document Collection and Queries

The document collection and the temporal annotations have been covered in detail in Section 5.4.1. In brief, the document collection used was The New York Times Annotated corpus [18]. The temporal annotations for it were obtained from the authors of [41], where they used TARSQI [200].

Queries. The challenging aspect of evaluating our approach was compiling a list of queries for temporally ambiguous class at different granularities. To this end, we use various previously published resources [93], TREC time-sensitive queries [70], NTCIR Geo-Time queries [86, 129], and also manually compiled some of them from the Web. We next describe the query sets that we have used for each temporal class. Table 6.1 summarizes the testbed. This testbed is publicly available at the following URL:

`http://resources.mpi-inf.mpg.de/dhgupta/data/spire2015` .

Temporally Unambiguous Queries were constructed with keywords describing historical events. Sources used were [70, 86, 93, 129]. Example queries that belong to this query set are: *`american civil rights activist rosa parks died`*, and *`concorde crash`*.

Temporally Ambiguous Queries were constructed for three different subsets at year, month and day granularity. For *year granularity* we accumulated international award events in various domains such as Sports, Science, Arts. Lists of such events is available in Wikipedia [1,2,3] pages. Since, these events are periodic in nature, it also serves us the purpose of periodic queries. Example queries in this set would be: *`us presidential elections`* and *`nobel prize in literature`*. *Temporally Ambiguous Queries at Month Granularity* considered major sports events in the United States (e.g., NBA Playoffs) with year of the season, e.g., *`nba playoffs 1990`*. *Temporally Ambiguous Queries at Day Granularity* were various sporting event competitions at a specific Summer Olympic games. Since each event competition spans multiple days but occur within the same month of the year. An example query would be, *`summer olympics 1992 archery`*. *Aperiodic Temporally Ambiguous Queries* were constructed via keywords that were either broad category of accidents or natural calamity (e.g., *`mercy killings`*, or *`earthquakes united states of america`*). Reliable sources for these queries were from related work [70, 86, 93, 127, 129]. Since events concerning known figures tend generally not be periodic in nature (e.g., *`abraham lincoln`*), we also considered several famous personalities as keyword queries in this category. A reliable source was obtained from Biography Online [4] website.

[1]http://www.wikipedia.org/wiki/List_of_prizes,_medals_and_awards
[2]http://www.en.wikipedia.org/wiki/List_of_multi-sport_events
[3]http://www.en.wikipedia.org/wiki/List_of_literary_awards
[4]http://www.biographyonline.net/people/famous-100.html

Set Id				Description	Size
TX	TA	TAY	TAYP	Periodic and ambiguous at year	113
			TAYA	Aperiodic and ambiguous at year	118
		TAM		Ambiguous at month	64
		TAD		Ambiguous at day	74
	TU			Unambiguous	142
AT				Atemporal	154

Table 6.1: Query set sizes for our evaluation setup.

Atemporal Queries, consist of general vocabulary words with no temporal significance (e.g., *apple*, *sardine*, and *guitar*). An English dictionary listing common food items and musical instruments was used for this purpose.

6.7.2 Setup

We discus various aspects related to the experimental setup next.

Baseline. We use the approach proposed by Jones and Diaz [127] as a baseline. As mentioned in [127] we selected the best-performing temporal features to build the baseline classifier. The temporal features considered by them were: first order autocorrelation, kurtosis, and features derived from a burst model. We consider these features at year level granularity for time intervals of interest. Since, we are considering time intervals of interest generated by the approach by Gupta and Berberich [93]; we take into account temporal expressions and publication dates at year granularity. This mimics the effect of the frequency counting of publication dates by Jones and Diaz but additionally tests the effectiveness of the features proposed by them and our proposed approach.

Parameters. For identifying time intervals of interest we considered top-50 ($|R| = 50$) pseudo-relevant documents. The mixing parameter for smoothing the distribution was set as $\lambda = 0.70$. For identifying the modality of the distribution we performed reversible jump MCMC procedure with 2,200 iterations with 200 initial burn-in iterations. The number of queries are mentioned in Table 6.1.

Implementation. All methods for feature extraction were implemented in *R*, a statistical programming language. Procedure for reversible MCMC sampling was obtained from [204] also in *R*. The decision tree classifier based on the CART algorithm was utilized from the *R* package *rpart* [197]. The generative model for time intervals of interest was programmed in Java.

Measures. For the classification task we report the standard measures for comparing performances – precision, recall, and F_1. In order to accurately gauge the performance we also report the confusion matrix for our classifier. Statistical significance of our results is reported with the p-value calculated using McNemar's test. We also show an unweighted κ statistic for the classifiers. The κ statistic measures the agreement between the observed accuracy to the expected accuracy by chance. Higher value of κ indicates better discrimination between different classes.

Statistics by Class						
	Precision		Recall		F_1	
Class	**B**	**A**	**B**	**A**	**B**	**A**
TX	0.81	**0.92**	0.79	**0.92**	0.80	**0.92**
├ TA	0.70	**0.87**	0.64	**0.71**	0.67	**0.78**
│ ├ TAY	0.45	**0.80**	0.34	**0.51**	0.39	**0.62**
│ │ ├ TAYP	0.29	**1.00**	0.26	**0.67**	0.27	**0.80**
│ │ └ TAYA	0.32	**0.55**	0.20	**0.33**	0.24	**0.41**
│ ├ TAM	0.24	**0.92**	0.50	**0.71**	0.32	**0.80**
│ └ TAD	0.36	**0.80**	0.22	**0.91**	0.28	**0.85**
└ TU	0.26	**0.39**	0.33	**0.64**	0.29	**0.48**
AT	0.38	**0.76**	0.41	**0.79**	0.39	**0.78**
Macroaverage	0.31	**0.74**	0.32	**0.67**	0.30	**0.69**
p-value	4.5×10^{-2}	2.2×10^{-16}				
κ-value	0.16	**0.62**				

Table 6.2: Statistics by class for decision trees: baseline (B) & proposed approach (A).

6.7.3 Results

Below we report the results for each temporal class. Training and test set were constructed by sampling without replacement. Train and test set split was 80% : 20% percent of the combined query workload (665 queries).

For the *temporally ambiguous* class we can classify very accurately at all levels of granularity. For the *atemporal* case we can also discern the class with high precision. However, it is relatively difficult to identify *temporally unambiguous* queries. Another class that is hard to detect is *aperiodic*. Compared to the baseline our approach performs better in all categories.

Predicted Class \ True Class	TAYP	TAYA	TAM	TAD	TU	AT
TAYP	**6**	4	0	2	5	4
TAYA	2	**6**	1	5	0	5
TAM	4	7	**6**	1	3	4
TAD	3	2	0	**4**	1	1
TU	6	4	1	1	**6**	5
AT	2	7	4	4	3	**13**

(a) Baseline (B)

Predicted Class \ True Class	TAYP	TAYA	TAM	TAD	TU	AT
TAYP	**14**	0	0	0	0	0
TAYA	0	**6**	0	0	5	0
TAM	0	0	**12**	1	0	0
TAD	1	1	0	**20**	0	3
TU	5	10	2	1	**14**	4
AT	1	1	3	0	3	**26**

(b) Proposed approach (A)

Figure 6.4: Confusion matrix for decision tree.

6.7.4 Failure Analysis

In this section, we give some example queries for the difficult classes for which our approach did not perform as compared to other classes.

The first class in which our proposed method does not perform well is *temporally unambiguous*. One reason that we anticipate for this is that since we consider pseudo-relevant documents it is inevitable to not consider other related events, which act as noise, for the keyword query in the probability distribution. Some misclassified example queries are : ***chernobyl soviet union*** and ***president nixon associated press orland***. The query ***chernobyl soviet union*** was wrongly labeled as aperiodic. This could potentially be due to discussion of aftermath of the event in the pseudo-relevant documents. The query ***president nixon associated press orland*** was misclassified as atemporal.

The second class which was hard to classify was *aperiodic*. Most of the queries here are misclassified as *unambiguous* – this can be attributed to very specific event in the corpus associated with the entity that comprises most of the queries in this category. Misclassified examples from this category are ***george bush jnr***, ***madrid bombing***, ***muhammad ali***, and ***ronald reagan***.

6.8 Conclusion

We have proposed how to solve the problem of temporal query classification at multiple levels of granularity. Additionally, we can predict the recurrence of events with very high accuracy. Our approach relies on inspecting both content temporal expressions as well as publication dates of pseudo-relevant documents given a keyword query. Our approach considers features based on Bayesian analysis of the time intervals of interest. Experiments clearly indicate that features identified by us are able to predict the temporal intent for ambiguous queries really well. In contrast, for *unambiguous* and *aperiodic* queries it is difficult to classify the intent by looking at the pseudo relevant documents. Overall, our classifier achieves the target of temporal intent classification with good accuracy.

CHAPTER 7

DIVERSIFYING SEARCH RESULTS USING TIME

• Dhruv Gupta and Klaus Berberich. Diversifying Search Results Using Time. In ECIR 2016, pages 789–795.

7.1 Introduction

Large born-digital document collections are a treasure trove of historical knowledge. Searching these large longitudinal document collections is only possible if we take into account the temporal dimension to organize them. In this chapter, we present a method for diversifying search results using temporal expressions in document contents. Our objective is to specifically address the information need underlying *history-oriented* queries; we define them to be keyword queries describing a historical event or entity. An ideal list of search results for such queries should constitute a *timeline* of the event or portray the *biography* of the entity. The approach described will prove to be useful for scholars in history and humanities who often search large text collections for *history-oriented* queries without knowing relevant dates for them apriori.

With growing amounts of information on the Web, modern retrieval system focus more on recently published documents by using their creation time or publication dates. However, little attention is given to the temporal expressions in document contents which can help us uncover a trove of historical knowledge. This is particularly challenging as temporal expressions may occur implicitly or explicitly. For example, "during the last three years of his presidency" [16]; the temporal expression last three years is implicit and should be resolved and normalized. Temporal expressions are usually mentioned in a *relative* sense and are highly uncertain in nature. As an example consider, early 1990s; here the exact time interval conveyed by the temporal expression 1990s is not clearly demarcated. Finally, temporal expressions can be present at different granularities of time. Consider the example, " On July 25, 2000, Bush . . . at the 2000 Republican National Convention " [16]; here we have 2000 present at year level of granularity while at day level of granularity we have July 25, 2000. Therefore, incorporating statistics about such temporal expressions in a search diversification method can be highly complex.

No work, to the best of our knowledge, has addressed the problem of diversifying search results using temporal expressions in document contents. Prior approaches in the direction of diversifying documents along time have relied largely on publication dates of documents. However a document's publication date may not necessarily be the time that the text refers to. It is quite common to have articles that contain a historical perspective of a past event from the current time. Hence, the use of publication dates is clearly insufficient for history-oriented queries.

In this chapter, we propose a probabilistic framework to diversify search results using temporal expressions (e.g., 1990s) from their contents. First, we identify time intervals of interest to a given keyword query, using our earlier work (see Chapter 5), which extracts them from pseudo-relevant documents. Having identified time intervals of interest (e.g., [2000,2004] for the keyword query *george w. bush*), we use them as aspects for diversification. More precisely, we adapt a well-known diversification method [25] to determine a search result that consists of relevant documents which cover all of the identified time intervals of interest.

Evaluation of historical text can be highly subjective and biased in nature. To overcome this challenge, we view the evaluation of our approach from a statistical perspective and take into account an objective evaluation for automatic summarization to measure the effectiveness of our methods. We create a large history-oriented query collection consisting of long-lasting wars, important events, and eminent personalities from reliable encyclopedic resources and from prior available research. As a ground truth we utilize articles from *Wikipedia* [1] concerning the queries. We evaluate our methods on two large document collections, the New York Times Annotated corpus and

[1] `https://en.wikipedia.org/`

the Living Knowledge corpus. Our approach is thus tested on two different types of textual data. One being highly authoritative in nature, in the form of news articles. Another being authored by real-world users, in the form of web documents. Our results show that using our method of diversifying search results using time, we can present documents that serve the information need in a history-oriented query very well.

Outline. The remainder of the chapter is structured as follows. Section 7.2 covers related work in the context of temporal information retrieval and text analytics. Section 7.3 presents our adapted probabilistic framework for diversifying search results along time. Section 7.4 covers in detail our evaluation framework. We discuss the results and some anecdotal results in Section 7.5. Finally, we end with concluding remarks in Section 7.6.

7.2 Related Work

Our research bridges a gap between two very important research themes: temporal information retrieval and temporal text analytics. Temporal IR centric methods have largely avoided leveraging complex temporal expressions in favor of document publication dates. Temporal text analytical methods on the other hand have largely relied on these temporal expressions for mining time-sensitive facts. We discuss some of these works next.

Temporal Information Retrieval. Diversifying search results using time was explored in [40]. In their preliminary study the authors limited themselves to using document publications dates. However they posed the open problem of diversifying search results using temporal expressions in document contents and the challenging problem of evaluation. Both these aspects have been adequately addressed in our work. More recently, Nguyen and Kanhabua [164] diversify search results based on dynamic latent topics. The authors study how the subtopics for a multi-faceted query change with time. For this they utilize a time-stamped document collection and an external query log. However for the temporal analysis they limit themselves to document publication dates. A recent survey of temporal information retrieval by Campos et al. [53] also highlights the lack of any research that address the challenges of utilizing temporal expressions in document contents for search result diversification along time.

Temporal Text Analytics. Using text analytics and temporal expressions, Yeung and Jatowt [207] study how the past is remembered. They study varying trends of topics identified via Latent Dirichlet Allocation along time in a document collection. Special emphasis has been laid on predictions of future events. In the seminal work by Baeza-Yates [33], the author explores how to model a future retrieval system that would take in to account temporal expressions in a document body. More recently, Jatowt and Yeung [124] explore this research direction by proposing a model-based clustering algorithm for extracting representative summaries for future events from the Google news archive.

7.3 Method

We now describe the probabilistic framework to diversify search results using temporal expressions mentioned in their contents. The probabilistic framework consists of three key components. The first key component is the representation of time that incorporates temporal uncertainty. The second key component, described elsewhere, is that of generating time intervals of interest for the given keyword query. The final key component is the objective function for determining the maximal subset of documents that covers the time intervals of interest.

Time Model

We consider a document collection $\mathcal{D}$. Each document $d \in \mathcal{D}$ consists of a multiset of keywords d_{text} drawn from vocabulary $\mathcal{V}$ and a multiset of temporal expressions d_{time}. Cardinalities of the multisets are denoted by $|d_{text}|$ and $|d_{time}|$. To model temporal expressions such as 1990s where the begin and end of the interval can not be identified, we utilize the work by Berberich et al. [41]. They allow for this uncertainty in the time interval by associating lower and upper bounds on begin and end. Thus, a temporal expression T is represented by a four-tuple: $\langle b_l, b_u, e_l, e_u \rangle$ where time interval $[b, e]$ has its begin bounded as $b_l \leq b \leq b_u$ and its end bounded as $e_l \leq e \leq e_u$. The temporal expression 1990s is thus represented as $\langle 1990, 1999, 1990, 1999 \rangle$. More concretely, elements of temporal expression T are from time domain $\mathcal{T}$ and intervals from $\mathcal{T} \times \mathcal{T}$. The number of such time intervals that can be generated is given by $|T|$.

Time Intervals of Interest

Time interval of interest to the given keyword query q_{text} are identified using our earlier work [93]. A time interval $[b, e]$ is deemed *interesting* if its referred frequently by highly relevant documents of the given keyword query. This intuition is modeled as a two-step generative model. Given, a set of pseudo-relevant documents R, a time interval $[b, e]$ is deemed interesting with probability:

$$P(\,[b,e] \mid q_{text}\,) = \sum_{d \in R} P(\,[b,e] \mid d_{time}\,) P(\,d_{text} \mid q_{text}\,).$$

To diversify search results, we keep all the time intervals generated with their probabilities in a set q_{time}.

Temporal Diversification

Our aim is to present documents that cover a variety of historical aspects underlying a history-oriented query. To this end we use the identified interesting time intervals as explicit temporal aspects that need to be satisfied by the documents. The diversified set of documents must thus try to cover these time intervals in proportion to the frequency of their occurrence.

To diversify search results we adapt the approach proposed by Agrawal et al. [25]. Formally, the objective is to maximize the probability that the user sees at least one result relevant to her time interval of interest. We thus aim to determine a query result $S \subseteq R$ that maximizes

$$\sum_{[b,e] \in q_{time}} \left(P(\,[b,e] \,|\, q_{text}\,) \left(1 - \prod_{d \in S} \left(1 - P(\,q_{text} \,|\, d_{text}\,) P(\,[b,e] \,|\, d_{time}) \right) \right) \right).$$

The probability $P(\,[b,e] \,|\, q_{text}\,)$ is estimated as described above and reflects the salience of time interval $[b,e]$ for the given query. We make an independence assumption and estimate the probability that document d is relevant and covers the time interval $[b,e]$ as $P(\,q_{text} \,|\, d_{text}\,)\, P(\,[b,e] \,|\, d_{time})$. To determine the diversified result set S, we use the greedy algorithm described in [25], which is known to give a $(1 - \frac{1}{e})$ approximation guarantee.

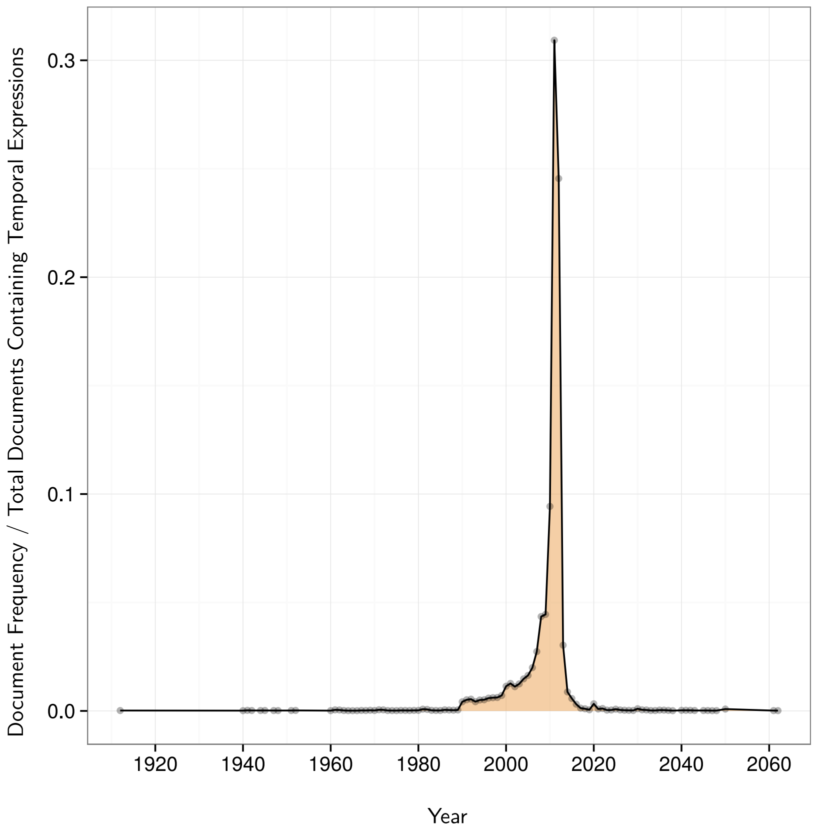
7.4 Evaluation

We next describe the setup of our experimental evaluation.

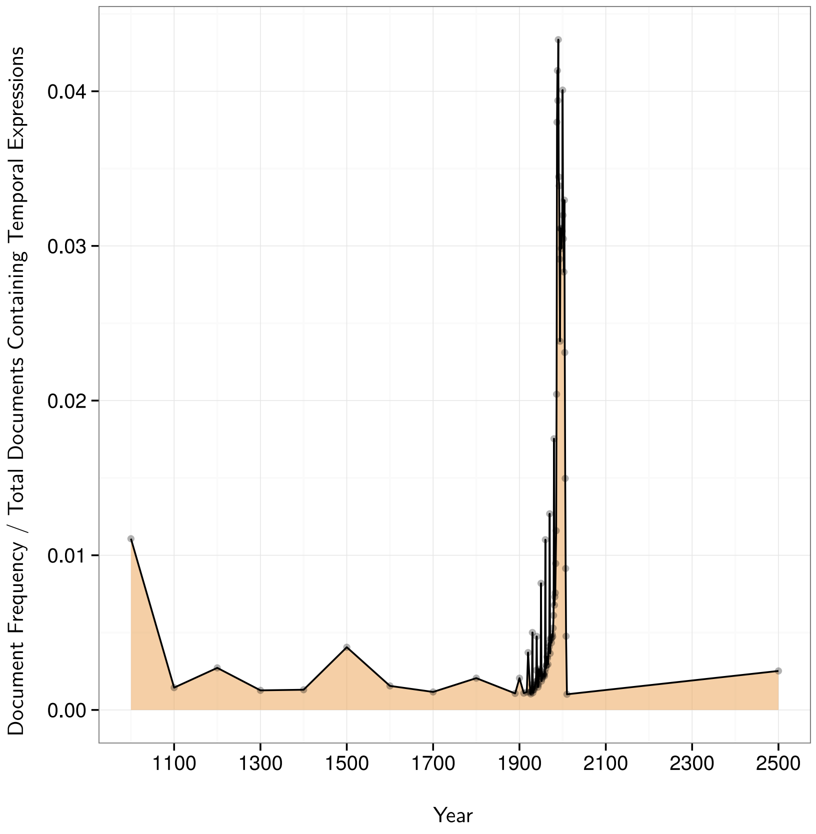
7.4.1 Document Collections

We used two document collections, one from a news archive and one from a web archive. The Living Knowledge [15] corpus is a collection of news and blogs on the Web amounting to approximately 3.8 million documents [125]. The documents are provided with annotations for temporal expressions as well as named entities. The New York Times (NYT) Annotated [18] corpus is a collection of news articles published in *The New York Times*. It reports articles from 1987 to 2007 and consists of around 2 million news articles. The temporal annotations for it were done via SUTime [58]. Both explicit and implicit temporal expressions were annotated, resolved, and normalized using SUTime.

Temporal Analysis. We did a simple analysis of temporal expressions in both document collections. This involved computing document frequency of temporal expressions at year granularity across the collections. The plots are shown in Figure 7.1. For the Living Knowledge corpus the kurtosis of the distribution of document frequency ordered by time is 421.4 and skewness of 19.9. For the NYT Annotated corpus the kurtosis is 6.8 and skewness is 2.8. This shows that the Living Knowledge corpus has a highly skewed nature of distribution of temporal expressions. Also take note of the time scale on the $x-$axis of both plots (7.1a, 7.1b). The NYT Annotated corpus has a wider temporal coverage in contrast to the Living Knowledge corpus. Given these observations we can conclude that the Living Knowledge corpus is *not* a true longitudinal corpus as the time-period the corpus covers is limited and distorted. These aspects affect any probabilistic analysis performed on the Living Knowledge corpus. The resulting effects show up in our results; which we discuss later.



(a) Living Knowledge corpus.



(b) New York Times Annotated corpus.

Figure 7.1: Distribution of temporal expressions at year granularity by document frequency.

Indexing. The document collections were preprocessed and subsequently indexed using the ElasticSearch software [4]. As an ad-hoc retrieval baseline and for retrieval of pseudo-relevant set of documents we utilized the state-of-the-art Okapi BM25 retrieval model implemented in ElasticSearch.

7.4.2 Collecting History-Oriented Queries

In order to evaluate the usefulness of our method for scholars in history, we need to find keyword queries that are highly ambiguous in the temporal domain. That is multiple interesting time intervals are associated with the queries. For this purpose we considered three categories of history-oriented queries: long-lasting wars, recurring events, and famous personalities. For constructing the queries we utilized reliable sources on the Web and data presented in prior research articles [93, 153]. We describe the details next.

Queries for long-lasting wars were constructed from the *WikiWars* corpus [153]. The corpus was created for the purpose of temporal information extraction. The keywords for the wars are given in Table 7.1a. For ambiguous important events we utilized the set of ambiguous queries used in our earlier work [93]. The queries used are listed in Table 7.1b. For famous personalities we utilized a list of most influential people available on the USA Today [2] website. The names of these famous personalities were used based on the intuition that there would be important events associated with them at different points of time. The list of all the entities is given in in Table 7.1c.

The objective of our method is to present documents that depict the historical timeline or biography associated with the keyword query describing the event or the named entity. We thus treat the diversified set of documents as a *historical summary* of the query. In order to evaluate this diversified summary we obtain the corresponding *Wikipedia*[3] pages of the queries as ground truth summaries. The entire testbed of history-oriented queries along with their corresponding *Wikipedia* articles is made publicly available at the following URL:

`http://resources.mpi-inf.mpg.de/dhgupta/data/ecir2016/`.

7.4.3 Systems and Metrics

Baselines. We considered three baselines, in order of increasing sophistication. As a naïve baseline, we first consider the pseudo-relevant documents retrieved for the given keyword query. The next two baselines utilize a well known implicit diversification algorithm *maximum marginal relevance* (MMR) [55]. Formally it is defined as:

$$\underset{d \notin S}{\operatorname{argmax}} \left(\lambda \cdot \mathrm{sim}_1 \left(q, d\right) - (1 - \lambda) \cdot \max_{d' \in S} \mathrm{sim}_2 \left(d', d\right) \right).$$

[2] `http://usatoday30.usatoday.com/news/top25-influential.htm`
[3] `https://en.wikipedia.org/`

Americas	*american civil war* | *american revolution* | *mexican revolution*
Europe	*world war II* | *world war I* | *french revolution* | *punic wars* | *spanish civil war* | *russo-polish war* | *second italo abyssinian war*
Africa	*french algerian war* | *biafran nigerian civil war*
Asia	*vietnam war* | *korean war* | *iraq war* | *persian wars* | *chinese civil war* | *iran iraq war* | *russian civil war* | *french indochina war* | *russo-japanese car*

(a) Wars.

Sports	*commonwealth games* | *asian games* | *summer olympics* | *winter olympics* | *super bowl winners*
Music	*u2 album* | *nirvana album* | *beatles album* | *red hot chilli peppers album* | *michael jackson album*
Movies	*harry potter movie* | *oscar academy awards* | *lord of the ring movie*
Politics	*german federal elections* | *us presidential elections* | *australia federal elections*

(b) Events.

Business	*bill gates* | *sergey brin* | *larry page* | *howard schultz* | *sam walton*
Science	*stephen hawking* | *francis collins* | *craig venter*
Politics	*ronald reagan* | *mikhail gorbachev* | *george w. bush* | *deng xiaoping* | *nelson mandela* | *bill clinton* | *hillary clinton*
Arts	*j. k. rowling* | *oprah winfrey* | *russell simmons* | *bono*
Religion	*pope john paul II*
Sports	*lance armstrong* | *michael jordan*
Other	*ryan white* | *homer simpson* | *osama bin laden*

(c) Entities.

Table 7.1: History-oriented queries.

MMR was simulated with sim_1 using query likelihoods and sim_2 using cosine similarity between the term-frequency vectors for the documents. The second baseline considered MMR with $\lambda = 0.5$ giving equal importance to query likelihood and diversity. While the final baseline considered MMR with $\lambda = 0.0$ indicating complete diversity. To initialize set S, we use the first document from pseudo-relevant set of documents. For all methods the summary is constructed by concatenating all the top-k documents into one large document.

Parameters. There are two parameters to our system. The first one is the number of documents considered for generating time intervals of interest $|R|$. The second parameter is the number of documents considered for *historical summary* $|S|$. We consider the following settings of these parameters: $|R| \in \{100, 150, 200\}$ and $|S| \in \{5, 10\}$.

Metrics. We use the Rouge-N measure [208] (implemented in [146]) to evaluate the *historical summary* constituted by a diversified set of documents with respect to the ground truth. Rouge-N is a recall-oriented metric which reports the number of n-grams matches in the candidate summary S and the reference summary G with respect to the reference summary G:

$$recall = \frac{\sum_{g \in ngram(G)} \sum_{s \in ngram(S)} \mathbb{1}(s = g)}{\mid ngram(G) \mid},$$

where, $ngram(\cdot)$ returns n-grams for piece of text, $\mid ngrams(\cdot) \mid$ gives the total number n-grams, and $\mathbb{1}(\cdot)$ is an indicator function that tests the equivalence of n-grams. Note that n in $ngram$ is the length of the gram to be considered; we limit ourselves to $n \in \{1, 3\}$. The precision is calculated in a similar manner although with respect to the candidate summary S. To combine both measures, F_β is used:

$$F_\beta = \frac{\left(1 + \beta^2\right) \cdot recall \cdot precision}{recall + \beta^2 \cdot precision}$$

7.5 Results

Results are shown for three different categories of history-oriented queries per document collection. For each category of history-oriented query we show *recall*, *precision*, and $F_{\beta=1.0}$ scores for Rouge-1 and Rouge-3 metrics. $F_{\beta=1.0}$ is a balanced metric that gives equal weight to both precision and recall. All values are reported are percentages of the metrics and averaged over all the queries in a group. The results for the New York Times Annotated corpus are presented in Tables 7.2 and for the Living Knowledge corpus are shown in Table 7.3.

Given a history-oriented query, an ideal list of documents should either give a timeline overview of the event or portray the biography of the entity. Therefore all the documents that the system presents must *recall* as many facts as possible when compared to a ground truth summary. The *precision* as it is computed with respect to the system generated summary may vary as we increase the number of pseudo-relevant documents. Regardless of this we present the $F_{\beta=1.0}$ scores that give equal weight to both precision and recall.

For the New York Times Annotated corpus we can clearly see that our method Time-Diverse outperforms all three baselines by a large margin in recalling most important facts concerning the history-oriented queries. This shows that using retrieval method informed by temporal expressions presents documents that are *retrospectively relevant* for history-oriented queries. The slightly higher precision values for baseline system in all the findings above can be attributed to the fact that most of the baseline summaries tended to be of shorter length than the summaries produced by Time-Diverse method. When increasing the size of $|R|$ we notice that recall also increases for Time-Diverse as compared to the baselines. Since the increase in $|R|$ also implies an increase in the length of the summary, the precision also drops.

For the Living Knowledge corpus we see that our method performs better than the baselines when considering the $F_{\beta=1.0}$ scores. When considering recall it performs at par with MMR ($\lambda = 0.0$). While looking at the precision of the summaries our method Time-Diverse constantly outperforms the baselines; considering that the length of the summaries in this case tend to be uniform for all methods. As discussed in Section 7.4.1, Living Knowledge presents us with the challenge of having a skewed and biased distribution of temporal expressions. Even taking into account this factor; our method shows an overall improvement over the baselines. This again empirically shows that our method is highly sensitive to temporal expressions and aids in *temporal diversification*.

There is no clear correlation between a good summary and the number of top-k documents $|R|$ considered for generating time intervals of interest; in most cases though it seems increasing the size of pseudo-relevant set generation of time intervals hurts the performance of the diversification algorithm. Considering a larger number of documents that are presented to the user $|S|$ increases the performance; indicating $|S| = 10$ as an optimal value. Overall, the results conclusively show that using our diversification algorithm taking into account temporal expressions gives us a better retrospective overview of a history-oriented query.

7.5.1 History by Algorithms

Here we present anecdotal results for two example history-oriented queries. Queries considered were `george w. bush` and `economic depression`. The results shown in Figures 7.2 and 7.3 are from the New York Times Annotated corpus. Individual results are shown with their article headline and their contained temporal expressions. In addition we show the time intervals identified and used for diversification.

For the query `george w. bush` the identified time intervals include the time intervals [2000, 2000] and [2000, 2004] marking the year of his first election and his first presidential term. This is covered by documents $D_{1117027}$, $D_{1461580}$, and $D_{1255229}$ returned by our diversification method. The second term of his

	Category	Historical Wars						Historical Events						Historical Entity					
	Metric	R		P		$F_{\beta=1.0}$		R		P		$F_{\beta=1.0}$		R		P		$F_{\beta=1.0}$	
	Rouge-N	1	3	1	3	1	3	1	3	1	3	1	3	1	3	1	3	1	3
$\|R\| = 100$	Naïve	30.5	12.0	62.7	23.5	33.9	13.2	43.3	18.0	42.4	15.7	21.0	8.4	19.9	7.9	74.6	29.8	24.4	9.8
	MMR ($\lambda = 0.5$)	30.5	12.0	62.8	23.6	33.9	13.2	43.3	18.0	42.6	15.6	21.1	8.4	20.0	7.9	74.3	29.6	24.6	9.8
$\|S\| = 5$	MMR ($\lambda = 0.0$)	30.5	12.0	62.8	23.6	33.9	13.2	43.3	18.0	42.6	15.6	21.1	8.4	20.0	7.9	74.3	29.6	24.6	9.8
	Time-Diverse	46.4	17.5	55.7	21.1	41.0	15.5	56.7	22.0	35.9	13.0	26.3	9.9	35.3	13.4	67.0	25.3	34.5	13.1
$\|R\| = 100$	Naïve	48.0	18.4	51.0	18.9	39.2	15.0	57.6	22.9	33.4	12.0	23.1	8.7	35.4	13.6	67.4	26.7	34.4	13.5
	MMR ($\lambda = 0.5$)	48.4	18.5	50.6	18.8	39.2	15.0	57.5	22.9	33.4	11.9	23.1	8.7	35.8	13.7	67.2	26.8	34.7	13.6
$\|S\| = 10$	MMR ($\lambda = 0.0$)	48.4	18.5	50.6	18.8	39.2	15.0	57.5	22.9	33.4	11.9	23.1	8.7	35.8	13.7	67.2	26.8	34.7	13.6
	Time-Diverse	64.8	24.4	43.2	16.5	42.6	16.3	66.1	24.3	27.1	8.9	23.1	8.0	48.2	17.8	56.9	21.1	36.8	13.7
$\|R\| = 150$	Naïve	30.5	12.0	62.7	23.5	33.9	13.2	43.3	18.0	42.4	15.7	21.0	8.4	19.9	7.9	74.6	29.8	24.4	9.8
	MMR ($\lambda = 0.5$)	30.5	12.0	62.8	23.6	33.9	13.2	43.3	18.0	42.6	15.6	21.1	8.4	20.0	7.9	74.3	29.6	24.6	9.8
$\|S\| = 5$	MMR ($\lambda = 0.0$)	30.5	12.0	62.8	23.6	33.9	13.2	43.3	18.0	42.6	15.6	21.1	8.4	20.0	7.9	74.3	29.6	24.6	9.8
	Time-Diverse	48.2	18.6	55.1	21.1	42.0	16.2	58.1	22.6	33.4	12.2	25.7	9.6	38.0	14.1	65.3	23.9	36.7	13.7
$\|R\| = 150$	Naïve	48.0	18.4	51.0	18.9	39.2	15.0	57.6	22.9	33.4	12.0	23.1	8.7	35.4	13.6	67.4	26.7	34.4	13.5
	MMR ($\lambda = 0.5$)	48.5	18.6	50.7	18.8	39.3	15.1	57.5	22.9	33.4	11.9	23.1	8.7	35.7	13.7	67.3	26.8	34.7	13.7
$\|S\| = 10$	MMR ($\lambda = 0.0$)	48.5	18.6	50.7	18.8	39.3	15.1	57.5	22.9	33.4	11.9	23.1	8.7	35.7	13.7	67.3	26.8	34.7	13.7
	Time-Diverse	65.4	24.9	42.1	16.4	42.2	16.3	67.0	24.9	26.4	9.2	23.1	8.1	54.2	20.1	55.7	20.9	40.8	15.5
$\|R\| = 200$	Naïve	30.5	12.0	62.7	23.5	33.9	13.2	43.3	18.0	42.4	15.7	21.0	8.4	19.9	7.9	74.6	29.8	24.4	9.8
	MMR ($\lambda = 0.5$)	30.5	12.0	62.8	23.6	33.9	13.2	43.3	18.0	42.6	15.6	21.1	8.4	20.0	7.9	74.3	29.6	24.6	9.8
$\|S\| = 5$	MMR ($\lambda = 0.0$)	30.5	12.0	62.8	23.6	33.9	13.2	43.3	18.0	42.6	15.6	21.1	8.4	20.0	7.9	74.3	29.6	24.6	9.8
	Time-Diverse	51.7	20.0	53.2	20.3	43.7	16.8	59.4	23.0	34.8	12.7	27.7	10.4	39.6	15.2	64.6	23.8	37.6	14.5
$\|R\| = 200$	Naïve	48.0	18.4	51.0	18.9	39.2	15.0	57.6	22.9	33.4	12.0	23.1	8.7	35.4	13.6	67.4	26.7	34.4	13.5
	MMR ($\lambda = 0.5$)	48.5	18.6	50.7	18.8	39.3	15.1	57.5	22.9	33.4	11.9	23.1	8.7	35.7	13.7	67.3	26.8	34.7	13.7
$\|S\| = 10$	MMR ($\lambda = 0.0$)	48.5	18.6	50.7	18.8	39.3	15.1	57.5	22.9	33.4	11.9	23.1	8.7	35.7	13.7	67.3	26.8	34.7	13.7
	Time-Diverse	66.4	24.8	38.2	14.3	39.4	14.8	69.5	25.9	25.2	8.8	24.1	8.7	54.7	20.0	54.2	19.5	41.5	15.3

Table 7.2: Results for the New York Times Annotated corpus.

	Category	Historical Wars						Historical Events						Historical Entity					
	Metric	R		P		$F_{\beta=1.0}$		R		P		$F_{\beta=1.0}$		R		P		$F_{\beta=1.0}$	
	Rouge-N	1	3	1	3	1	3	1	3	1	3	1	3	1	3	1	3	1	3
$\|R\| = 100$	Naïve	14.5	1.7	62.0	16.8	18.4	2.3	21.5	6.4	53.2	16.4	22.4	6.5	4.6	1.2	80.2	37.0	7.6	2.0
	MMR ($\lambda = 0.5$)	14.5	1.7	62.0	16.6	18.4	2.4	21.3	6.4	52.6	15.9	22.3	6.6	4.6	1.2	79.0	34.6	7.7	2.0
$\|S\| = 5$	MMR ($\lambda = 0.0$)	28.3	1.8	49.7	6.8	23.4	1.7	37.5	6.7	38.5	5.2	24.9	3.6	14.1	1.5	66.5	9.3	20.2	2.1
	Time-Diverse	26.1	1.9	52.1	6.1	24.9	1.9	36.7	6.9	40.7	5.8	26.9	4.1	15.0	1.5	68.8	12.7	19.9	2.0
$\|R\| = 100$	Naïve	24.8	2.9	55.0	11.3	25.7	3.3	38.2	11.2	42.9	11.5	27.7	7.0	9.2	2.0	77.0	32.1	13.1	3.1
	MMR ($\lambda = 0.5$)	25.4	2.9	54.7	10.8	26.4	3.3	38.4	11.2	42.6	10.2	28.0	7.0	9.5	2.0	75.4	28.6	13.5	3.1
$\|S\| = 10$	MMR ($\lambda = 0.0$)	40.7	3.3	40.8	4.2	29.4	2.5	50.3	11.3	32.7	4.4	26.5	4.1	22.4	2.4	62.5	8.5	27.1	2.9
	Time-Diverse	39.2	3.4	43.0	4.8	30.6	2.7	50.6	11.7	34.1	4.8	29.4	4.7	22.5	2.5	64.6	10.5	25.8	3.0
$\|R\| = 150$	Naïve	14.8	1.7	60.7	14.3	18.8	2.4	21.5	6.4	53.2	16.4	22.4	6.5	4.3	1.1	81.4	39.2	7.2	2.0
	MMR ($\lambda = 0.5$)	14.8	1.7	60.6	14.2	18.8	2.4	21.3	6.4	52.6	15.9	22.3	6.6	4.4	1.1	80.4	36.9	7.4	2.0
$\|S\| = 5$	MMR ($\lambda = 0.0$)	31.5	2.3	46.5	4.5	26.2	1.9	40.3	7.1	37.3	4.7	24.0	3.3	15.1	1.5	63.2	9.1	19.7	2.1
	Time-Diverse	28.6	2.1	49.9	4.5	27.5	2.1	36.7	6.8	39.2	5.8	25.6	4.1	16.4	1.5	66.5	9.0	20.7	2.0
$\|R\| = 150$	Naïve	24.8	2.9	55.0	11.3	25.7	3.3	38.2	11.2	42.9	11.5	27.7	7.0	9.2	2.0	77.0	32.1	13.1	3.1
	MMR ($\lambda = 0.5$)	25.4	3.0	54.8	10.8	26.5	3.3	38.8	11.2	42.4	10.2	28.1	7.0	9.5	2.0	75.3	28.6	13.5	3.1
$\|S\| = 10$	MMR ($\lambda = 0.0$)	42.9	3.8	39.1	3.8	30.5	2.6	53.5	11.8	31.4	3.9	26.6	3.8	25.4	2.6	58.9	7.3	28.1	2.9
	Time-Diverse	41.5	3.6	41.2	3.7	32.3	2.7	50.0	11.7	33.2	4.8	28.1	4.7	25.1	2.6	61.3	7.8	28.0	2.9
$\|R\| = 200$	Naïve	14.8	1.7	60.7	14.3	18.8	2.4	21.5	6.4	53.2	16.4	22.4	6.5	4.3	1.1	81.4	39.2	7.2	2.0
	MMR ($\lambda = 0.5$)	14.8	1.7	60.4	13.9	18.9	2.4	21.3	6.4	52.6	15.9	22.3	6.6	4.4	1.1	80.4	36.9	7.4	2.0
$\|S\| = 5$	MMR ($\lambda = 0.0$)	31.5	2.3	45.3	4.1	26.1	1.9	38.5	6.8	36.9	4.9	23.3	3.2	18.2	1.5	61.3	7.3	21.7	1.8
	Time-Diverse	30.2	2.1	48.8	4.3	27.8	2.0	38.3	7.0	41.0	5.5	27.4	4.1	15.0	1.4	68.4	8.9	20.7	2.0
$\|R\| = 200$	Naïve	24.8	2.9	55.0	11.3	25.7	3.3	38.2	11.2	42.9	11.5	27.7	7.0	9.2	2.0	77.0	32.1	13.1	3.1
	MMR ($\lambda = 0.5$)	25.4	3.0	54.7	10.7	26.5	3.3	38.8	11.2	42.4	10.2	28.1	7.0	9.5	2.0	75.3	28.6	13.5	3.1
$\|S\| = 10$	MMR ($\lambda = 0.0$)	43.0	3.7	37.9	3.3	30.5	2.4	54.0	11.7	30.7	3.8	27.0	3.7	27.9	2.5	56.8	6.2	28.7	2.6
	Time-Diverse	42.3	3.6	40.1	3.6	31.0	2.6	53.2	12.1	33.3	4.5	29.4	4.4	25.4	2.6	61.6	7.1	29.5	3.0

Table 7.3: Results for the Living Knowledge corpus.

presidency is marked by the category [2004, 2004] and [2004, 2007] (corpus covers the time period 1987-2007) described by the document $D_{1610342}$. The last temporal category is [1992, 1992] in which George W. Bush worked as campaign advisor for his father's presidential campaign. The 1992 presidential campaign of his father is given in D_{537116}. This is a more temporally diverse set of documents as compared against the baseline set of documents centered around 2000.

For the query `economic depression` identified time intervals explicitly cover the various periods when a downturn occurred. The periods [1990, 1991] & [1990, 1995] represent the *early 90's depression* covered by documents D_{491246}, D_{113808}, & D_{741011} ; [1987, 1987] covers the *Black Monday* when stock markets crashed, the story is reported in D_{88390} and finally [1930,1930] marks the year that begins the *Great Depression* covered in document $D_{1349556}$. Clearly, this is a more temporally diverse and interesting set of documents output by our approach as compared to the baseline where documents focus more on the slump in markets in the 1990s.

7.6 Conclusion

In this chapter, we considered the task of diversifying search results by using temporal expressions in document contents. Our proposed probabilistic framework utilized time intervals of interest derived from the temporal expressions present in pseudo-relevant documents and then subsequently using them as aspects for diversification along time. To evaluate our method we constructed a novel testbed of history-oriented queries derived from authoritative resources and their corresponding *Wikipedia* entries. We showed that our diversification method presents a more complete retrospective set of documents for the given history-oriented query set. This approach described is largely intended to help scholars in history and humanities to explore large born-digital document collections quickly and find relevant information without knowing time intervals of interest to their queries.

Query: *george w. bush*	
Identified Time Intervals: [2000, 2000] ; [2000, 2004] ; [2004, 2004] ; [2004, 2007] ; [1992, 1992]	
NAÏVE	TIME-DIVERSE
$D_{1181696}$ – Heir Apparent? – 2000	$D_{1117027}$ – Ideas & Trends; Republicans Stalk a Slogan, Hunting for Themselves – 1999; 1992; 1996
$D_{1142543}$ – Rival Biographies of Bush Are Rushing to Print – 1999; 1992	$D_{1461580}$ – The World: A Calling to Heal; Getting Religion on AIDS – 2003; 1999; 1986; 1991; 1995
$D_{1242996}$ – THE 2000 CAMPAIGN: THE CANDIDATES; In Final Days, Rallying Supporters and Attempting to Sidestep a Volatile Issue – 2000; 1996	$D_{1610342}$ – THE 2004 CAMPAIGN: THE DEMOCRATIC NOMINEE; Kerry Invokes the Bible In Appeal for Black Votes – 2004; 2000
$D_{1609737}$ – A Ketchup Too Spicy for the G.O.P. – 2004	$D_{1255229}$ – THE 43rd PRESIDENT: THE MOOD IN TEXAS; Victory Celebration Is Tempered by Bush's Need to Focus on Reconciliation – 2000
$D_{1255563}$ – The George Bush I Knew – 2000; mid-1960's	D_{537116} – THE MEDIA BUSINESS: ADVERTISING; Bringing Madison Avenue Polish to Bush's Campaign Ads – 1992; 1988; 1974

Figure 7.2: Top-5 results for *george w. bush*

Query: *economic depression*	
Identified Time Intervals: [1990, 1990] ; [1990, 1995] ; [1987, 1987] ; [1930, 1930] ; [1998, 1998]	
NAÏVE	TIME-DIVERSE
D_{175692} – Economic Scene; Forecasters' Art In 1929 and Now – 1988; 1920's ;1929 ; 1930-31 ; 1929 ; mid-1931 ; 1929-30	D_{491246} – U.N. Report Warns of Crisis in Eastern Europe – 1988; 1929 to 1933 ; 1992 ; 1992
D_{653581} – Costs of Depression Are on a Par With Heart Disease, a Study Says – 1993; 1990	D_{113808} – WASHINGTON TALK: BRIEFING; 'The Great Correction' – 1988
D_{317417} – The Reagan Boom - Greatest Ever – 1990; 1930's;1980's;1989; 1982; 1990's;1960's;1990; 1970 to 1982; 1982; 1983; 1980's	D_{741011} – Let's Look at Biases in the History Standards; Skewed Economics – 1995; 1920's ;1995
D_{741011} – Let's Look at Biases in the History Standards; Skewed Economics – 1995; 1920's ;1995	D_{88390} – THE ECONOMIC HISTORIANS' VIEW: Comparing the Collapses; C.P. Kindleberger: Watch Prayerfully – 1987; 1929; 18th and 19th century; 1836; 1857; 1873; 1907; 1929; 1987; 1920
D_{327066} – Economic Scene; High Hopes And Deep Fears – 1990;1991	$D_{1349556}$ – Don't Blame Wall Street – 2001; 1929; 1920's;early 20's;90's;1921; 1991; 1920-21; 1921; 1913; 1927; 1912; 1920 to 1929; 1931; 1929; 1955; 1969; 1998; 1989; 1926; 1990's; 1939

Figure 7.3: Top-5 results for *economic depression*

CHAPTER 8

A PROBABILISTIC FRAMEWORK FOR TIME-SENSITIVE SEARCH

- Dhruv Gupta and Klaus Berberich. A Probabilistic Framework for Time-Sensitive Search: MPII at the NTCIR-12 Temporalia-2 Task. In NTCIR-12.

8.1 Introduction

Temporal Information Retrieval (T-IR) is a field concerned with organization of longitudinal document collections by utilizing the temporal information in them. In order to identify relevant documents for time-sensitive queries, a system must understand the temporal annotations present in the document contents. Naïve approaches that utilize document metadata such as creation time or publication date may be insufficient to address information needs for time-sensitive queries (e.g., *history of rap*). Analysis of temporal expressions in document contents is critical for any time-sensitive search engine. However, analysis of temporal expressions is difficult as they can vary with granularity (e.g., 1960 versus May 19, 1960) and further they can be highly uncertain (e.g., 1960s). Moreover given a implicitly time-sensitive query, where the temporal intent is not explicitly specified, prior art which largely relied on the signals derived from publication dates may fail. The temporal expressions again must be accounted by any method that attempts to retrieve or diversify documents by time. Advances in the field of T-IR have many benefits for scholars in humanities who need to analyze massive born-digital document collections for anthropological, historical, and linguistic trends.

In this chapter, we describe TimeSearch a probabilistic framework that utilizes temporal expressions in document contents to generate interesting time intervals for implicit time-sensitive queries. Further it utilizes the identified time intervals to re-rank and diversify documents along the temporal dimension. The system described is completely unsupervised in nature, i.e., it needs no training labels to function. The Temporalia-2 tasks that we participated in were: temporal intent disambiguation and temporally diversified retrieval.

Temporal Intent Disambiguation subtask required the participants to estimate the likelihood that the query has an information need in the classes: past, recent, future, and atemporal given a keyword query. Formally stated as:

Problem Temporal Intent Disambiguation. Given, classes $C = \{past,\ recent,\ future,\ atemporal\}$ and keyword query q, estimate $P(C|q)$.

Temporally Diversified Retrieval subtask required the participants to identify temporally relevant search results for a query in the classes: past, recent, future, and atemporal from the document collection Living Knowledge. Formally, we can state this subtask as follows:

Problem Temporally Diversified Retrieval. Given, keyword query q and document collection D, estimate $P(d|q, C)$.

System Overview. Our probabilistic framework consists of models constructed from our earlier research [93, 94, 96]. In a nutshell, our system analyzes the statistics of frequently occurring temporal expressions in relevant documents retrieved for a keyword query. Time is modeled so as to account for its inherent uncertainty [41]. Using this model we then generate interesting time intervals (in contrast to only time points in prior art) [93]. This analysis is carried out at multiple levels of temporal granularity. The time intervals are then used in a time-sensitive language model [41] and a time-sensitive diversification algorithm [96]. The integral components of the system in an overview are depicted in Figure 8.1.

Outline. The remainder of the chapter is structured as follows. We put our work into context with respect to prior art in Section 8.2. Section 8.3 describes the distribution of temporal expressions in the document collection used for the competition and how we pre-process the data for our methods. Section 8.4 describes the TimeSearch system. In Section 8.5 we describe our method and its performance in the temporal intent disambiguation subtask. In Section 8.6 we illustrate our approach and its results for the temporally diversified retrieval subtask. We present concluding remarks in Section 8.7.

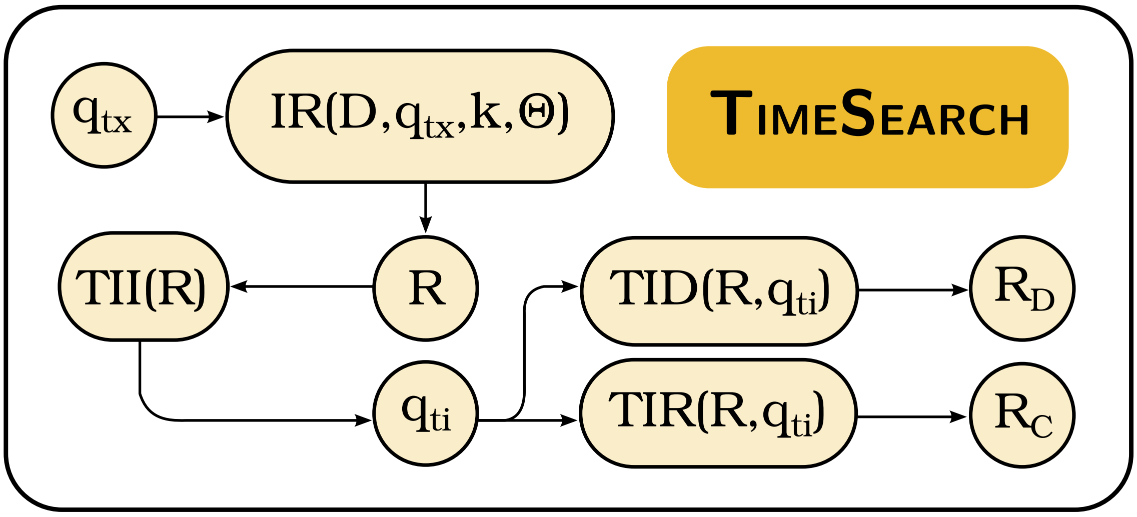


Figure 8.1: Given a keyword query q_{tx}, TIMESEARCH uses a pseudo-relevant set of documents R to identify interesting time intervals q_{ti} using the time intervals of interest model (TII). The time intervals q_{ti} are subsequently used for query expansion to obtain a temporally diversified set of documents R_D by temporal diversification model (TID). They are also used for retrieving temporally relevant documents R_C using the temporal language model (TIR).

8.2 Related Work

Temporal information retrieval is now a well established field of information retrieval which tries to analyze text and its accompanying temporal expressions. It has received substantial attention given the fact that around 1.5 % of web queries are explicitly time-sensitive in nature [168] and around 7 % of web queries are implicitly time-sensitive in nature [155]. We begin by what types of temporal expressions in text can be identified and which tools can be used to detect them. We then present the relevant prior art for the Temporalia-2 [126] task that belong to two broad classes: understanding time-sensitive queries and time-sensitive document retrieval.

Temporal Expressions in text can be of three types [53]: explicit, implicit, and relative. Explicit temporal expressions are exact mentions of date and time (e.g., April 1, 2005). Furthermore, these explicit temporal expressions may occur at different levels of granularity say, day (e.g., May 15, 1990), month (e.g., May, 1990), or year (e.g., 1990) level. Implicit temporal expressions are conveyed by words that carry a latent temporal intent behind them, e.g., spring. Relative temporal expressions may occur relative to a temporal expression elsewhere in text, e.g., last year. Temporal taggers such as SUTime [58] and HeidelTime [188] offer the capability to detect and resolve these temporal expressions to human-interpretable dates.

Understanding Time-Sensitive Queries. One of the early works in temporal query classification was by Jones and Diaz [127]. The authors described a taxonomy for temporal classes; which were ambiguous, unambiguous and atemporal. Their machine learning approach incorporated signals from the distribution of document publication dates. Some of these features were temporal clarity, kurtosis, and auto-correlation. The first edition of the Temporalia competition [125] considered temporal query classification with a qualitative set of temporal classes, namely: past, recency, and future. More recently additional classes have been explored by us [94] and by Kanhabua et al. [131]. Kanhabua et al. [131] study the case of seasonality and periodicity associated with web queries. They use features acquired from web query logs, and publication date distribution of an external document collection. In our earlier work [94], we additionally considered the task of disambiguating the temporal class of a query at multiple levels of granularity and also determining its temporal (a)periodicity. Our approach looked at publication dates as well as temporal expressions in document contents.

Time-Sensitive Document Retrieval. Berberich et al. [41] presented a time-sensitive language model that answers explicit temporal queries by analyzing temporal expressions in document contents. Building on this Kanhabua and Nørvåg [132] look at automatically suggesting years of interest to implicit temporal queries. To this end, they only utilize document publication dates. Work by Dakka et al. [70] relies on document publication dates to improve the retrieval effectiveness by analyzing their frequency histograms. A comprehensive survey of temporal information retrieval by Campos et al. [53], also notes the lack of any active research in the area of diversifying search results using temporal expressions. In our prior work [96], we presented an algorithm for diversifying search results that utilizes temporal expressions. This approach has been used in the larger system, TimeSearch, described in this chapter. An alternative approach would be to anchor documents in time which Jatowt et al. [123] address. They look at the problem of estimating the time period which the document focuses on. They do this by constructing a weighted undirected graph which captures the associations between terms and time.

Applications. We now discuss applications that leverage temporal expressions in document contents. Swan and Allan [191] investigate how to automatically generate timelines using text-based features. To do so, they first constructing a timeline of the corpus at day granularity. They then test the significance of the features, e.g., named entities and noun phrases via χ^2 statistic. Alonso et al. [30] specifically look at the temporal information contained in documents to organize and explore them along timelines constructed at multiple granularities. Furthermore, the authors propose the concept of temporal document profiles using which document clustering and re-ranking can be performed. Yeung and Jatowt [149] use temporal expressions and Latent Dirichlet Allocation (LDA) to study how the past is remembered in text collections. Using the distributions of topics along time, the authors identify significant topics in years, evolution of events over time, and historical similarities between countries.

8.3 Document Collection: Its Analysis and Indexing

For the subtasks in Temporalia, the Living Knowledge [15] Web collection was used. It comprises of news articles and blogs amounting to approximately 3.8 million documents [125]. The documents are provided with annotations for temporal expressions as well as named entities.

Temporal Analysis

We did a simple temporal analysis of the Living Knowledge document collection; by computing the document frequency of various temporal expressions at year granularity across the collection. The resulting plot is shown in Figure 7.1a. As can be seen from Figure 7.1a there is a large number of documents containing temporal expressions in the year "2011" and "2012". Table 8.1 gives the top-5 frequently occurring years with their relative document frequency to the total number of temporal expression containing documents. The kurtosis of the distribution is 421.39 and skewness of 19.85. This highly skewed nature of the distribution affects any probabilistic analysis that is performed on the Living Knowledge document collection. We take this background distribution of temporal expressions into account when analyzing time-sensitive queries.

Pre-processing and Indexing

We utilized the temporal expressions provided with corpus for analysis. We did not utilize any external temporal annotator. Each document was pre-processed using Hadoop map-reduce framework to a JSON document representation. The schema used for encoding each document is detailed in Figure 8.2. We store all the metadata (`category`, `host`, `pubDate`, `url`) along with all the temporal expressions (`allTime`) in the document. We also do not utilize the named entity annotations in provided with the document collection. We indexed the documents along with their temporal expressions using the ElasticSearch [4] software. For retrieval of a pseudo-relevant set of documents we used its built-in implementation of the Okapi BM25 method with parameters $k_1 = 2.00$ and $b = 1.00$.

An Illustrative Example

In order to illustrate the workings of the various models in TimeSearch, consider the following fictitious toy example illustrated in Figure 8.3. The likelihood of generating the query q given the document d (document score for the given query) is naïvely measured as a value proportional to the normalized product of term frequency of query terms in document. We will use this as a running example for explaining TimeSearch.

Time	**Frequency**
2011	0.31
2012	0.25
2010	0.10
2009	0.04
2008	0.04
2013	0.03

Table 8.1: Top-5 frequently occurring years with their relative frequency in the Living Knowledge document collection.

```
{
  "lk": {
   "mappings": {
     "docs": {
       "properties": {
         "allTime": {
           "properties": {
             "value": {
               "type": "string"
             }
           }
         },
         "category": {
           "type": "string"
         },
         "docId": {
           "type": "string"
         },
         "host": {
           "type": "string"
         },
         "pubDate": {
           "type": "string"
         },
         "source": {
           "type": "string"
         },
         "text": {
          "type": "string"
         },
         "url": {
           "type": "string"
         }
       }
      }
    }
  }
 }
```

Figure 8.2: JSON schema for a document in our index.

8.4 System Design

In this section, we outline the various components that were used to solve the subtasks in Temporalia-2. The entire system was programmed in Java.

Preliminaries

We consider a document collection $\mathcal{D}$. Each document $d \in \mathcal{D}$ consists of a bag of keywords d_{tx} and a bag of temporal expressions d_{ti}. We let $|d_{tx}|$ and $|d_{ti}|$ denote the cardinalities of these bags. Also, let $\text{tf}(v, d)$ denote the term frequency of the keyword v, drawn from vocabulary V, in document d. Let, q_{tx} denote the keywords of the query. To retrieve the pseudo-relevant set of documents R, we utilize a retrieval method:

$$R = \text{IR}(\mathcal{D}, q_{tx}, k, \Theta),$$

where, D is the document collection, k is the number of top-k results required and $\Theta \in \mathbb{R}^m$ denotes a set of parameters. Each document $d \in R$ is further accompanied by a document score. To represent and model temporal expressions we leverage the work by Berberich et al. [41]. The uncertainty-aware time model has been described in Section 5.3.2.

Time Intervals of Interest

Given a keyword query, we first identify interesting time intervals, which are used for predicting its temporal intent and subsequently for retrieval of documents that satisfy the temporal intent. Time intervals of interest to a given keyword query q_{tx} are identified using our earlier work [93]. The probability that a time interval $[b, e]$ is deemed interesting for a given keyword query q_{tx} is modeled as a two-step generative model:

$$P(\,[b, e] \mid q_{tx}\,) = \sum_{d \in R} P(\,[b, e] \mid d_{ti}\,) \cdot P(\,d \mid q_{tx}\,).$$

The first step involves involves retrieving a pseudo-relevant set of documents R using the keyword query q_{tx}. The probability $P(d|q_{tx})$ thus measures the likelihood of generating the document given the query. This is estimated by using the document scores given by the retrieval method. In the second step, a time interval $[b, e]$ is in turn generated from each of the temporal expressions in d:

$$P(\,[b, e] \mid d_{ti}\,) = \frac{1}{|d_{ti}|} \sum_{T \in d_{ti}} \frac{\mathbb{1}([b, e] \in T)}{|T|}.$$

Generating time intervals immediately at day granularity is an expensive operation; since it may require discretization of the temporal dimension into hundreds of thousands of days. Subsequently representing each interval in our time model increases the space complexity quadratically. Thus, to overcome

Query: `summer olympics`		
Id	**Contents**	**Score**
d_1	summer olympics 2008 took place in beijing, china.	0.25
d_2	summer olympics 2012 took place in london, england.	0.25
d_3	summer olympic games during 1990s were very competitive.	0.25
d_4	summer olympic games during August, 1992 to September, 1992 were very competitive.	0.25
d_5	olympic games were competitive during 1973.	0.17

Figure 8.3: A toy example showing an ordered-set of pseudo-relevant documents returned for the keyword query `summer olympics`.

the problem we apply the generative model recursively to obtain time intervals of interest at year, month, and day granularity. That is, we first identify interesting years, for those years we generate interesting months and subsequently the interesting days in those months. The time intervals generated for a given keyword query q_{tx} are kept in set q_{ti}. For determining the intent we utilize the time intervals at year granularity.

Let us apply this model to the toy example given in Figure 8.3 to understand the intuition. Assume all documents are in the pseudo-relevant set, i.e., $R = \{d_1, d_2, d_3, d_4, d_5\}$. Since the scores of documents $\{d_1, d_2, d_3, d_4\}$ are larger than d_5; this indicates that the temporal expressions in them have a higher relevance to the temporal intent behind the keyword query.

Next all temporal expressions in the documents are represented in our time model and time intervals are generated. Consider the interesting case of {1990s; August, 1992; September, 1996}. The time interval $[08 - 1992, 09 - 1996]$ gets a higher relevance to the query then the other temporal expressions present due to high redundancy (see Figure 8.4). Thus the top-3 time intervals will be $\langle[08 - 1996, 09 - 1992], [2008, 2008], [2012, 2012]\rangle$, with $[2008, 2008]$ and $[2012, 2012]$ having equal likelihood (but less than $[08 - 1996, 09 - 1996]$) of being generated from the documents.

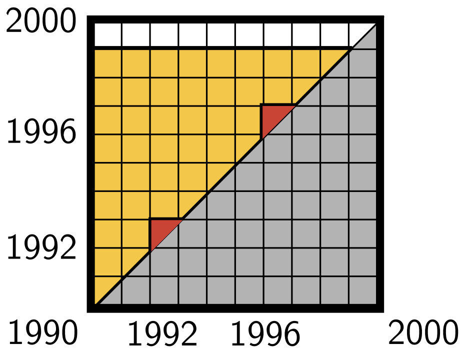


Figure 8.4: Graphical illustration of how the temporal expressions {1990s; August, 1992; September, 1996} overlap in our time model. The grayed-out region below the diagonal represents invalid time intervals.

Temporal Language Model

Having obtained a set of interesting time intervals q_{ti}, we use them for retrieving documents that lie in these time intervals as explained next. Given a query q_{tx}, we utilize the approach previously developed by Berberich et al. [41] for re-scoring the documents after expanding it with time intervals from q_{ti}. Assume an independence between generation of textual q_{tx} and temporal intents q_{ti} of query. The probability of generating the query q can be written as [41]:

$$P(q|d) = P(q_{tx}|d_{tx}) \cdot P(q_{ti}|d_{ti}).$$

The first probability $P(q_{tx}|d_{tx})$ can be estimated as described earlier using document scores. The second probability gives the likelihood of generating a time interval of interest $[b, e] \in q_{ti}$ from the document's temporal expressions d_{ti} as follows:

$$P(q_{ti}|d_{ti}) = \prod_{[b,e]\in q_{ti}} P([b, e] \mid d_{ti}).$$

To understand this model, let us return to the toy example. The current document ranking is $\langle d_1, d_2, d_3, d_4, d_5\rangle$. Let the query `summer olympics` be expanded with the time interval $[08 - 1996, 09 - 1996]$. Since documents $\{d_3, d_4\}$ both have temporal expressions that contain the interesting time interval $[08-1992, 09-1992]$ they will be promoted up in the rankings, followed by $\langle d_1, d_2\rangle$; and finally with $\langle d_5\rangle$ at the end with no temporal expression that mentions a interesting time interval. Thus, the final ranking with this model will be $\langle d_3, d_4, d_1, d_2, d_5\rangle$.

Temporal Diversification

The set of interesting time intervals q_{ti} can also be used as temporal intents to perform temporal diversification, as described in our work [96]. This is done by adapting the work of Agarwal et al. [25], to maximize the following objective function over the temporal intents:

$$\sum_{[b,e]\in q_{ti}} P(\,[b, e] \mid q_{tx}\,)\left(1 - \prod_{d\in R_D} \Big(1 - P(\,q_{tx} \mid d_{tx}\,)P(\,[b, e] \mid d_{ti})\Big)\right).$$

The objective maximizes the probability that at least one document from each temporal intent is in the diversified set of results R_D. The importance of the temporal intent is given by the probability $P([b, e]|q_{tx})$ as described earlier. The probability that a document satisfies this temporal intent is estimated by:

$$P(\,q_{tx} \mid d_{tx}\,)\,P(\,[b, e] \mid d_{ti}).$$

Lets apply this model on the toy example. Assume that we require only top-2 documents in our diversified set $|R_D| = 2$. In the set of documents we will see that the pairs $\{d_1, d_3\}$, $\{d_1, d_4\}$, $\{d_2, d_3\}$, and $\{d_2, d_4\}$ are the only optimal sets for the objective function above. This is because each document in them represents at least one of the temporal intents. Hence we can choose any one of them as R_D.

8.5 Temporal Intent Disambiguation Subtask

We next discuss our approach to address the Temporal Intent Disambiguation subtask. In this section, we describe the query set description, our method and the metrics used for evaluation.

Query Set. The organizers of the Temporalia-2 task made available to the participants a total of 93 queries in the dry run; consisting of 73 training queries and 20 queries for testing. For the formal run; we were provided 300 queries in total. Each query consists of the following fields: query keywords, query issue time and and probabilities for four different classes (training data). A sample query with the markup is displayed in Figure 8.5.

Method. In the Temporal Intent Disambiguation subtask, we were asked to estimate $P(C \mid q)$ given the classes $C = \{past,\ recent,\ future,\ atemporal\}$. For this we used the probability distribution $P([b,e]|q)$ of unit time intervals at the year granularity. For the atemporal class, we can compute the probability $P(C = atemporal|q)$ as:

$$P(C = \text{atemporal}|q) = \sqrt{|T|} \max_{[b,e]\in T} |P([b,e]|q) - P([b,e]|D_{ti})|,$$

which is essentially a two-sample Kolmogorov-Smirnov test [152] between the distribution of unit time intervals estimated by our model and the distribution of unit time intervals occurring in the background document collection. It tests whether these two samples were generated by a common distribution. For this we utilized the implementation in Apache Commons Math 3.6 API[1]. For the past, recent and future class, we utilize the query issue time at year granularity t_{issue}. We specifically look at the orientation of the interesting time intervals with respect to the query issue time to determine whether the temporal intent lies in past, present or future. Thus the probabilities for the different classes are measured as follows:

$$P(C = \text{past }|q) = \frac{1}{|T|} \sum_{[b,e]\in T} \mathbb{1}(t_{issue} < b),$$

$$P(C = \text{recent}|q) = \frac{1}{|T|} \sum_{[b,e]\in T} \mathbb{1}(b \le t_{issue} \le e),$$

$$P(C = \text{future}|q) = \frac{1}{|T|} \sum_{[b,e]\in T} \mathbb{1}(t_{issue} > e).$$

[1]`https://commons.apache.org/proper/commons-math/apidocs/org/apache/commons/math3/stat/inference/KolmogorovSmirnovTest.html`

```
<query>
  <id>033</id>
  <query_string>
        weather in London
  </query_string>
  <query_issue_time>
        May 1, 2013 GMT+0
  </query_issue_time>
  <probabilities>
        <Past>0.0</Past>
        <Recency>0.9</Recency>
        <Future>0.1</Future>
        <Atemporal>0.0</Atemporal>
  </probabilities>
</query>
```

Figure 8.5: Sample query from TID subtask

Metrics. The organizers of the task provided the participants with two metrics, loss and similarity, for evaluating the participating systems. The loss and similarity metrics are computed per query and then averaged to obtain the final system performance. **Loss** between two k-discrete probability distribution M and N is measured as:

$$\text{loss} = \frac{1}{k} \cdot \sum_{i=1}^{k} \mid m_i - n_i \mid .$$

Similarity between two k-discrete probability distribution M and N is measured by treating each distribution as a vector in k-dimensional space and computing the cosine of the angle between them:

$$\text{sim} = \frac{\sum_{i=1}^{k} m_i \cdot n_i}{\sqrt{\sum_{i=1}^{k} m_i^2} \cdot \sqrt{\sum_{i=1}^{k} n_i^2}}.$$

Results. At each stage of the competition we kept the same system configuration by utilizing the top-1000 documents per query for determining the time intervals of interest. As an initial experiment we utilized all the 73 training queries for testing our system. This was possible since our method is unsupervised in nature. A rudimentary baseline for our system can be a uniform distribution assigned to all four classes, i.e., $P(C|q) = 0.25$. The results for this training experiment is shown in Table 8.2. Results for the two systems named Mpii-Tid-Train and Baseline corresponding to our method and the baseline respectively are reported in Table 8.2. For the **dry-run** we submitted a single run for the set of 20 queries in the dry-run query set. Our systems performance is reported against the system named Mpii-Tid-Dry in Table 8.2. For the **formal-run** we again submitted a single run for the set of 300 queries provided for the formal-run. The results for this stage are report against the system named Mpii-Tid-Formal in Table 8.2.

System	Loss	Similarity	#Queries
Mpii-Tid-Formal	0.35	0.35	300
Mpii-Tid-Dry	0.34	0.39	20
Mpii-Tid-Train	0.30	0.48	73
Baseline	0.26	0.66	

Table 8.2: Results for our proposed system at different stages of the temporal intent disambiguation subtask.

Discussion. The proposed method for predicting the probabilities of the temporal classes overall shows poor performance as compared to a naïve baseline. One potential reason we suspect is the temporal bias in the distribution as discussed in Section 8.3. For most queries the interesting time intervals arose in the time interval $[2011, 2013]$. For queries such as: *uk 2009 balance of payments*, *the advantages of hosting the olympic games*, *freedom of information act*, *when did ww2 start*, *how did bin laden die*, *when was television invented*, *history of slavery*, *susan miller 2012*, and *occupy wall street movement* our system predicted the temporal class distribution with high similarity and low loss. This shows us that given an explicit temporal query or a history-oriented query, our method can predict the distribution quite well. However, for queries such as: *naming university buildings with commercial brands*, *body posture alteration*, *dressing code in job interview*, *badminton games*, *advanced english*, and *time warner austin* the predicted distribution deviated from the human-provided distribution. This observation leads us to suspect that comparing the distribution of time intervals of interest with respect to the background distribution for the atemporal class may not work for all atemporal queries. An alternative approach will be to resort to a learning approach whereby the distribution for the atemporal class can be induced from the training set of queries.

8.6 Temporally Diversified Retrieval Subtask

This subtask required the participants to retrieve documents for each four of the temporal classes as well a diversified set of documents along time for a given query topic. In this section, we describe the query set, our method and the results obtained for our system at various stages of the competition for the temporally diversified retrieval subtask.

Query Set. The organizers provided us with 8 queries in the dry run and 50 queries in the formal run to evaluate. Each query consisted of the following fields: query title; query description; query issue time; and query subtopic description for past, recent, future and atemporal. A sample query from this set is shown in Figure 8.6. As an input to our system we chose only to use the keywords from the `query title` field.

```
<topic>
 <id>002</id>
 <title>
             Junk food health effect
 </title>
 <description>
             I am concerned about the health effects of junk food in general.
             I need to know more about their ingredients, impact on health,
             history, current scientific discoveries and any prognoses.
 </description>
 <query_issue_time>
             Mar 29, 2013 GMT+0:00
 </query_issue_time>
 <subtopics>
   <subtopic id="002a" type="atemporal">
             How junk foods are defined?
   </subtopic>
   <subtopic id="002p" type="past">
             When did junk foods become popular?
   </subtopic>
   <subtopic id="002r" type="recency">
             What are the latest studies on the effect of junk foods on our health?
   </subtopic>
   <subtopic id="002f" type="future">
             Will junk food continue to be popular in the future?
   </subtopic>
 </subtopics>
</topic>
```

Figure 8.6: Sample query from TDR subtask

Temporal Ranking and Diversification of Documents. For retrieval of time-sensitive documents we utilized the temporal language model (outlined in Section 8.4). For retrieving time-sensitive documents, we first determine q_{ti} for the given keyword query as follows. For the class recent, we utilized t_{issue}, i.e., $q_{ti} = t_{issue}$. For the class atemporal, the retrieved documents were the same as given by Okapi BM25 retrieval scheme. For classes past and future, we considered interesting time intervals that lie before t_{issue} and after t_{issue} respectively. Temporal Diversification of documents was done by considering top-5 identified interesting time intervals as temporal categories for the diversification algorithm.

Evaluation Metrics. For the evaluation of this task the organizers used the standard Cranfield methodology. This is done by first pooling all relevant documents from the submitted runs of participants. Subsequently their relevance grade is identified via online crowdsourcing methods. To assess the quality of ranking in specific classes (e.g., past, recent, future, and atemporal), the organizers used nDCG to measure retrieval effectiveness. While for diversified list of documents, α-nDCG and D#-nDCG was used.

Results. For each stage of the competition we submitted a single run comprising of top-100 documents for each temporal class and for the diversified set of documents. We report the results in Table 8.3 and 8.4 for metrics that we discussed above for the dry-run and formal-run stage of the competition for our systems.

Category	**Dry-run nDCG@20**	**Formal-run nDCG@20**
Atemporal	0.17	0.34
Past	0.19	0.39
Recent	0.05	0.34
Future	0.02	0.34
All	0.11	0.35

Table 8.3: Results for our proposed system for retrieving time-sensitive documents at different stages of the temporally diversified retrieval subtask.

Stage	**nDCG@20**	**D#-nDCG@20**
Dry-run	0.18	0.41
Formal-run	0.33	0.57

Table 8.4: Results for our proposed system for diversifying time-sensitive documents at different stages of the temporally diversified retrieval subtask.

Discussion. Concerning the temporal retrieval of documents; from the results we observe that for the dry run stage our system performed very well in the future and past classes. However, it did not perform well for the recent class. On the other hand for the formal run our system performed well for the class past and equally well for the rest of the classes. As for the temporal diversification of documents; our system performed well in formal run stage as compared to the dry run stage. Overall comparing to the organizers' system, our method did not fare as well. This can be attributed largely to two insights: the role of the retrieval method for producing an initial set of pseudo-relevant documents and the role that temporal expressions play in our approach. In order to improve our system we can attempt to replace the current Okapi BM25 method with other state-of-the-art retrieval methods. We also did not use any external temporal annotator and opted in favor for the annotations provided with the document collection. The provided annotations had a temporal bias which we discussed in Section 8.3, which we suspect may be the reason our system does not perform well.

8.7 Conclusion

In this chapter, we described TimeSearch a probabilistic framework for time-sensitive search. It understands the temporal uncertainty in time, which we leverage to generate time intervals of interest to a given keyword query. These time intervals are then used to retrieve time-sensitive documents or to generate a temporally diverse set of documents. Our methods for the Temporalia-2 task utilize this system in order to identify the temporal intent in past, recent, future, atemporal classes and to retrieve time-sensitive documents in those classes.

CHAPTER 9

GENERATING SEMANTIC ASPECTS FOR QUERIES

• Dhruv Gupta, Klaus Berberich, Jannik Strötgen, and Demetrios Zeinalipour-Yazti. Generating Semantic Aspects for Queries.
In JCDL 2018, pages 335–336.

• Dhruv Gupta, Klaus Berberich, Jannik Strötgen, and Demetrios Zeinalipour-Yazti. Generating Semantic Aspects for Queries.
In ESWC 2019, pages 162–178.

9.1 Introduction

When querying large document collections or the Web, it is challenging to guide the user to relevant documents. This is because short and ambiguous keyword queries posed to information retrieval (IR) systems represent many possible information needs [62]. This is a known acute problem; it has been shown that around 46% of users issue reformulated queries [113]. To assist users in refining their search, existing approaches use related terms [181], named entities in knowledge graphs (KGs) [48] or hand-crafted knowledge-panels [115]. Still with these aids, the user must read and consult individual documents in the ranked list to check for their relevance. What is therefore needed is a way of

Query: *olympic medalists*		
Time	**Entities**	**Entity Type**
[1980,1988]	*yago:sergei-grinkov, yago:maya-usova, yago:marina-klimova, yago:evgeni-platov, yago:oksana-grishuk, yago:sergei-ponomarenko, yago:ekaterina-gordeeva*	*wiki-category: olympic medalists in figure skating*
[1992,1992]	*yago:leroy-burrell, yago:jon-drummond, yago:dennis-mitchell, yago:michael-marsh-(athlete)*	*wiki-category: american sprinters*
[1996,1996]	*yago:dominique-dawes, yago:shannon-miller, yago:kerri-strug*	*wiki-category: olympic medalists in gymnastics*
[1998,1998]	*yago:jenni-meno, yago:kyoko-ina, yago:todd-eldredge, yago:todd-sand, yago:nicole-bobek*	*wiki-category: figure skaters at the 1998 winter olympics*

Table 9.1: Generated semantic aspects for the query *olympic medalists* from the New York Times document collection (covering 1987-2007). Each row in the table corresponds to one semantic aspect.

uplifting the unstructured text in documents to a structured representation that exposes its key *aspects*. To this end, we propose the novel concept of *semantic aspects* (e.g., ⟨{MICHAEL PHELPS}, {ATHENS, BEIJING, LONDON}, [2004,2016]⟩) that help users posing ambiguous queries (e.g., *olympic medalists*) explore document collections without reading their contents.

To generate semantic aspects for ambiguous keyword queries, we turn to natural language processing (NLP) tools that can help us enrich text with annotations. In particular, we make use of annotations in the form of named entities (persons and organizations), geographic locations, and temporal expressions. These are extremely important annotations in the domain of IR [91, 211]: 71% of Web queries were found to mention named entities, while 30.9% of Web queries were either explicitly or implicitly temporal in nature. Generation of meaningful semantic aspects and their evaluation is challenging. To generate them, we must first model and interpret the semantics underlying the annotations. For example, temporal expressions can be highly uncertain (e.g., *90s*) and two locations or named entities in a KG can be related by many facts, e.g., 'Maria Sharapova lives in US but represents Russia in sports' [17]. Moreover, queries can signify different kinds of ambiguities: temporal ambiguity (e.g., *tokyo summer olympics* – 1964 or 2020), location ambiguity (e.g., *rome* – many US cities are named after European cities), or entity ambiguity (e.g., *spitz* – Mark or Elisa Spitz). Moreover, since semantic aspects are more than "related terms" or facts in KGs there currently exists no benchmark for their automatic evaluation.

Approach Outline. To solve the above challenges, we propose the following solutions in this chapter. To generate semantic aspects, we describe the XFACTOR algorithm (Section 9.5). XFACTOR takes as an input a large set of annotated documents retrieved for a keyword query and outputs a set of semantic aspects. XFACTOR generates the semantic aspects in three steps. First, it partitions the input document set by identifying salient sets of annotations in formal models that capture the semantics of an annotation type (e.g., named entities). Second, xFactor additionally considers salient co-occurrences of annotations with different semantics (e.g., named entities and temporal expressions) by virtue of them being present in the same document partition. Third, it outputs all possible ways of analyzing the initial ambiguity behind the query. This is done by permuting the order in which the annotations are considered for partitioning the initial set of documents. Table 9.1 shows examples of generated semantic aspects for the ambiguous query ***olympic medalists***. To perform automated evaluation of the generated semantic aspects, we provide a novel evaluation benchmark compiled from Wikipedia with new measures to the research community (Section 9.7).

9.2 Related Work

Structuring Text for Search. Hearst and Plaunt [114] proposed TextTiling, an algorithm for identifying coherent passages (subtopics) in text documents. Koutrika et al. [135] utilized LDA to identify topics in documents for their structured representation. However, both approaches were not informed of semantic annotations, which we leverage to identify aspects for structuring text for search.

Faceted Search. Faceted Search systems allow a user to navigate document collections and prune irrelevant documents by displaying important features about them. Dou et al. [74] and Kong and Allan [134] rely only on text to mine keyword lists present in pseudo-relevant documents for generating aspects. Ben-Yitzhak et al. [39] discussed various algorithms that allowed business intelligence aggregations and advanced dynamic discovery of correlated facets across multiple dimensions. Li et al. [142] leveraged semantic metadata present in Wikipedia such as entities and their associated category for automated generation of facets for exploring Wikipedia articles. Grau et al. [88] considered the use of named entities and their relationships in graphs for generating facets in DBpedia abstracts. Our approach, in contrast, considers the underlying semantics for each annotation during the generation of aspects. We additionally consider temporal expressions and geographic locations as annotations. Moreover, we model the co-occurrences between different annotation types for generating aspects.

Temporal Search. Tran et al. [198] analyzed annotated documents for recommending related entities given an entity and an associated text. For this, the authors used temporal expressions, KGs and word embeddings. In a similar vein, Bianchi et al. [44] looked into incorporating temporal knowledge into embeddings for KGs. Nguyen et al. [165] on the other hand, analyze query logs to recommend time intervals for queries about events. In contrast, our approach models the uncertainty behind temporal expressions when generating query aspects. Our xFactor algorithm is additionally extensible to other annotation types (e.g., locations, numbers, and sentiment). Moreover, xFactor allows query pivoting, thereby disambiguating query intent using different annotation types.

Entity Search. Reinanda et al. [177] leverage search-engine query logs to mine and suggest entities of relevance given an entity-oriented query. They consider metadata associated with the queries in the query log for their approach, e.g., user clicks, user sessions, and query issue timestamps. Schuhmacher et al. [184] propose a method for recommending related entities for entity-centric queries using pseudo relevance feedback from retrieved documents and KGs. Their work, however, does not tap into the document contents or temporal expressions for generating aspects.

9.3 Preliminaries

Document Model. Consider, a large annotated document collection $\mathcal{D} = \{d_1, d_2, \ldots, d_{|\mathcal{D}|}\}$. Each document $d \in \mathcal{D}$ is processed with natural language processing (NLP) annotators that tag sequences of words in the document with annotations from a given type $\mathcal{X}$ (e.g., temporal expressions). Formally, let each document be represented by n bags-of-annotations:

$$d = \{d_{\mathcal{X}_1}, d_{\mathcal{X}_2}, \ldots, d_{\mathcal{X}_n}\}. \tag{9.1}$$

Specifically, for this work, we consider the following semantic annotations: temporal expressions, geographic locations, and other named entities (persons and organizations). Thus, our document model refers to the bags-of-annotations for entities ($d_{\mathcal{E}}$), locations ($d_{\mathcal{G}}$), and temporal expressions ($d_{\mathcal{T}}$):

$$d = \{d_{\mathcal{E}}, d_{\mathcal{G}}, d_{\mathcal{T}}\}.$$

Aspect Model. Let an ambiguous query q reflect the *information need* of a user. An information retrieval (IR) method in response to the query q returns a set of pseudo-relevant documents $\mathcal{R} \subset \mathcal{D}$. The document set $\mathcal{R}$ conveys many implicit information needs. The user's information need may be a subset of those reflected by the documents [62]. Our aim is to extract an ordered set of semantic aspects $\mathcal{A}$ that make these implicit information needs explicit, thereby pointing the user to the relevant documents:

$$\mathcal{A} = \langle a_1, a_2, \ldots, a_{|\mathcal{A}|}\rangle. \tag{9.2}$$

An aspect a (e.g., $\langle$\{MICHAEL PHELPS\}, \{ATHENS, BEIJING, LONDON\}, [2004, 2016]$\rangle$) is determined by considering the salience of annotations sharing the same semantics (e.g., temporal expressions) as well as co-occurrence with annotations of different semantics (e.g., temporal expressions and named entities). An aspect $a \in \mathcal{A}$ is modeled as:

$$a = \langle x_1, \ldots, x_n\rangle,$$

where, x (e.g., time interval) corresponds to salient annotation(s) from type $\mathcal{X}$ (e.g., temporal expressions). For this work, the aspects are modeled as: $a = \langle a_{\mathcal{E}}, a_{\mathcal{G}}, a_{\mathcal{T}}\rangle$.

Annotation Models. For large-scale enrichment of text documents with semantic annotations we utilize two different NLP tools. First, we make use of a named entity recognition and disambiguation (NERD) tool Aida [118] to annotate and disambiguate mentions of named entities to canonical entries in KGs (i.e., Yago [190]). Second, to resolve expressions of time in text we leverage a temporal tagger HeidelTime [188] that can annotate them with high precision. We next explain the formal models for each of the annotation types.

Named Entities and Entity Model. Named entities in text are modeled as canonical entries of a KG (e.g., Yago [190]). These annotations are obtained by using NERD tools (e.g., Aida [118]). We differentiate between locations and other named entities, by the presence of geographic coordinates in their KG entry. Let, $\mathcal{G}$ and $\mathcal{E}$ denote the type information associated with locations and other entities, respectively. Named entities may share common relationships that convey a degree of their relatedness. For example, *tokyo* and *beijing* are related as they both lie in *asia*. These relationships in Wikipedia are encoded in the form of an explicit link structure. Concretely, each named entity mention disambiguated by Aida, is linked to its Wikipedia article. Each Wikipedia article contains links to other Wikipedia articles, indicating their semantic relatedness. We model each named entity by its Wikipedia link structure. Formally, each named entity e can be described by the links ℓ its article W_e has to other articles in Wikipedia W:

$$W_e = \{\ell_1, \ell_2, \ldots, \ell_{|W|}\}. \tag{9.3}$$

Temporal Expressions and Time Model. Temporal expressions in documents can be annotated using temporal taggers (e.g., HeidelTime [188]). Such annotators are able to identify and resolve explicit, implicit, relative, and underspecified temporal expressions using metadata such as publication dates [188]. Let, $\mathcal{T}$ denote the type information associated with temporal expressions. Each annotation in $\mathcal{T}$ (i.e., dates) is represented by their UNIX time epochs (i.e., number of milliseconds since 01-January-1970). To model temporal expressions, we utilize the time model that incorporates uncertainty (see Section 5.3.2). With the uncertainty-aware time model each temporal expression can be represented as a four-tuple of time elements as:

$$T = \langle b_\ell, b_u, e_\ell, e_u \rangle. \tag{9.4}$$

For example, *1990s* can now be represented as: $\langle 1990, 1999,\ 1990, 1999 \rangle$.

9.4 Generating Factors

To generate semantic aspects for a given query, we first need to compute salience of annotations in models informed of their semantics. Thus, we are not simply counting annotations but rather considering the salience of entities, locations, and temporal expressions in their respective semantic models. We denote the methods that compute salience as **factor methods** and the resulting salient annotations as **factors** (e.g., sets of locations or time intervals). We next describe how to find factors associated with each annotation type $\mathcal{X}$ in a document set $\mathcal{R}$ by using its factoring method, $\text{factor}(\mathcal{X}, \mathcal{R})$. Algorithm 8 outlines how factor methods use the salience computation for a given document partition and annotation type.

Factoring Named Entities - $\text{factor}(\mathcal{X}_\mathcal{G}, \mathcal{R})$ and $\text{factor}(\mathcal{X}_\mathcal{E}, \mathcal{R})$. The factor method for named entities outputs sets of entities and locations where each entity in the set is related to the others in highly relevant documents. Concretely, to create the factors (i.e., sets of locations and entities) we first compute: $\text{sim}(e, e')$, semantic relatedness between entities e and e'. This is done by calculating the Jaccard coefficient of links shared by the Wikipedia entries of e and e':

$$\text{sim}(e, e') = \frac{|W_e \cap W_{e'}|}{|W_e \cup W_{e'}|}, \tag{9.5}$$

We make use of the Jaccard coefficient as an entity relatedness measure, as it has shown good performance over other relatedness measures [57]. Second, we weight the entity-entity relatedness by the document relevance $s(d, \mathcal{R})$ (given by the IR method during retrieval) that contains them to compute entity salience $\text{s}(e, \mathcal{R})$. That is,

$$\text{s}(e, \mathcal{R}) = \sum_{d \in \mathcal{R}} \text{s}(d, \mathcal{R}) \cdot \sum_{e' \in d_\mathcal{E}} \text{sim}(e, e'). \tag{9.6}$$

Factoring Time - $\text{factor}(\mathcal{X}_\mathcal{T}, \mathcal{R})$. Temporal expressions are challenging to analyze. For instance, uncertain temporal expressions such as *the 90s* can refer to an infinite number of time intervals. Thus, it becomes quite difficult to identify salient time intervals. To overcome these limitations, we use the approach described in Chapter 5 to generate factors for time. In brief, salient time intervals (time factors) in $\mathcal{R}$ (i.e., $\text{s}([b, e], \mathcal{R})$) can be found by generating overlaps of the temporal expressions in the uncertainty-aware time model and weighting them by the document's relevance, which contains the temporal expressions:

$$\text{s}([b, e], \mathcal{R}) = \sum_{d \in \mathcal{R}} \text{s}(d, \mathcal{R}) \cdot \text{sim}([b, e], d_\mathcal{T}), \tag{9.7}$$

where, $\text{s}(d, \mathcal{R})$ denotes the document relevance and $\text{sim}([b, e], d_\mathcal{T})$ denotes the salience of the time interval $[b, e]$ in the document's bag of temporal expressions $d_\mathcal{T}$. The value of $\text{sim}([b, e], d_\mathcal{T})$ is computed as follows:

$$\text{sim}([b, e], d_\mathcal{T}) = \frac{1}{|d_\mathcal{T}|} \cdot \sum_{T \in d_\mathcal{T}} \frac{\mathbb{1}([b, e] \in T)}{|T|}. \tag{9.8}$$

In Equation 9.8, the cardinality $|T|$ denotes the number of time intervals T can generate and the indicator function $\mathbb{1}(\bullet)$ tests the membership of $[b, e]$ in T.

9.5 The xFactor Algorithm

In addition to annotation salience, we consider the co-occurrence salience of annotations from different types to generate semantic aspects. To this end, we propose the xFactor algorithm. Our xFactor algorithm is inspired by the Apriori algorithm for frequent itemset mining [26]. The Apriori algorithm, however, is not informed of annotation semantics. Thus, its direct application, will not capture any semantic co-occurrence among different annotation types.

Consider, a document set $\mathcal{R}$ and its n annotation types $\{\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_n\}$. A set of salient aspects $\mathcal{A}$ can be derived by iteratively partitioning $\mathcal{R}$ for different annotation factors:

$$\text{Basis Step} : \{x_1\} = \text{factor}(\mathcal{X}_1, \mathcal{R}) \tag{9.9}$$

$$\text{Inductive Step} : \{x_k\} = \text{factor}(\mathcal{X}_k, \mathcal{R}^{(k-1)\ldots(1)}). \tag{9.10}$$

First and foremost, the salience of a factor (e.g., time interval) in each aspect is obtained by the factor method (Section 9.4) that considers a semantic model corresponding to its annotation type (e.g., temporal expressions). Second, each factor for an annotation type allows us to partition the document set $\mathcal{R}$ into documents that contain annotations that helped derive the factor and those documents which did not help. Thus, by iteratively applying the factor methods for the different annotation types, we can identify the salience between factors by virtue of their co-occurrence in the same partition. Mathematically given by:

$$\text{factor}(\mathcal{X}_1, \mathcal{R}) : \{d_{\mathcal{X}_1} \in d \mid \forall d \in \mathcal{R}\} \to 2^{\mathcal{X}_1}, \tag{9.11}$$

$$\text{factor}(\mathcal{X}_k, \mathcal{R}^{(k-1)\ldots(1)}) : \{d_{\mathcal{X}_k} \in d \mid \forall d \in \mathcal{R}^{(k-1)\ldots(1)}\} \to 2^{\mathcal{X}_k}. \tag{9.12}$$

Third and finally, we create a partition index that keeps track of factors that were generated by a particular partition of the pseudo-relevant set of documents $\mathcal{R}$. The partition index for iteration i of the recursive algorithm keeps track of:

$$\textsf{Partition Index:}\ \langle \{x_k\}_i, R_i^{(k-1)\ldots(1)} \rangle. \tag{9.13}$$

By concatenating the factors of different annotation types, from the same document partition, we can generate the aspects. Figure 9.1 exemplifies the recursive xFactor algorithm and how the aspects are generated. The xFactor algorithm thereby allows us to extract a subset of aspects that contain salient relationships among their factors from all possible combinations of different annotation types:

$$\mathcal{A} \subset 2^{\mathcal{X}_1 \times \mathcal{X}_2 \times \ldots \times \mathcal{X}_n}.$$

Algorithm 9 illustrates a tail-recursive version of the xFactor algorithm.

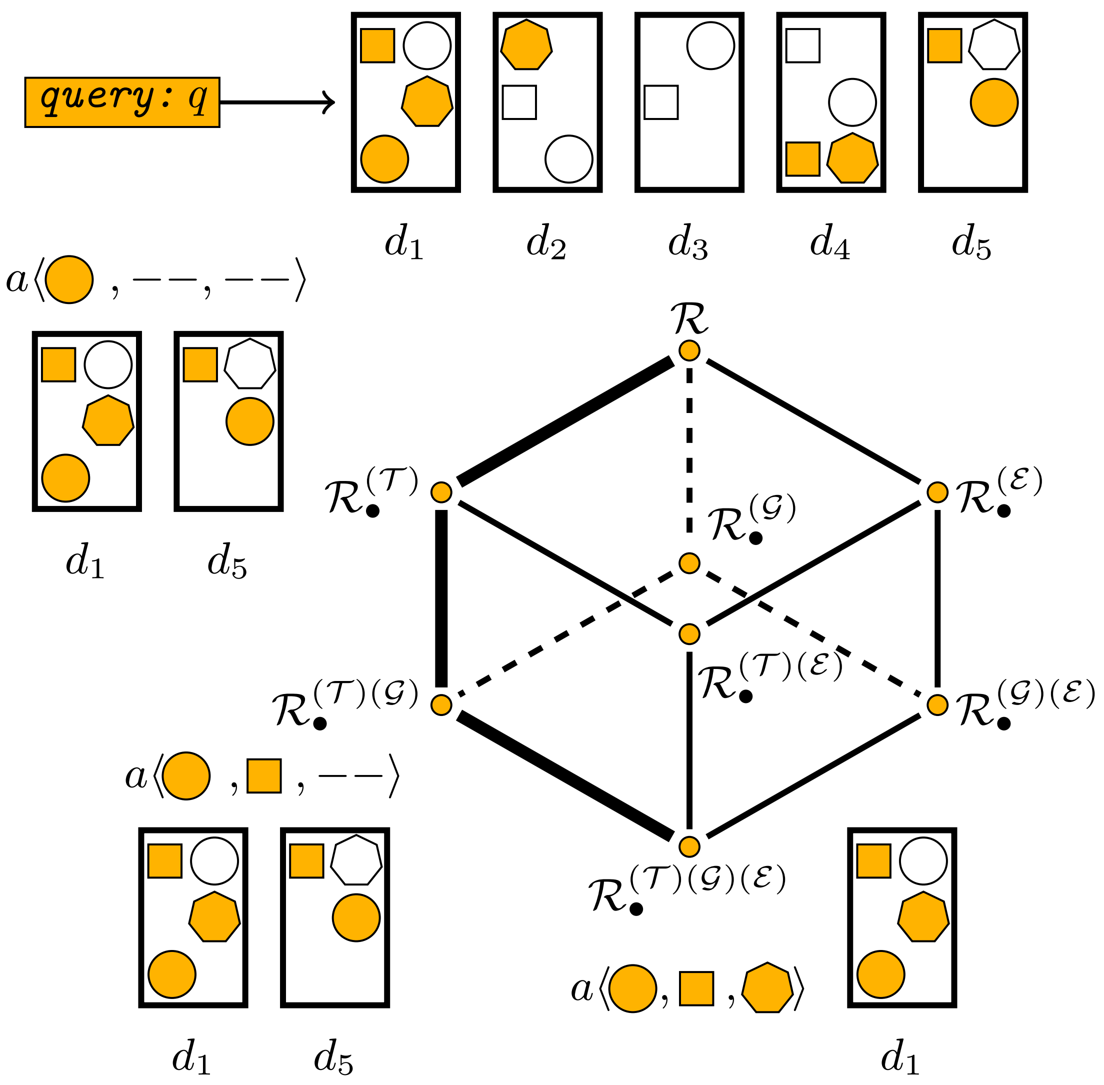

query: q
d_1
d_2
d_3
d_4
d_5
$a\langle\ ,--,--\rangle$
d_1
d_5
$\mathcal{R}$
$\mathcal{R}_\bullet^{(\mathcal{T})}$
$\mathcal{R}_\bullet^{(\mathcal{G})}$
$\mathcal{R}_\bullet^{(\mathcal{E})}$
$\mathcal{R}_\bullet^{(\mathcal{T})(\mathcal{E})}$
$\mathcal{R}_\bullet^{(\mathcal{T})(\mathcal{G})}$
$\mathcal{R}_\bullet^{(\mathcal{G})(\mathcal{E})}$
$a\langle\ ,\ ,--\rangle$
$\mathcal{R}_\bullet^{(\mathcal{T})(\mathcal{G})(\mathcal{E})}$
$a\langle\ ,\ ,\ \rangle$
d_1
d_5
d_1

Figure 9.1: The lattice structure for aspect generation by the xFactor algorithm is shown. Shapes in documents d represent annotation types. While colors represent different annotation values for same annotation type. Each element in the lattice corresponds to the partition of documents that arises by applying the factor method for that annotation type. For example, $\mathcal{R}_\bullet^{(\mathcal{T})}$ is generated by factoring $\mathcal{R}$ along time. The time factor $a = \langle \bullet, -, - \rangle$ is generated by documents $\{d_1, d_5\} \in \mathcal{R}_\bullet^{(\mathcal{T})}$. Continuing in this recursive manner over the geographic annotation type $\mathcal{G}$ we get $a = \langle \bullet, \blacksquare, - \rangle$. The sequence of factoring operations can be permuted to obtain different partitions; $\mathcal{R}_\bullet^{(\mathcal{T})(\mathcal{G})(\mathcal{E})}$ corresponds to time $\rightarrow$ geography $\rightarrow$ entity (traversing the bold edges).

Minimum Salience and Aspect Ranking. The xFactor algorithm is still computationally expensive if we were to consider every factor for each annotation type. To prune the recursion depth, we utilize a minimum salience criteria. For a given value of minimum salience $\sigma \in [0, 1]$, a factor is deemed salient if and only if: $\mathrm{s}(x, \mathcal{R}) \geq \sigma$. Using the salience we furthermore rank the aspects presented to the user as:

$$\mathrm{s}(a, d) = \prod_{x_i \in a} \mathrm{s}(x_i, d).$$

Algorithm 8: Generate Factors.

```
1  Function factor($\mathcal{X}, \mathcal{R}, \sigma$)
2      $\bigcup\langle x, \mathcal{R}'\rangle \leftarrow$ generate pairs of:  factors using the semantic
        model for $\mathcal{X}$ and the originating document partition.
3      factors $\leftarrow \varnothing$
4      foreach ($\langle x, \mathcal{R}'\rangle \in \bigcup\langle x, \mathcal{R}'\rangle$) do
5          if ($\mathrm{s}(x, \mathcal{R}') \geq \sigma$) then
6              factors.add($x$)
7              PartitionIndex.put($\langle x, \mathcal{R}'\rangle$)
8      return factors
```

Algorithm 9: The xFactor Algorithm.

```
1  Function xFactor($\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_n, \mathcal{R}, \sigma$)
       // The set of aspects to return
2      $\mathcal{A} \leftarrow \varnothing$
3      $x_n$Factors $\leftarrow$ factor ($\mathcal{X}_n, \mathcal{R}, \sigma$)
4      foreach ($x_n$Factor $\in x_n$Factors) do
5          $\mathcal{R}' \leftarrow$ PartitionIndex.get($x_n$Factor)
6          $x_{n-1}$Factors $\leftarrow$ factor($\mathcal{X}_{n-1}, \mathcal{R}', \sigma$)
7          ...
8          foreach ($x_2$Factor $\in x_2$Factors) do
9              $\mathcal{R}' \leftarrow$ PartitionIndex.get($x_2$Factor)
10             $x_1$Factors $\leftarrow$ factor($\mathcal{X}_1, \mathcal{R}', \sigma$)
               // Generate aspects
11             foreach ($x_1$Factor $\in x_1$Factors) do
12                 $\mathcal{A} \leftarrow \mathcal{A} \cup \langle x_1$Factor$, \ldots, x_n$Factor$\rangle$
13         ...
14     return $\mathcal{A}$
```

9.6 Properties of the xFactor Algorithm

Structured Representation of Documents. The aspects which are assimilated from multiple documents can be used to transform the semi-structured documents with annotations (i.e., $d = \{d_{\mathcal{E}}, d_{\mathcal{G}}, d_{\mathcal{T}}\}$) into a structured representation of aspects (i.e., $d = \langle a_1, a_2, \ldots, a_n \rangle$). This structured representation of documents using aspects is then immediately useful for applications in search tasks, such as result diversification. To represent documents using aspects, we can obtain the inverse mapping of documents to aspects by looking for all $a \in \mathcal{A}$ associated with a particular document d in the partition index.

Query Pivoting. If the order in which the annotation types are factored are permuted then the XFACTOR algorithm will produce different sets of aspects. This is because the factor methods rely on a given document partition to generate the factors. For instance, for three annotation types, we can realize six different sets of aspects by permutation of the different factor methods. This in turn provides us different ways of analyzing three different kinds of initial ambiguity underlying the query: temporal ambiguity, e.g., ***tokyo summer olympics***; geographical ambiguity, e.g., ***springfield***; and named entity related ambiguity, e.g., ***george bush***. If the sequence of factor methods is TIME $\rightarrow$ ENTITY $\rightarrow$ GEOGRAPHY, then the resulting set of aspects will be denoted by $\mathcal{A}_{\langle \mathcal{T},\mathcal{E},\mathcal{G} \rangle}$. The other five possibilities are: $\mathcal{A}_{\langle \mathcal{T},\mathcal{G},\mathcal{E} \rangle}$, $\mathcal{A}_{\langle \mathcal{G},\mathcal{T},\mathcal{E} \rangle}$, $\mathcal{A}_{\langle \mathcal{G},\mathcal{E},\mathcal{T} \rangle}$, $\mathcal{A}_{\langle \mathcal{E},\mathcal{T},\mathcal{G} \rangle}$, and $\mathcal{A}_{\langle \mathcal{E},\mathcal{G},\mathcal{T} \rangle}$. Using the illustration in Figure 9.1, these six factor sequences can be obtained by following different paths in the lattice.

Summarizing Entity Sets. For each of the generated aspects, we can summarize the resulting factors (e.g., named entities) using background knowledge (e.g., KG) into broader semantic classification types (e.g., categories from Wikipedia). For instance, in Table 9.1, we have summarized all the named entities into categories from Wikipedia. Concretely, to arrive at the types for named entities, we look up all the types that an entity belongs to (following RDFS:TYPE and RDFS:SUBCLASSOF links). Thereafter, we select the summary type as the one that covers most of the entities in the entity factor.

9.7 Evaluation

We next describe the setup and results of our experimental evaluation.

9.7.1 Annotated Document Collections

We test our algorithm on two different types of document collections. The first category of document collections consists of news articles. News archives have the benefit of being accompanied by rich metadata in the form of accurate publication dates and well-written text. This can aid NLP tools to provide accurate annotations. For example, temporal taggers can resolve relative temporal expressions (e.g., YESTERDAY) and implicit temporal expressions (e.g., GOOD FRIDAY) with respect to the publication date. We consider two document

Collection	#Documents	Avg. Time	Avg. Entities	Avg. Locations
New York Times	1,679,374	12.50	16.25	8.65
Stics	4,075,720	10.09	10.89	5.93
ClueWeb'09	50,220,423	30.59	8.23	9.49
ClueWeb'12	408,878,432	5.80	7.74	5.61

Table 9.2: Collection statistics.

collections in this category. One of them is a collection of approximately two million news articles published in the New York Times between 1987 and 2007. It is publicly available as the New York Times Annotated Corpus [18]. The other one is a collection of approximately four million news articles collected from various online sources during the period of 2013 to 2016, called Stics [117].

The second category of document collection consists of web pages. Web crawls unlike news articles have unreliable metadata and ill-formed language. This hampers us in obtaining high-quality semantic annotations for them. For example, we cannot resolve relative and underspecified temporal expressions, as the document creation time for Web pages may not reflect their true publication dates. We consider two web crawls [1, 2] from 2009 and 2012, which are publicly available as ClueWeb'09 and ClueWeb'12 document collections, respectively. Statistics for the document collections are summarized in Table 9.2.

Annotating Documents. Semantic annotations are central to our approach. To obtain them, we utilize publicly available annotations for the document collections or automatically generate them using NLP tools. For the news archives and for ClueWeb'09, we utilized Aida [118], which performs named entity recognition and disambiguation. Each disambiguated named entity is linked to its canonical entry in the Yago KG and Wikipedia. As a subset of these named entities, we can obtain geographic locations. We keep only those documents in the news archives and ClueWeb'09 where at least one disambiguated named entity occurs. For ClueWeb'12, we utilized the FACC annotations [77] provided by Google. The FACC annotations contain the offsets of high precision entities spotted in the web pages. Temporal expressions for all the document collections were obtained using the HeidelTime temporal tagger [188]. In Table 9.2, we additionally report the average counts of the three types of semantic annotations found in at most 10,000 documents retrieved for each query in our testbed.

9.7.2 Ground Truth Semantic Aspects and Queries

To evaluate our system, we extracted 5,122 aspects from Wikipedia. This was done considering their diversity along annotation types of time, locations, and other named entities for a set of twenty-five keyword queries. The broad topics of the aspects along with the specific keyword queries and the number of aspects generated are listed in Table 9.4.

Years	Description	Locations
2008 to 2016	*Usain Bolt won total of 9 Olympic medals during the Summer Olympic games in the years he was active.*	*Beijing, London, and Rio de Janeiro*
2004 to 2016	*Michael Phelps has won a record number of 23 gold medals at various Olympic games during his career.*	*Athens, Beijing, London, and Rio de Janeiro*

Table 9.3: An example table of events generating ground truth.

For each query, we constructed a set of ground-truth aspects by considering the table of events present on the Wikipedia page corresponding to the query [13, 42]. For the table, we considered each row consisting of time, locations, and other entities as an aspect. If no locations or entities were mentioned, we extracted them from the associated event page of the row, by running Aida on the introductory paragraph of the event's Wikipedia page. For instance, consider Table 9.3 as an example Wikipedia table for Olympic medalists. Treating each row as a ground truth aspect, we look for temporal expressions, e.g., $[2008, 2016]$ as a time factor; locations, e.g., Beijing, London, and Rio de Janeiro as a location factor; and other named entities, e.g., Usain Bolt as an entity factor. Similarly, for the second row in Table 9.3, the extracted aspect is: $\langle$[2004,2016] , {ATHENS, BEIJING, LONDON, RIO DE JANEIRO}, {MICHAEL PHELPS}$\rangle$. The testbed is publicly available at the following URL:

`http://resources.mpi-inf.mpg.de/dhgupta/data/eswc2019/`.

9.7.3 Measures

The two key characteristics for evaluating aspects are: their correctness with respect to a ground truth and their novelty with respect to other aspects in the set. These two characteristics taken together ensure that our aspect sets are meaningful and non-redundant. We next describe the measures of correctness and novelty.

Similarity computation between aspects is central to both the correctness and novelty. To compute the similarity between the two aspects, a (system generated) and b (ground truth), we consider their similarity dimension-wise:

$$\text{sim}(a,b) = \frac{1}{3}\left(\frac{|a_{[b,e]} \cap b_{[b,e]}|}{|a_{[b,e]}|} + \frac{|a_{\mathcal{E}} \cap b_{\mathcal{E}}|}{|a_{\mathcal{E}}|} + \frac{|a_{\mathcal{G}} \cap b_{\mathcal{G}}|}{|a_{\mathcal{G}}|}\right),$$

where, for temporal similarity we coarsen the time intervals at year granularity to make them comparable. The temporal overlaps are computed using the uncertainty-aware time model [41] by converting the time intervals to the four-tuple notation. While for the other two dimensions the similarity is akin to computing the overlap between bag-of-locations and bag-of-entities.

Entity - $\mathcal{A}_{\langle \mathcal{E},\bullet,\bullet\rangle}$: *nobel prize* [114] | *oscars* [1,167] | *space shuttle missions* [155] | *olympic medalists* [48] | *paralympic medalists* [24]
Location - $\mathcal{A}_{\langle \mathcal{G},\bullet,\bullet\rangle}$: *aircraft accidents* [513] | *avalanches* [56] | *epidemics* [211] | *famines* [133] | *genocides* [35] | *volcanic eruptions* [171] | *hailstorms* [39] | *landslides* [85] | *earthquakes* [39] | *nuclear accidents* [26] | *oil spills* [140] | *tsunamis* [88]
Time - $\mathcal{A}_{\langle \mathcal{T},\bullet,\bullet\rangle}$: *assassinations* [130] | *cold war* [81] | *corporate scandals* [44] | *proxy wars* [34] | *united states presidential elections* [57] | *terror attacks* [316] | *treaties* [1,057] | *wars* [359]

Table 9.4: Query categories with factor operation sequences and aspect counts (in brackets).

Note that the similarity computation is done with respect to the system generated aspect (a in the denominator). This is done to avoid matching (and thereby not rewarding) those system aspects a with a very large time interval, bag-of-locations or bag-of-entities, to every ground truth aspect (b).

Correctness. Given a set of aspects $\mathcal{A}$ generated by our algorithm for a query q and the set of aspects $\mathcal{B}$ corresponding to the ground truth derived from the Wikipedia page for the same query, correctness is given by:

$$\text{correctness}(\mathcal{A},\mathcal{B}) = \frac{1}{|\mathcal{A}|}\sum_{a\in\mathcal{A}}\frac{1}{|\mathcal{B}|}\sum_{b\in\mathcal{B}}\text{sim}(a,b).$$

Novelty for the set of aspects $\mathcal{A}$ can be intuitively thought of measuring the dissimilarity with respect to $\mathcal{A}$ itself:

$$\text{novelty}(\mathcal{A}) = \frac{1}{|\mathcal{A}|}\sum_{a\in\mathcal{A}}\frac{1}{|\mathcal{A}|}\sum_{(a'\in\mathcal{A}/\{a\})}\Big(1-\text{sim}(a,a')\Big).$$

We can additionally conform the correctness measure to the standard information retrieval measures such as **precision** and **recall** as follows:

$$\text{precision} = \frac{1}{|\mathcal{A}|}\sum_{a\in\mathcal{A}}\max_{b\in\mathcal{B}}\Big(\text{sim}(a,b)\Big) \quad\text{and}\quad \text{recall} = \frac{1}{|\mathcal{B}|}\sum_{b\in\mathcal{B}}\max_{a\in\mathcal{A}}\Big(\text{sim}(a,b)\Big).$$

9.7.4 Setup

Baselines and Systems. We consider two baselines to compare our proposed approach. As a naïve baseline, we treated each document in the pseudo-relevant set to represent an aspect. This is equivalent to presenting the user a ranked list of documents to satisfy her information need. The equivalent aspect for a document is constructed by considering the earliest and latest time point in the document as its time interval and bag-of-locations and bag-of-entities to represent the other two dimensions. As a second baseline, we consider

Latent Dirichlet Allocation (LDA) [47]. With this baseline, we want to cluster together those documents that are semantically similar using only text. Using LDA, we discover k topics from the pseudo-relevant set of documents. From each topic's partition of documents, we derive the corresponding semantic aspect by considering the earliest and latest time point in the partition as its time interval and bag-of-entities and bag-of-locations to represent the two remaining dimensions. We refer to this baseline as LDA-k. For the xFactor algorithm, we considered the specific sequence of factor operations that were deemed meaningful for that query (as shown in Table 9.4). For instance, since the query *earthquakes* is oriented towards locations we considered the factor sequence operations $\mathcal{A}_{\langle \mathcal{G},\mathcal{E},\mathcal{T}\rangle}$ and $\mathcal{A}_{\langle \mathcal{G},\mathcal{T},\mathcal{E}\rangle}$.

Parameters. For each query in Table 9.4, we retrieve at most 10,000 documents with disjunctive operator using Okapi BM25 as the retrieval method. We used the standard parameters, $b = 0.75$ and $k_1 = 1.20$, for its configuration. For the LDA baseline, we followed Griffiths and Steyvers [90] for setting its parameters. Specifically, β was set to 0.1 and α was set to ${}^{50}/_{|\text{topics}|}$. We considered three topic set sizes for LDA namely, $|\text{topics}| \in \{50, 100, 200\}$ and the same number of top-k documents for each topic, e.g., for $|\text{topics}| = 50$, we picked top-50 documents for each topic as its generating partition. For our method, the minimum salience was set to $\sigma = 0.001$.

9.7.5 Results for Quality

Results for News Archives. We first consider the results of the systems in terms of correctness and novelty as reported in Table 9.5. We additionally report the average number of aspects $\mu_{|\mathcal{A}|}$ for each system under comparison. Note that the Okapi BM25 baseline gives us an upper bound for the value of correctness that can be obtained against the ground-truth. As, ultimately we generate the LDA topics and aspect from this set of documents. For the New York Times collection, our method identifies the most correct aspects with respect to the ground truth as compared to the LDA baselines. Despite the observation that Okapi BM25 wins in terms of novelty by considering all pseudo-relevant documents, our method still achieves a high degree of novelty, thereby identifying the most non-redundant set of aspects and is able to partition the set of pseudo-relevant documents to the greatest degree. For the Stics news collection, our method outperforms the LDA baselines in terms of correctness and is close to the upper bound that can be achieved from the given set of pseudo-relevant documents. Okapi BM25 achieves a higher novelty value, however, the increase compared to our method is not significant. Observing both correctness and novelty our method excels in providing both relevant and non-redundant sets of aspects when compared to the LDA baselines which can only achieve high novelty.

	New York Times			Stics		
	$\mu_{\|\mathcal{A}\|}$	C	N	$\mu_{\|\mathcal{A}\|}$	C	N
BM25	3,096	**.0237**	**.4269**	3,797	**.0204**	**.4227**
LDA-50	50	.0067	.2634	50	.0102	.2910
LDA-100	100	.0053	.2107	100	.0080	.2404
LDA-200	200	.0046	.1497	200	.0063	.1639
xFactor	2,261	.0161	.4201	503	.0190	.3862

Table 9.5: Results for correctness (C) and novelty (N) on news archives.

	New York Times			Stics		
	$\mu_{\|\mathcal{A}\|}$	P	R	$\mu_{\|\mathcal{A}\|}$	P	R
BM25	3,096	.1577	**.2749**	3,797	.1414	**.2828**
LDA-50	50	.0471	.0160	50	.0634	.0311
LDA-100	100	.0432	.0102	100	.0483	.0207
LDA-200	200	.0400	.0070	200	.0456	.0129
xFactor	2,261	**.2804**	.1777	503	**.2400**	.1427

Table 9.6: Results for precision (P) and recall (R) on news archives.

Now, consider precision and recall for the systems, as reported in Table 9.6. For the New York Times, while considering precision, our system consists of more relevant aspects compared to the baselines. With respect to recall, the Okapi BM25 baseline wins by considering the entire pseudo-relevant set of documents. Note that the LDA baselines and our algorithm xFactor can not achieve this value as they discard many annotations, favoring precision over recall. Therefore, considering both precision and recall together, our system presents a balanced performance: high precision and recall. While the baselines achieve high recall only. For the Stics collection, when considering precision and recall, our method again shows significant improvements over the baselines. Thus, by taking all the four measures, correctness, novelty, precision, and recall, our method allows us to distill interesting aspects which can guide the user to navigate through a large number of documents.

Results for Web Collections. Web archives give us more challenging documents to test the effectiveness of our approach. Particularly, since they are not well-formed, they have a lower average number of annotations per document, the annotations in them are prone to more errors, and the size of the web archives is magnitudes larger than news archives. Hence, they present a challenging real-world scenario to test our methods. We first consider the results for the web archives when measuring correctness and novelty that are reported in Table 9.7. For ClueWeb'09, our method outperforms both baselines in terms of novelty. In particular, for correctness our method comes close to the upper bound established by Okapi BM25. For ClueWeb'12 our method

	ClueWeb'09			ClueWeb'12		
	$\mu_{\mid\mathcal{A}\mid}$	C	N	$\mu_{\mid\mathcal{A}\mid}$	C	N
BM25	9,580	**.0155**	.3958	9,742	**.0266**	**.4531**
LDA-50	50	.0123	.3275	50	.0177	.3478
LDA-100	100	.0092	.2880	100	.0126	.3025
LDA-200	200	.0064	.2441	200	.0097	.2554
xFactor	1,822	.0146	**.4239**	616	.0173	.4176

Table 9.7: Results for correctness (C) and novelty (N) on web collections.

	ClueWeb'09			ClueWeb'12		
	$\mu_{\mid\mathcal{A}\mid}$	P	R	$\mu_{\mid\mathcal{A}\mid}$	P	R
BM25	9,580	.1151	**.3595**	9,742	.1494	**.4018**
LDA-50	50	.0734	.0999	50	.0930	.1049
LDA-100	100	.0560	.0808	100	.0718	.0770
LDA-200	200	.0433	.0502	200	.0585	.0500
xFactor	1,822	**.2218**	.1880	616	**.2433**	.1648

Table 9.8: Results for precision (P) and recall (R) on web collections.

performs at par with baselines in terms of novelty. When considering the measures in isolation, for correctness the LDA baseline wins over our method and Okapi BM25 baseline has higher novelty than our method. However, when considering both correctness and novelty together, xFactor is consistent in providing more correct and novel aspects as opposed to the LDA baselines.

Next, we consider the second set of experimental results for web collections when measuring precision and recall that are reported in Table 9.8. For ClueWeb'09, our method in terms of precision and recall outperforms the LDA baselines significantly. For ClueWeb'12, our method outperforms both baselines with respect to precision. However, in terms of recall Okapi BM25 outperforms our method when considering all the pseudo-relevant documents. Despite of this, our method provides a balanced performance with high precision and moderate recall as compared to the baselines which have high recall but very low precision.

9.7.6 Results for Ranking

Results for News Archives. The results of precision and recall at $k = \{10, 25, 50\}$ for the news archives are shown in Table 9.9. As we can observe, the ranking provided by xFactor surpasses both the Okapi BM25 and LDA baselines in terms of precision for both the New York Times and Stics archives. While, our proposed algorithm provides a high level of recall for both the news archives when compared to LDA baseline.

	New York Times			**Stics**		
	P@10	P@25	P@50	P@10	P@25	P@50
BM25	.1874	.1863	.1909	.1380	.1394	.1461
LDA-50	.0453	.0464	.0471	.0592	.0590	.0634
LDA-100	.0426	.0431	.0430	.0487	.0481	.0476
LDA-200	.0396	.0397	.0400	.0440	.0434	.0440
xFactor	**.2283**	**.2450**	**.2366**	**.1690**	**.1869**	**.1740**
	R@10	R@25	R@50	R@10	R@25	R@50
BM25	**.2593**	**.2672**	**.2510**	**.2732**	**.2834**	**.2700**
LDA-50	.0172	.0148	.0133	.0310	.0294	.0270
LDA-100	.0130	.0104	.0092	.0216	.0192	.0174
LDA-200	.0101	.0075	.0062	.0148	.0126	.0113
xFactor	.1729	.1691	.1633	.1274	.1295	.1237

Table 9.9: Results for precision and recall at $k = \{10, 20, 50\}$ on news archives.

	ClueWeb'09			**ClueWeb'12**		
	P@10	P@25	P@50	P@10	P@25	P@50
BM25	.1111	.1183	.1217	.1436	.1451	.1521
LDA-50	.0792	.0770	.0734	.0982	.0961	.0930
LDA-100	.0714	.0634	.0601	.0876	.0828	.0756
LDA-200	.0591	.0526	.0485	.0697	.0644	.0602
xFactor	**.1771**	**.1909**	**.1919**	**.1999**	**.1990**	**.2018**
	R@10	R@25	R@50	R@10	R@25	R@50
BM25	**.3632**	**.3973**	**.3380**	**.3867**	**.4357**	**.3727**
LDA-50	.1000	.1036	.0894	.0963	.1153	.0940
LDA-100	.0864	.0856	.0750	.0722	.0832	.0681
LDA-200	.0524	.0571	.0450	.0446	.0534	.0421
xFactor	.1523	.1649	.1532	.1469	.1607	.1362

Table 9.10: Results for precision and recall at $k = \{10, 20, 50\}$ on web collections.

Results for Web Collections. The results of precision and recall at $k = \{10, 25, 50\}$ for the Web collections are shown in Table 9.10. Similar to the observations made in the case of news archives, we see that the aspects extracted by xFactor at increasing ranks result in higher precision than both the baselines. Our algorithm further provides high recall at increasing rank positions as compared to the LDA baselines, while the Okapi BM25, considering all the annotations in each document outperform our system and the LDA baseline.

Overall Summary

Our experiments on two large news archives show that annotations in the form of temporal expressions, locations, and other entities can be used to identify semantic aspects that are correct and novel for document exploration. On Web-scale corpora, where quality annotations are few, xFactor can also identify precise aspects for information consumption. Moreover, using annotation and co-occurrence salience, xFactor shows it can produce ranked list of highly-relevant aspects.

9.8 Conclusion

In this chapter, we discussed the xFactor algorithm that leverages semantic annotations such as temporal expressions, geographic locations, and other named entities to generate semantic aspects. The xFactor algorithm consists of factor methods that model the semantics of annotations in order to compute their salience. xFactor additionally considers the co-occurrence salience of annotations of different types to generate semantic aspects. Furthermore, the factor methods can be applied in different orders to disambiguate different types of ambiguities underlying the query and thereby identifying the most relevant set of semantic aspects for it. Our experiments on two types of document collections that include news archives and Web collections, show that the xFactor algorithm allows the user to navigate through messy unstructured text in a structured manner.

PART III

MINING ANNOTATED DOCUMENT COLLECTIONS

CHAPTER 10

IDENTIFYING TIME INTERVALS FOR KNOWLEDGE GRAPH FACTS

• Dhruv Gupta and Klaus Berberich. Identifying Time Intervals for Knowledge Graph Facts. In WWW 2018, pages 37–38.

10.1 Introduction

Journalists often face the challenging task of substantiating an investigative story with temporal facts in order to convey the validity of certain arguments. The tool of choice for many such scenarios are commercial search engines. The journalist often collects many documents using keyword queries, manually accesses and inspects documents for temporal facts and then compiles a spreadsheet of the observed dates. However, this manual process becomes impossible if the temporal evidence is spread over thousands of document and if multiple facts need to be verified.

In this work, we leverage temporal expressions in document contents to identify time intervals for temporal facts in knowledge graphs (KGs). A temporal fact is a sentence that connects an entity (person, organization, or location) via a temporal predicate to a temporal expression. Computational analysis of temporal expressions is challenging as they can be present at different granularities from precise timestamps to coarse level mentions of decades, e.g., `the 60s`. Furthermore, temporal expressions can be implicit and relative to dates mentioned elsewhere in the text, e.g., `last sunday`.

In short, our approach overcomes the above challenges as follows. First, we extract temporal facts using information extraction templates that rely on word sequences and temporal expressions. Second, we model the temporal expressions identified for the facts in a time model that understands uncertainty and incorporates temporal granularity. Third and finally, we aggregate the temporal evidence found and rank the time intervals that are of interest to the KG fact.

10.2 Related Work

Temporal information associated with KG facts is of high importance. A notable attempt in this direction was by Talukdar et al. [193]. Their method to identify the temporal scope of a KG fact counted frequency of publication dates associated with the documents containing its mention. For validating the temporal scopes of facts Gerber et al. [85] rely only on year granularity temporal expression tokens. Thus, they disregard the rich temporal information that is conveyed by implicit and relative temporal expressions. More recently, Kuzey et al. [137] aimed at leveraging the temporal information associated with facts in the Yago KG and temporal expressions associated with the mention of the fact in document collection for tagging "temponymns". However, all these approaches utilize a rather naïve notion of time, i.e., simple frequency of timestamps. Without modeling the inherent uncertainty and dynamics associated with the temporal expressions, these prior approaches can not identify concrete time intervals associated with KG facts.

10.3 Approach

We next define concrete definitions and steps of our approach.

Temporal Facts. A temporal fact spotted in a document collection consists of natural language representation of a KG fact with a co-occurring temporal expression. Formally,

$$\text{(F)ACT} \equiv \langle \text{(S)UBJECT}, \text{(P)REDICATE}, \text{(O)BJECT} \rangle [\text{TIME}]. \tag{10.1}$$

A natural language representation of a KG fact $\langle \mathsf{S}, \mathsf{P}, \mathsf{O} \rangle$ is obtained by looking at all possible surface forms of all three arguments of a fact from paraphrase and surface form dictionaries. Thus, to detect a KG fact we have to detect its equivalent natural language representation:

$$\mathsf{F} \equiv \Big\langle \big\{s_1, s_2, \ldots, s_m\big\}, \big\{p_1, p_2, \ldots, p_n\big\}, \big\{o_1, o_2, \ldots, o_l\big\} \Big\rangle. \tag{10.2}$$

Query Processing. For each surface form or paraphrase of the predicate in the natural language representation of the fact (Equation 10.2) we utilize an inverted index over phrases that detects the positional span of each mention of the named entity or predicate. By intersecting the document identifiers of the mentions we obtain a set of candidate documents that contain the surface forms of the arguments of the KG fact. We restrict ourselves to those documents in which the positional spans of the surface forms and predicate paraphrases that occur within a sentence of the document. We further distill this candidate set of sentences by considering only those sentences that contain a temporal expression that co-occurs with the other arguments of the fact. Thus, with this final candidate set of sentences $\mathcal{S}$ we have identified all sentences in which the fact is materialized in natural language and an evidence in the form of a temporal expression is present.

Time Model. As mentioned earlier, temporal expressions are challenging to analyze as they are present at different granularities and are inherently uncertain in nature. For instance, the temporal expression `the 60s` is highly uncertain as to which time interval it refers to: is it referring to $[1962, 1965]$ or $[1963, 1967]$? However, if we can model every possible time interval referred by such temporal expressions then we can rely on more precise evidence in form of fine granular temporal expressions (e.g., `1963 to 1965`) that can help us ascertain the time interval the fact may refer to. In order to model this uncertainty we rely on the uncertainty-aware time model proposed by Berberich et al. [41]. Each temporal expression T is modeled by a four-tuple that contains bounds to: when the time interval could have begun $[b_\ell, b_u]$ and when the time interval could have ended $[e_\ell, e_u]$. Thus, the temporal expression T is represented by the following four-tuple:

$$T = \langle b_\ell, b_u, e_\ell, e_u \rangle. \tag{10.3}$$

Temporal expressions found co-occurring with the natural language representation of KG facts are then compared in this time model to aggregate and identify the time interval of relevance for the fact.

Identifying Time Intervals. Our hypothesis is that the time scope of a KG fact F can be found by aggregating all the temporal expressions T, found to co-occur with its mention (Equation 10.2) in text, in the uncertainty-aware time model (Equation 10.3). Following our earlier work [93], we define the probability of a time interval being relevant to a given fact F as follows:

$$P([b, e] \mid \mathsf{F}) = \frac{1}{|\mathcal{S}|} \cdot \sum_{s \in \mathcal{S}} P([b, e] \mid s_{\text{time}}), \tag{10.4}$$

where, $s \in \mathcal{S}$ denotes a sentence containing the fact F and s_{time} denotes the temporal expressions associated with the sentence s. The likelihood of a time interval being relevant to the fact F given the sentence s is aggregated as (following [41]):

$$P([b, e] \mid s_{\text{time}}) = \frac{1}{\mid s_{\text{time}} \mid} \sum_{T \in s_{\text{time}}} \frac{\mathbb{1}([b, e] \in T)}{\mid T \mid}. \tag{10.5}$$

10.4 Evaluation

We next describe the evaluation setup for the experiments and a discussion of the obtained results.

Document Collections and Annotations. We utilized two news document collections that together comprise of more than eleven million documents. Sentence splitting and temporal expressions for all the documents in these collections were obtained by using the SUTime component of the Core NLP toolkit [151]. Each time annotation is normalized to a time interval in the following format: $[\text{(b)egin} = \text{yyyy-MM-dd}, \text{(e)nd} = \text{yyyy-MM-dd}]$. Furthermore, we translate these annotations into the uncertainty-aware time model as follows: $[b, e] \equiv \langle b, e, b, e \rangle$. Annotation and collection statistics are given in Table 10.1.

Knowledge Graph Facts. We utilize the temporal fact benchmark by Gerber et al. [85] for experimental evaluation. Since our approach is completely unsupervised in nature we considered all the positive examples in train and test splits in the benchmark. Furthermore, for reasons of efficiency and recall we disregard the paraphrases of the predicates and only consider spotting the surface forms of the subject and object within a sentence of a document. For the baseline we considered the time interval correct when the begin and end year of the temporal expression matched to that of the ground truth begin and end years. For our proposed method, we considered the time interval determined correct if the overlap between generated time interval and the ground truth time interval, which is at the year granularity, was over 75% with respect to the generated time interval.

Systems and Baselines. As a baseline, we consider a method that simply ranks the time annotations, when coarsened to year granularity, by its frequency. For each temporal fact in the benchmark, if the number of sentences matched exceeds a threshold we sample a small subset of sentences to execute our method and the baseline. Note that each sentence can contain more than one temporal expression. We set the threshold of the maximum number of evidences considered as 25.

Metrics. We evaluate to observe at what rank we can identify the correct time interval for a given fact. This is measured by the mean reciprocal rank (MRR). Furthermore, we evaluate if we can identify the correct ground truth time interval at rank 1, 5, and 10. That is, we measure precision at rank 1 (P@1), rank 5 (P@5), and rank 10 (P@10). We report averages over these metrics for the temporal facts in the benchmark.

COLLECTION	$n_{\text{documents}}$	$n_{\text{sentences}}$	$n_{\text{temporal expressions}}$
NEW YORK TIMES	1,855,623	54,024,146	15,411,681
GIGAWORD	9,870,655	181,386,746	72,247,124

Table 10.1: Document collection statistics.

	NEW YORK TIMES			
SYSTEM	**P@1**	**P@5**	**P@10**	**MRR**
BASELINE	0.21	0.12	0.12	0.25
PROPOSED METHOD	**0.26**	0.12	0.12	**0.30**

	GIGAWORD			
SYSTEM	**P@1**	**P@5**	**P@10**	**MRR**
BASELINE	0.24	0.13	0.12	0.29
PROPOSED METHOD	**0.34**	**0.16**	**0.15**	**0.39**

Table 10.2: Results obtained for different collections.

Results. The results for the two document collections are given in Table 10.2. For the New York Times document collection we were able to evaluate 526 temporal facts in the benchmark, while for the Gigaword collection we were able to evaluate 740 temporal facts out of 1,500 facts in the benchmark. The remaining facts could not be evaluated by our method since we could not spot sentences containing the entity surface forms for them in the document collections. From Table 10.2 we see that our method identifies the correct time interval for the facts at around rank three. Considering precision at one, our method outperforms the baseline for both collections. The results thus show that our method identifies the correct time intervals for facts by leveraging uncertainty and modeling complex temporal expressions at different granularity.

10.5 Conclusion

In this work, we described an approach to determine the time intervals of interest to temporal facts in knowledge graphs. To do so, our approach first rewrites the temporal facts into a natural language representation and retrieves text evidences containing temporal expressions for them. Our approach then leverages a time model that incorporates uncertainty to aggregate the temporal evidence. Using a probabilistic generative process, we can determine a ranked list of time intervals of relevance to the knowledge graph fact. Based on our evaluation we find that by carefully modeling the uncertainty behind temporal expressions we can determine time intervals to temporal facts with high precision.

CHAPTER 11

EVENTMINER: MINING EVENTS FROM ANNOTATED DOCUMENTS

- Dhruv Gupta, Jannik Strötgen, and Klaus Berberich. EventMiner: Mining Events from Annotated Documents. In ICTIR 2016, pages 261–270.

11.1 Introduction

Real-world events are signified by time, locations, and entities. These event dimensions also frequently occur in queries issued to commercial search engines. Concretely, 17.1% of Web queries are implicitly temporal [211]; 12.7% of the queries in Yahoo! query logs contain geographic locations [213]; and 71% of Web queries contain named entities [91]. Events as a collective representation of time, locations, and entities can thus be an important feature to navigate large document collections or the Web. An example of a textually realized event, is presented in Figure 11.1. With marked accuracy and scalability of natural language processing (NLP) tools that can provide semantic annotations in the form of temporal expressions, geographic locations, and named entities, scalable extraction of real-world events is now possible.

Cold war was a conflict involving <u>Soviet Union</u>$^{\langle \text{GEO:Soviet\_Union} \rangle}$ and the <u>US</u>$^{\langle \text{GEO:USA} \rangle}$ under the presidency of <u>Mikhail Gorbachev</u>$^{\langle \text{WIKI:Mikhail\_Gorbachev} \rangle}$ and <u>Ronald Reagan</u>$^{\langle \text{WIKI:Ronald\_Reagan} \rangle}$ respectively, during late <u>1980s</u>$^{\langle 01-01-1985,31-12-1989 \rangle}$.

Figure 11.1: Sample text with semantic annotations.

Understanding the semantics underlying the annotations of time, locations, and entities is essential to detect events. Temporal expressions can be highly uncertain (e.g., `late 1980s`) and vague (e.g., `last spring`). Mentions of locations in text can also be highly ambiguous and thus difficult to anchor on a map (e.g., SPRINGFIELD). A document may describe events about a single entity in entirety or may shift its focus between different events concerning related entities. Thus, to determine the importance of events from documents, we have to carefully model and consider the semantics of the annotations in text. Prior approaches such as [30, 138, 175, 187] to mine events disregard annotation semantics when mining events from text.

In this chapter, we describe EVENTMINER, a probabilistic framework which leverages text and semantic annotations in the form of temporal expressions, geographic locations, and other named entities to mine events. EVENTMINER is based on a family of Dirichlet process mixture models to cluster semantically annotated text to mine events. The notion of importance in our algorithm is measured by statistical frequency. In order to analyze these annotations, we present mathematical models for notions of time, locations, and entities that can aid in computing cohesive clusters. Concretely, our model of time takes into account temporal uncertainty as well as temporal proximity in order to identify events that might occur very close to each other on a timeline (e.g., 1990 is close to 1991). We model geographic locations so that events happening at locations that neighbor each other are also captured (e.g., USA neighbors CANADA). We take into account similarity behind named entities (e.g., RONALD REAGAN is related to NANCY REAGAN) when identifying important events. We evaluate our methods by comparing the summary of events computed for history-oriented queries [95] with their Wikipedia page.

Applications. In the context of digital humanities, there is a need for tools to analyze large text collections. Given keyword only queries related to (historical) entities or events, EVENTMINER is able to mine these important events which can be further used to explore the retrieved documents.

Outline. The remainder of the chapter is structured as follows. We first survey the related work in Section 11.2. We then formally define the problem setting in Section 11.3. We describe computational models for annotations in Section 11.4 and EVENTMINER in Section 11.5. Our evaluation setup and experimental findings are given in Section 11.6. Conclusions drawn from the study are summarized in Section 11.7.

11.2 Related Work

In this section, we describe prior work with respect to our problem setting.

Related Models. Prior work [174, 215] are examples of clustering algorithms that compute document similarity based on text and contextual temporal expressions. Both have leveraged the framework of Dirichlet process mixture (DPM) models. Zhu et al. [215] present a time-sensitive Dirichlet process mixture model. The authors consider the use case of organizing an email inbox by using the arrival time of these emails. For computing similarity between different instances, they utilize an exponential decay function which gives more weight to recent emails. Building on this work, Qamra et al. [174] develop a content-community-time model that tries to identify similar blogs in a community of blogosphere. While utilizing the framework of DPM models, we expand on the notion of temporal similarity by taking into account uncertainty and proximity. Additionally, we incorporate similarity along the geographic dimension and also similarity between named entities in an ontology.

Temporal Information Retrieval is a sub-field of IR with special emphasis on temporal information present in documents in the form of metadata (e.g., publication dates or creation dates) and temporal expressions [53]. Anchoring documents or queries in time is an integral part of a time-sensitive search engine. Jatowt et al. [123] address the problem of estimating the time period which the document focuses on. To do so, they construct a weighted undirected graph which captures the associations between terms and time. Gupta and Berberich [93] address the problem of providing interesting time intervals to a keyword query by using temporal expressions in pseudo- relevant document set. Building on this they can also classify keyword queries to determine if it's: i. (a)temporal ii. temporally unambiguous iii. temporal ambiguity at different granularities (e.g., year, month, or day) iv. temporally (a)periodic [94].

Timeline Generation. One of the initial works for automatic timeline generation was by Swan and Allan [191]. To generate timelines their model used relative frequency of important features such as named entities and noun phrases. The method involved calculating the frequency of the features at different time points and capturing the significance of its frequency by computing χ^2 statistic. In [30], the authors specifically pay attention to the temporal expressions in the documents for constructing temporal document profiles. These profiles are then used for clustering and re-ranking of documents.

Event Detection. A large body of work exists in analyzing different kinds of semantic annotations in isolation. However, we address the interplay between different kinds of semantic annotations, which in the past has received markedly little attention. Topic detection and tracking (TDT) [28] was a seminal initiative in event extraction. TDT tasks focused on organizing an incoming stream of text first into topics, composed of stories describing events. However, initial approaches described in [28] did not have access to reliable NLP annotators. In light of these advancements in NLP, several works have incorporated semantic annotations for event extraction [24, 124, 138, 175].

Kuzey et al. [138] address the task of event extraction from news archives for ontology population. To do so, they construct a multi-view attribute graph using: document text, publication date, and named entities (e.g., people, organization, and location) in documents with their semantic types (e.g., protest, hurricane). They then mine the multi-view attribute graph for important events. [24] mines events from annotated corpora by utilizing frequent itemset mining. Both approaches disregard any special treatment for time and geographic location; which has been adequately addressed in our work.

Focusing specifically on predicting the future, Radinsky et al. [175] developed the *Pundit* algorithm. Events are modeled as a tuple of state, actor, objects, instrument, location, and time. Pundit predicts future events by performing hierarchical agglomerative clustering on events. The similarity between events is computed by using distances in a semantic network. The aim is to derive future events via causality in events that have already occurred. Utilizing only temporal data, Jatowt and Yeung [124] also present a model-based clustering algorithm for predicting future events. They capture the inherent uncertainty in temporal expressions by modeling them as probability distributions. The model-based clustering subsequently derives the similarity between these distributions using the Kullback-Leibler (KL) divergence.

Yeung and Jatowt [207] study how topics change with time. They analyze the Google news archive for a twenty year time period (1990-2010) for thirty two different countries. For analysis, they extract temporal expressions and also topics by using Latent Dirichlet Allocation. They then study the distributions of temporal expressions that refer to the past. Specifically, they look at how topics change over time, what caused the re-collection of past events, how events were forgotten over time, and how countries are similar with respect to their topic distribution over time.

Event-Centric Search. [180, 187, 189] are example works that leverage semantic annotations for document clustering and exploration. All these approaches however do not make use of ontological named entities in form of person or organization. Strötgen and Gertz [187] present methods for event extraction using using only temporal expressions and geographic locations. They model an event as a co-occurrence of temporal expression and geographic location. Building on this notion of event they provide a extended Backus-Norm-Form (EBNF) query language for event exploration. Another work by Strötgen and Gertz [189] re-ranks documents for a given query by computing similarity and proximity with respect to text, time, and geography. Samet et al. [180] discuss *NewsStand*, a system that allows users to find news anchored by its location on a map. *NewsStand* does this by detecting and resolving toponyms. Their method makes use of a streaming clustering algorithm. Some of the features used for ranking the clusters are the size of the clusters, number of news sources, cluster's rate of propagation, and its timestamp.

11.3 Preliminaries

We next describe the annotations we use; how we pre-process and annotate document collections; and we also formalize the event extraction problem.

11.3.1 Semantic Annotations

Consider Figure 11.1 as an illustrative example for the following discussion.

Temporal Expressions

Temporal expressions are highly ambiguous in nature. They can be categorized as explicit, implicit, relative, and underspecified [53, 97, 188]. An explicit temporal expression in Figure 1 is `late 1980s`. As with all temporal expressions, explicit temporal expressions can be present at different granularities, e.g., `May 5, 2001` or `1992`. Implicit temporal expressions may not be immediately identifiable as they are characterized by words that carry a latent temporal meaning, e.g., `Christmas`. Words whose temporal meaning can only be resolved with respect to some other time point (e.g., the publication date) are known as relative (e.g., `yesterday`) or underspecified if the relation to the reference time has to be determined additionally (e.g., `April`). NLP tools that can extract and normalize such types of temporal expressions are HeidelTime [188] and SUTime [58] which we used in this work to resolve temporal expressions.

Geographic Locations and Other Named Entities

We use Aida [118] to identify, disambiguate, and link named entities in text documents to an external ontology. Aida performs named entity recognition and disambiguation by leveraging statistical popularity of named entities and contextual similarity to disambiguate them. Examples of named entities in Figure 11.1 are MIKHAIL GORBACHEV and RONALD REAGAN which have been disambiguated and linked to the Yago ontology [190]. Geographic locations mentioned in text are called toponyms [180]. The process of resolving these toponyms to a specific location is known as toponym resolution. We use the geographical locations obtained as part of the detected and disambiguated named entities by Aida. Examples of disambiguated locations in Figure 11.1 are SOVIET UNION and US.

Document Collection

We used the New York Times Annotated corpus [18] which consists of around two million news articles published between 1987 and 2007. We utilized a news archive for evaluating our methods since it has been shown in prior work [95, 124] that they cover historic events very well. All the annotations along with text are preprocessed using the Hadoop map-reduce framework and subsequently indexed using the ElasticSearch [4] software.

11.3.2 Problem Statement

Consider a document collection D consisting of N documents d:

$$D = \{d_1, d_2, \ldots d_N\}$$

further each document $d \in D$ consists of sentences s:

$$d = \langle s_1, s_2, \ldots s_n \rangle$$

Each sentence s then contains a multiset of temporal expressions $s_{\mathcal{T}}$, geographic locations $s_{\mathcal{G}}$, named entities $s_{\mathcal{E}}$, and words $s_{\mathcal{W}}$ from a vocabulary $\mathcal{V}$:

$$s = \langle s_{\mathcal{T}}, s_{\mathcal{G}}, s_{\mathcal{E}}, s_{\mathcal{W}} \rangle.$$

The cardinalities of these multisets are given by $|s_{\mathcal{T}}|$, $|s_{\mathcal{G}}|$, $|s_{\mathcal{E}}|$, and $|s_{\mathcal{W}}|$. The aim is to design an algorithm:

$$\textsc{EventMiner}(S, Q, \Lambda),$$

where, S is a set of input sentences, Q is a keyword query, and Λ consists of a set of parameters $\Lambda \in \mathbb{R}^m$. The input set of sentences S is obtained from the pseudo-relevant set of documents R obtained via an information retrieval engine using the keyword query Q (following the notational convention from [55]):

$$R = \textsc{Ir}(D, Q, K, \Theta),$$

where, Θ is set of parameters and K specifies the number of documents to be returned by the retrieval method. The algorithm should output a totally ordered set of events:

$$\mathcal{C} = \langle c_1, c_2, \ldots c_k \rangle,$$

where, c_i is an event. The ordering of events in $\mathcal{C}$ is done by using the scores obtained for each cluster given by the EventMiner algorithm. Using $\mathcal{C}$ we re-rank R to obtain a set of documents $\hat{R}$ so that the user sees at least one document from each event $c \in \mathcal{C}$.

An *event* that can be detected in text, is defined to involve related named entities $c_{\mathcal{E}}$, occurring during a time interval $[b, e] \in c_{\mathcal{T}}$ at related locations $g \in c_{\mathcal{G}}$, and described by words $c_{\mathcal{W}}$. Thus each event $c \in \mathcal{C}$ can be modeled as:

$$c = \langle c_{\mathcal{T}}, c_{\mathcal{G}}, c_{\mathcal{E}}, c_{\mathcal{W}} \rangle.$$

We hypothesize that using events as proxies for user intents we can improve the retrieval effectiveness of traditional information retrieval methods, which have largely relied on term statistics [95].

11.3.3 Assumptions

We make three assumptions while designing our EVENTMINER algorithm. First, each semantic annotation occurs independent of each other. Hence, we can consider a sentence to contain a multiset of temporal expressions $s_{\mathcal{T}}$, geographic locations $s_{\mathcal{G}}$, and named entities $s_{\mathcal{E}}$. Second, a multiset of geographic locations $s_{\mathcal{G}}$ or named entities $s_{\mathcal{E}}$ can be empty. However, they cannot be empty simultaneously in a sentence. A multiset of temporal expressions $s_{\mathcal{T}}$ in a sentence cannot be empty. In case no temporal expression occurs in a sentence we utilize the document publication date. Third, geographic annotations are a subset of named entity annotations, that is $s_{\mathcal{G}} \subseteq s_{\mathcal{E}}$.

11.4 Computational Models

Here we describe the computational models used to represent and compute similarities between annotations for time, geographic locations, and named entities.

11.4.1 Time Models

We next explain the two time models that we use in the EVENTMINER algorithm. Figure 11.2 gives an illustrative explanation of these models.

Uncertainty-aware Time Model (UTM)

UTM is a time model for uncertain temporal expressions [41], for instance `1990s`, where begin and end of the time interval $[b, e]$ cannot be clearly identified. UTM models this uncertainty by allowing for a lower and upper bound in the start (end) of the interval. Formally, a temporal expressions is modeled as a four-tuple:

$$t = \langle b_\ell, b_u, e_\ell, e_u \rangle,$$

where, $b \leq e$, $b_\ell \leq b \leq b_u$ and $e_\ell \leq e \leq e_u$. The elements of t are obtained from a time domain $\mathbb{T}$. Thus, $[b, e] \in \mathbb{T} \times \mathbb{T}$. The number of time intervals that can be generated from t is denoted by $|t|$. Using this model, we can represent `1990s` as $\langle 1990, 1999, 1990, 1999 \rangle$. The probability of generating a time interval $[b, e]$ from temporal expression t is estimated as [41]:

$$P([b, e]|t) = \frac{\mathbb{1}([b, e] \in t)}{|t|}.$$

Thus, the likelihood of generating t' from t is:

$$P(t'|t) = \frac{1}{|t'|} \sum_{[b,e] \in t'} P([b, e]|t). \tag{11.1}$$

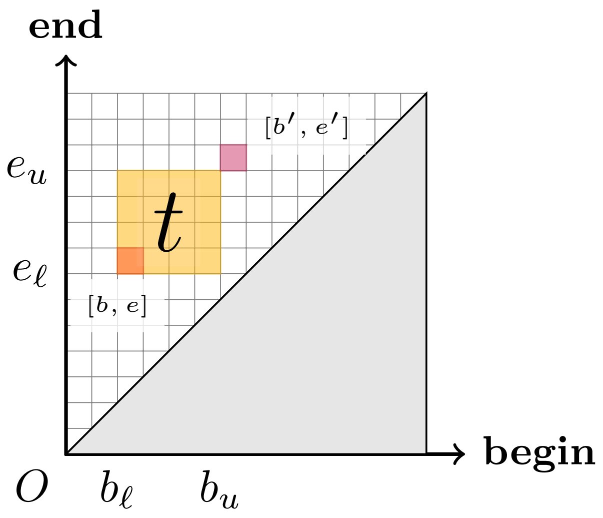


Figure 11.2: Graphical representation illustrating how a time interval $[b, e]$ is generated from t using UTM. It also represents graphically how the time interval $[b', e']$ obtains zero probability from UTM but a non-zero probability using PTM.

Following the uncertainty-aware time model, we can generate the time interval $[1995, 1998]$ from the temporal expression ⟨1990, 1999, 1990, 1999⟩. However, a proximate time interval $[2000, 2001]$ will receive zero probability given the same temporal expression.

Proximity-aware Time Model (PTM)

In UTM a time interval $[b, e] \notin t$ obtains zero probability. However, time intervals that are temporally close to time intervals in t should obtain non-zero probability [189]. This is required for computing similarity between events that have close but non-overlapping time intervals of occurrence (e.g., 1990 and 1991). We compute the proximity by multivariate kernel density estimates. Concretely,

$$P([b, e]|t) = \frac{1}{|t|} \sum_{[b', e'] \in t} \mathcal{K}_A([b, e] - [b', e']),$$

where, $\mathcal{K}_A$ is a multivariate kernel estimator with bandwidth matrix A. The difference between time intervals is carried out element-wise, i.e., $[b, e] - [b', e'] = [b - b', e - e']$. The kernel density estimator $\mathcal{K}_A$ is defined as [110]:

$$\mathcal{K}_A(\bullet) = \frac{1}{|A|} \mathcal{K}\left(A^{-1} \bullet\right),$$

where, the bandwidth matrix A is:

$$A_{2\times 2} = k \cdot I_{2\times 2} = \begin{bmatrix} k & 0 \\ 0 & k \end{bmatrix},$$

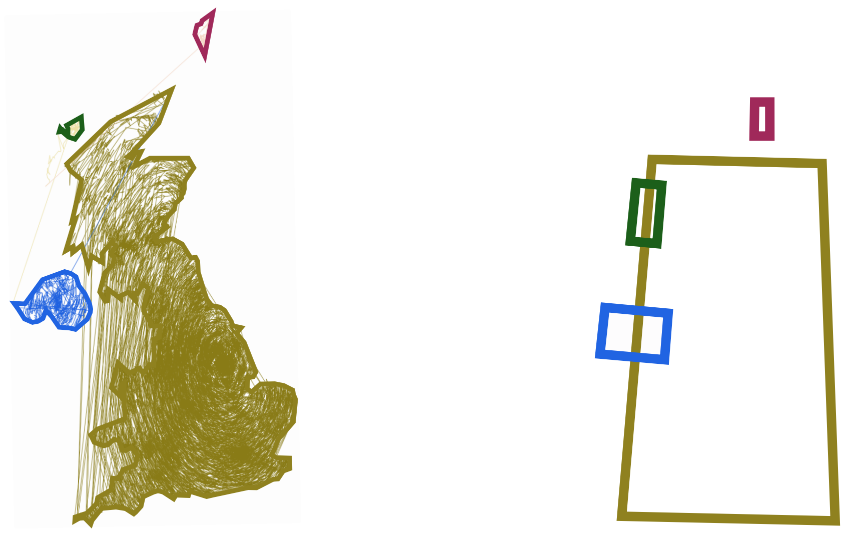

(a) Plot of coordinates of UK. (b) Plot of the MBRs of UK.

Figure 11.3: Depiction of how MBRs are computed using a set of geographic coordinates. For each dense region of coordinates we compute a MBR.

where, $|A|$ represents the matrix determinant and A^{-1} its inverse. Proximity between t' and t can be estimated by:

$$P(t'|t) = \frac{1}{|t'|} \sum_{[b,e] \in t'} \frac{1}{|t|} \sum_{[b',e'] \in t} \mathcal{K}_A([b,e] - [b',e']). \tag{11.2}$$

For $\mathcal{K}_A(\bullet)$, we utilized the Epanechnikov kernel, since its support is $[-1, 1]$ and density at a point is computed using the values lying in the cube surrounding it [110]. This kernel can be written as [110]:

$$\mathcal{K}_A(v) = \frac{3}{4}(1 - v^T v)\mathbb{1}(|\sqrt{v^T v}| \leq 1),$$

where, $v = [b, e]$ is a time interval and the indicator function $\mathbb{1}(\bullet)$ evaluates to one if and only if the argument is true. The proximity between two temporal expressions can be varied by scaling k in the bandwidth matrix A. The proximity model thus allows us to associate a non-zero probability with non-overlapping temporal expressions. Therefore, the time interval [2000, 2001] can now be generated from the temporal expression $\langle 1990, 1999, 1990, 1999 \rangle$.

11.4.2 Space Model

Each geographic location g in our model is represented as a minimum bounding rectangle (MBR) with coordinates for its lowermost coordinate ℓ and uppermost coordinate u: $g = \langle \ell, u \rangle$. Each coordinate lies in a two dimensional geodesic space, that is $\ell = (x_\ell, y_\ell)$ and $u = (x_u, y_u)$. For any two geographic locations described by their MBRs we can find similarity by computing the area overlap in the geodesic space. Similarly proximity can be found by their closeness in the geodesic space. Using the geodesic system to represent MBRs helps in avoiding distortion along the poles. Figure 11.3 shows how a MBR for UNITED KINGDOM (UK) is obtained from its set of coordinates. For an efficient indexing and querying of geographic locations, we utilize a R-Tree [108]. Each location's MBRs are indexed in an R-Tree. We can subsequently query the R-Tree for containment or proximity queries.

11.4.3 Entity Model

Each entity disambiguated by Aida [118] is linked to the Yago [190] ontology which in turn linked to its Wikipedia entry. To compute the similarity between two entities e and e' we thus look at the relatedness in terms of Wikipedia links. This similarity is known as Milne-Witten entity-entity relatedness: $\textsf{MWSim}(e, e')$ [158]. Formally, with W_e and $W_{e'}$ being the sets of articles that are linked to the Wikipedia articles corresponding to the entities e and e', respectively. Let W represent all articles in Wikipedia; the similarity is computed as [158]:

$$\textsf{MWSim}(e, e') = \frac{log(max\,(|W_e|, |W_{e'}|)) - log\,(|W_e \cap W_{e'}|)}{log\,(|W|) - log\,(min\,(|W_e, W_{e'}|))}$$

11.5 EventMiner Algorithm

The EVENTMINER algorithm is a probabilistic model that takes into account the similarity between the semantic annotations in order to mine important events. It is based on the family of Dirichlet process mixture (DPM) models [31, 174, 215]. The rationale behind using DPM models is two-fold. First, they allow us to jointly model the marginal distributions underlying the event annotations. Second, they allow to model an infinite number of clusters without knowing their number apriori. Each cluster identified by EVENTMINER is treated as an important event for the given keyword query. We next describe how the various semantic similarities are considered together and how to perform inference over the probabilistic model.

Generating Sentences. Let $\mathcal{V}$ be the vocabulary associated with the document collection. Further, let the event clusters that are identified by EVENTMINER be described by a multinomial θ_c distributed over $\mathcal{V}$. We describe the probability of generating the sentence s_i given θ_c as [174, 215]:

$$P(s_i|\theta_c) = \prod_{v \in \mathcal{V}} \theta_c(v)^{\mathrm{tf}(v, s_i)},$$

where, the probability of obtaining the term v from cluster c is denoted by $\theta_c(v)$ and the term frequency of v in sentence s_i is denoted by $\mathrm{tf}(v, s_i)$. Given a term distribution over sentences, we now consider the similarity between the various semantic annotations.

Temporal Similarity. To compute the temporal similarity of a sentence s to an existing existing event cluster c, we consider the similarity between their temporal expressions $s_{\mathcal{T}}$ $c_{\mathcal{T}}$. This is done by considering the temporal similarity of $t \in s_{\mathcal{T}}$ with all the temporal expressions $t' \in c_{\mathcal{T}}$ in the event cluster c as:

$$w_t(s, c) = \frac{1}{|c_{\mathcal{T}}|} \sum_{t' \in c_{\mathcal{T}}} \frac{1}{|s_{\mathcal{T}}|} \sum_{t \in s_{\mathcal{T}}} \mathbb{1}(t \in t'),$$

where, the function $\mathbb{1}(t \in t')$ indicates likelihood of generating the temporal expression t from t' by using either uncertainty-aware time model (Equation (11.1)) or proximity-aware time model (Equation (11.2)).

Geographic Similarity. Similarly, given an event cluster c, to consider the similarity of a sentence along the geographical dimension we compute:

$$w_g(s, c) = \frac{1}{|c_{\mathcal{G}}|} \sum_{g' \in c_{\mathcal{G}}} \frac{1}{|s_{\mathcal{G}}|} \sum_{g \in s_{\mathcal{G}}} \mathbb{1}(g \in g'),$$

where, the function $\mathbb{1}(g \in g')$ indicates the likelihood of generating the geographic location g from the geographic location g'.

Entity Similarity. On similar lines, we can compute the similarity between named entities in a sentence with an event cluster c as follows:

$$w_e(s, c) = \frac{1}{|c_{\mathcal{E}}|} \sum_{e' \in c_{\mathcal{E}}} \frac{1}{|s_{\mathcal{E}}|} \sum_{e \in s_{\mathcal{E}}} \mathbb{1}(e \sim e'),$$

where, the function $\mathbb{1}(e \sim e')$ computes the relatedness between the entities e and e' using MwSim(●). The motivation for computing the average entity-entity relatedness (as compared to maximum entity-entity relatedness) between sentence s and cluster c is to maintain the cluster coherence.

Joint Similarity. We combine all the semantic similarities in a weighted average. This can be written as:

$$w(s, c) = \frac{\rho_1 \cdot w_t(s, c) + \rho_2 \cdot w_g(s, c) + \rho_3 \cdot w_e(s, c)}{\rho_1 + \rho_2 + \rho_3}.$$

Chinese Restaurant Process. To incorporate the different semantic similarities in the Dirichlet process mixture model framework, we need to consider the clustering behavior of Dirichlet processes [194]. We briefly explain the framework next, following its description in [194, 196]. Consider a Dirichlet process (DP), $G_0 \sim DP(\alpha, H_0)$, with a prior (base) distribution H_0 [194, 215]. Further, let the prior follow a Dirichlet distribution, $H_0 \sim Dir(\beta m)$; where, m is the normalized term frequency vector over $\mathcal{V}$ and β is a hyperparameter [120, 148, 194, 215]. Let, the sample multinomial distributions be drawn iid from G_0 be $\theta_1, \theta_2, \ldots, \theta_i$. For the predictive distribution θ_{i+1} it has been shown in [46, 194, 196] that:

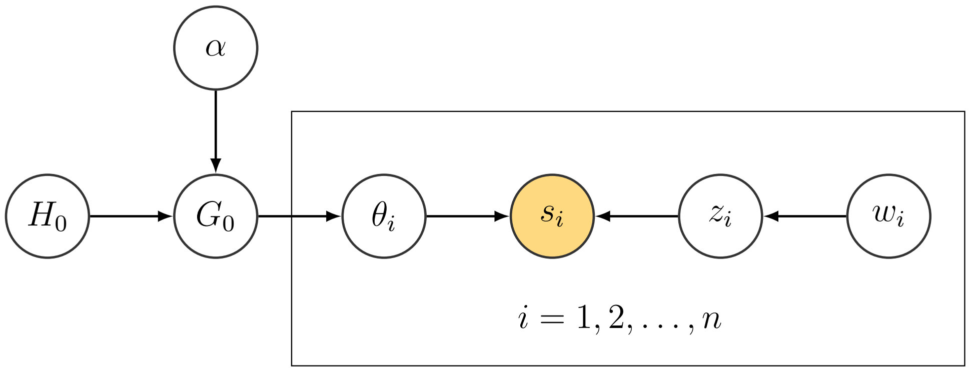


Figure 11.4: Graphical representation of the EventMiner algorithm based on [215]. It shows a Dirichlet process G with concentration parameter α and its base measure H_0. Where, a sentence is denoted by s, its cluster label is denoted by z, and the joint similarity incorporating time, geographic location, and named entities is denoted by w. The duplicated random variables are represented in the box; with i denoting the multiplicity. Observed node is shaded in yellow.

$$\theta_{i+1}|\theta_1, \ldots, \theta_i, \alpha, H_0 \sim \sum_{\ell=1}^{i} \frac{1}{i+\alpha}\delta_{\theta_\ell} + \frac{\alpha}{i+\alpha}H_0,$$

where, δ_{θ_ℓ} is a Dirac delta at θ_ℓ. From the equation above, we see that if a sample is drawn more than once, it shall have a higher probability of being drawn again leading to a “positive reinforcement effect” or equivalently “rich gets richer effect” [120, 194, 196]. This phenomenon is more often described in the literature as the Chinese Restaurant Process: consider unlimited tables to sit at a Chinese restaurant; a new customer will sit at a table in the restaurant with probability that is dependent on the count of customers already occupying that table [194]. Following this framework, we can incorporate the probability of assigning sentence s_i to cluster c, given all the other event cluster assignments as follows:

$$P(z_i = c|z_{-i}) = \begin{cases} \frac{w(s_i,c)}{\sum_{c'} w(s_i,c')+\alpha}, & \text{if } c \text{ is in cluster set} \\ \frac{\alpha}{\sum_{c'} w(s_i,c')+\alpha}, & \text{if } c \text{ is a new cluster} \end{cases}$$

where, α is the concentration parameter.

Inference. To infer the cluster label z given the semantically annotated text, we use the following inference:

$$P(z_i = c|z_{-c}, S) \propto P(z_i = c|z_{-i})P(s_i|s_{-i} \in z_i = c),$$

where, z_{-c} is used to denote all the cluster assignments except z_c, and $s_{-i} \in z_i = c$ denotes all the sentences in cluster c except s_{-i} [174, 215]. Since each type of expression is associated with a multiset, the order, in which semantic annotations are observed, can safely be ignored. Thus the first term, $P(z_i|z_{-i})$, can be computed by taking z_i as last in the order [174, 194].

The second term, $P(s_i|s_{-i} \in z_i = c)$, can be computed with the help of the Dirichlet process described earlier. It has shown to be equal to [148, 174, 215]:

$$\begin{aligned} P(s_i|s_{-i} \in z_i = c) &= \int p(s_i|\theta)p(\theta|s_{-i} \in z_i = c)d\theta \\ &= \left(\frac{\Gamma(\sum_v tf(v, s_{-i} \in z_i = c) + \beta)}{\prod_v \Gamma(tf(v, s_{-i} \in z_i = c) + \beta m_v)} \right) \\ &\left(\frac{\prod_v \Gamma(tf(v, s_i) + tf(v, s_{-i} \in z_i = c) + \beta m_v)}{\Gamma(\sum_v tf(v, s_i) + \sum_v tf(v, s_{-i} \in z_i = c) + \beta)} \right) \end{aligned}$$

where, the Gamma function is denoted by $\Gamma(\bullet)$, the term frequency of v in the set of sentences (excluding s_i) in cluster c is given by $tf(v, s_{-i} \in z_i = c)$, and the term frequency of v in sentence s_i is given by $tf(v, s_i)$.

The graphical model corresponding to our EventMiner algorithm, is illustrated in Figure 11.4. We utilized a modified Gibbs Sampler based on [215] for a single machine implementation. Similar to their implementation, our modified Gibbs Sampler also has time complexity of $\mathcal{O}(|S|^2)$.

Exploring Search Results with Event-Clusters. For each event-cluster c we can associate the documents from which the sentences were derived. This can thus be used to explore the search results in an event-centric manner.

11.6 Evaluation

In this section we first describe the experimental setup and then discuss the results of the experiments.

11.6.1 Setup

History-Oriented Queries. In order to test the effectiveness of our method, we utilize history-oriented queries that have multiple important events associated with them. Specifically, we considered three types of query categories, namely: events [93], entities[1], and wars [153]. The keywords corresponding to these queries are shown in Table 7.1. This dataset of history-oriented queries and corresponding Wikipedia articles has been made publicly available as part of the research work carried out by Gupta and Berberich [95].

Metrics. For each history-oriented query, our aim is to detect as many important events as possible. This task can be considered equivalent to producing a summary for a given topic. However, in our setting we disregard the grammatical structure or temporal ordering. Rather our focus is specifically on precision and recall of information. We consider the frequent words in each event cluster $\mathcal{W}$ as a sentence and concatenate them from top-ranked clusters

[1]`http://usatoday30.usatoday.com/news/top25-influential.htm`

to form a system-generated summary. In order to test this objectively, we utilized the Rouge-N measure [208] for evaluating the quality of summaries produced by our methods. We used the Wikipedia entry corresponding to the history-oriented query as the gold-standard summary. Rouge-N then computes the overlap of n-grams between the gold standard summary and the summary produced by the system under test. We report the Rouge-N precision, recall, and F_1.

Systems. We evaluate three variations of EventMiner algorithm with increasing sophistication. For the first system, we consider a naïve variation of EventMiner that computes similarities of various semantics by only considering surface-level equivalence. That is two semantic annotations are considered to be similar if and only if their tokens match. For example, 1995 and 1990s are not similar to each other. We call this system EMBaseline. For the second system, we compute the temporal similarity using uncertainty-aware time model Utm; the geographic similarity by computing area overlap in MBRs; and entity-entity similarity with MWSim. Therefore, in this system, 1995 and 1990s are similar; as are new york and usa; and also ronald reagan and nancy reagan. We call this system EMSimilar. For the final system we compute the similarity between temporal expressions using proximity-aware time model Ptm; the geographic similarity by computing closeness between MBRs; and entity-entity similarity with MWSim. Therefore, this system considers 1999 and 2000 proximate; also canada and usa are considered to be proximate. We refer to this system as EMProximity.

Parameter Tuning. We performed the experiments by retrieving top-25 pseudo-relevant documents, i.e., $K = 25$ for every keyword query in the testbed. We choose this as a conservative estimate for the number of highly relevant documents given the keyword query. For retrieving these documents, we used the Okapi BM25 method with standard parameter settings $k_1 = 1.2$ and $b = 0.75$. These documents are subsequently split at sentence-level granularity.

The different EventMiner systems are next executed with these sentences as input. Weights for different similarities are: $\rho_1 = 0.50$, $\rho_2 = 0.25$, and $\rho_3 = 0.25$, giving more importance to temporal similarity as compared to the other two similarities. This was due to the observation that annotations for time were more accurate as compared to named entity annotations. Hence, temporal similarity was given a higher weight in computing joint similarity to obtain coherent clusters. Further, the concentration parameter was set to $\alpha = 0.1$ and the strength of prior for text similarity $\beta = 0.1$. The concentration parameter is directly proportional to the probability of creating a new event cluster. Both these values were set by observing their effect on three sample queries, namely: *summer olympics*, *us presidential elections*, and *george w bush*. We perform Gibbs Sampling for a total of 50 iterations, with early termination of the algorithm if cluster assignments do not change between subsequent iterations. We limited ourselves to a moderate number of iterations keeping in mind the quadratic complexity of the EventMiner algorithm. For each system, we picked top-5 clusters ranked by their scores.

Category	System	Rouge-1		
		Recall	*Precision*	F_1
Event	EMBaseline	0.31	0.24	0.17
	EMSimilar	0.29	0.26	0.17
	EMProximity	0.25	0.28	0.18
War	EMBaseline	0.15	0.31	0.17
	EMSimilar	0.15	0.32	0.17
	EMProximity	0.11	0.36	0.15
Entity	EMBaseline	0.13	0.49	0.16
	EMSimilar	0.12	0.50	0.15
	EMProximity	0.10	0.52	0.15

Table 11.1: Rouge-1 scores for various systems which are grouped by different categories of history-oriented queries. Precision, recall and F_1 scores are averaged for all queries in that category for each system. Across query categories, we can clearly see an increase in the values of precision as we consider more advanced models for time and space that incorporate proximity.

After obtaining the top-most clusters, we next take the most frequent keywords in each cluster and concatenate them into one sentence and then each of the sentences is concatenated into a system-generated summary. We compare this system summary with the model or ground truth summary from the corresponding Wikipedia entry of the issued keyword query. For comparing the summaries, we use the Rouge-1 score which tests the overlap of unigrams between the two summaries and provides us with precision, recall and F_1 scores averaged over all the queries in that category. Note that the order of the words appearing in each summary has no consequence on the scores. The Rouge software package[2] was utilized for this purpose. We computed these scores by stemming all the words and by removing all stopwords in all the summaries. Further, 1000 samples were considered for its bootstrap re-sampling. The results reported are within 95% confidence intervals.

11.6.2 Results

In this section, we describe the results obtained for the evaluation setup presented earlier.

Results for the three different categories of the history-oriented queries are shown in Table 11.1. The results for the history-oriented queries concerning events show that considering EventMiner algorithm with similarity-based models for semantic annotations increases precision as compared to the baseline method considering only surface level similarity. However, this comes at marginal cost of decrease in recall. Taking into account time and geo-

[2] http://www.berouge.com/Pages/default.aspx

graphical model that considers proximity increases the precision of the events identified by the EventMiner algorithm again at small decrease in recall. This trend is replicated in other history-oriented queries concerning wars and entities. We additionally present anecdotal results generated by EventMiner in Section 11.8.

11.6.3 Discussion

In conducting this research, we faced several pitfalls which we aim to solve in future.

Quality of Annotations. We noticed that empirically the quality of annotations for temporal expressions and for geographic locations is of significantly better quality than for other named entities. Thus, there is a need of restricting the analysis to only high precision annotations.

Cluster Coherence. Considering a joint similarity between time and geographic locations can cause cluster coherence to decrease. This might arise due to the fact that some keyword queries may have large temporal ambiguity but little or no geographic ambiguity and vice-versa.

Dependencies between Annotations. Clearly making an independence assumption between the annotations has not helped us in recalling more events. Our model thus needs to incorporate this aspect.

Scalability. Worst-case time complexity of the EventMiner algorithm is quadratic; this is not desirable if analysis is required for large number of documents. To tackle this, one potential solution is to use hierarchical Dirichlet process mixture models [195, 196].

Evaluation Metrics. Our evaluation metric was highly objective and based on overlap of n-grams computed from text. However, Rouge metric can additionally be modified to take into account similarity between summaries in terms of time, geography and named entities in an ontology. Another avenue to explore will be to have subjective crowd-sourced based evaluation for the identified events.

11.7 Conclusion

In this chapter, we presented EventMiner, an algorithm that clusters sentences in a semantically annotated corpus to identify important events. Our proposed method is based on the framework of Dirichlet process mixture models. We adapted this framework to incorporate similarity along time that leverages uncertainty and proximity in temporal expressions. It also considers similarity and proximity between geographic expressions. Finally, it also accounts for similarity between named entities in an ontology. We tested our method on a collection of history-oriented queries and their corresponding Wikipedia pages to show that considering proximity between temporal and geographic dimension as well as similarity between named entities in an ontology can recall accurate events.

11.8 Anecdotal Results

Next we discuss some anecdotal results for a few history-oriented queries, namely: *george w. bush*, *bill clinton*, *ronald reagan*, *ryan white*, *us presidential elections*, *deng xiaoping*, *pope john paul ii*, *stephen hawking*, *iraq war*, *iraq iran war*, *soviet afghanistan war*, and *lord of the rings movie*, . Each result shows the most coherent and representative cluster with its ten most frequent keywords, geographic locations and time intervals that appear in that cluster. The results were obtained by executing the EMProximity system with 150 iterations of Gibbs Sampling and rest settings same as used for the experimental setup. The captions accompanying Tables 11.2 to 11.13 elaborate on the event depicted.

Keywords	[bush] [wife] [campaign] [george] [washington] [marvin] [gore] [al] [st] [louis]
Time	[12-Apr-2000 , 12-Apr-2000] [04-Aug-2000 , 04-Aug-2000] [01-Jan-2000 , 31-Dec-2000]
Locations	none
Entities	[YAGO:George_W._Bush] [YAGO:Republican_Party_(United_States)] [YAGO:Republican_National_Convention]

Table 11.2: An event cluster for query *george w. bush*. The event identified is that of the inaugural presidential campaign of George W. Bush, whose political affiliation was to the Republican Party.[3]

Keywords	[clinton] [bill] [top] [address] [news] [page] [africa] [front] [african] [home] [hillary]
Time	[01-Jan-1999 , 01-Jan-1999] [23-Aug-1998 , 23-Aug-1998] [03-Apr-1998 , 03-Apr-1998] [01-Jan-1999 , 31-Dec-1999]
Locations	[YAGO:Africa] [YAGO:White_House] [YAGO:United_States]
Entities	[YAGO:Bill_Clinton] [YAGO:Monica_Lewinsky] [YAGO:Hillary_Rodham_Clinton] [YAGO:White_House] [YAGO:Africa] [YAGO:United_States]

Table 11.3: An event cluster for query *bill clinton*. The event identified Bill Clinton's impeachment from presidency due to the affair with Monica Lewinsky in 1999.[4]

[3]https://en.wikipedia.org/wiki/George_W._Bush
[4]https://en.wikipedia.org/wiki/Bill_Clinton
[5]https://en.wikipedia.org/wiki/Ronald_Reagan
[6]https://en.wikipedia.org/wiki/Deng_Xiaoping
[7]https://en.wikipedia.org/wiki/Pope_John_Paul_II
[8]https://en.wikipedia.org/wiki/A_Brief_History_of_Time
[9]https://en.wikipedia.org/wiki/Soviet-Afghan_War
[10]https://en.wikipedia.org/wiki/The_Lord_of_the_Rings_(film_series)
[11]https://en.wikipedia.org/wiki/Iraq_War
[12]https://en.wikipedia.org/wiki/Iran-Iraq_War

Keywords	[reagan] [ronald] [legacy] [opinion] [top] [government] [social] [politics] [corrections] [wilson] [attack]
Time	[27-Jun-2004 , 27-Jun-2004] [01-Jan-2004 , 01-Jan-2004] [07-Jun-2004 , 07-Jun-2004] [01-Jan-1911 , 01-Jan-1911] [11-Jun-2004 , 11-Jun-2004] [14-Jan-1993 , 14-Jan-1993]
Locations	[YAGO:Palo_Alto,_California] [YAGO:California] [YAGO:United_States]
Entities	[YAGO:Ronald_Reagan] [YAGO:California] [YAGO:Nancy_Reagan] [YAGO:United_States] [YAGO:Palo_Alto,_California] [YAGO:Culture_of_the_United_States] [YAGO:Dick_Cheney] [YAGO:Ron_Reagan]

Table 11.4: An event cluster for query *ronald reagan*. Ronald Reagan passed away on June 5, 2004 in California.[8] The cluster reports his funeral which took place on June 11, 2004.[5]

Keywords	[presidential] [elections] [2000] [election] [government] [quest] [gore] [al] [pres] [vice] [politics] [campaign]
Time	[01-Jan-2000 , 01-Jan-2000] [01-Jan-2000 , 31-Dec-2000]
Locations	[YAGO:United_States]
Entities	[YAGO:United_States] [YAGO:Al_Gore]

Table 11.5: An event cluster for query *us presidential elections*. The cluster points to the US Presidential Elections in 2000 for which Al Gore ran as vice president.

Keywords	[top] [china] [deng] [lee] [news] [li] [asia] [world] [government] [nicholas]
Time	[17-Jun-1989 , 17-Jun-1989] [03-Jan-1996 , 03-Jan-1996] [01-Jan-1970 , 31-Dec-1970] [23-Jul-1989 , 23-Jul-1989]
Locations	[YAGO:China] [YAGO:Australia] [YAGO:New_York_City] [YAGO:Ithaca,_New_York]
Entities	[YAGO:China] [YAGO:Australia] [YAGO:Fang_Lizhi] [YAGO:New_York_City] [YAGO:Tiananmen_Square_protests_of_1989] [YAGO:Deng_Xiaoping] [YAGO:Overseas_Chinese] [YAGO:Ithaca,_New_York]

Table 11.6: An event cluster for query *deng xiaoping*. It reports the Tiananmen Square protests of 1989; in which Fang Lizhi and Deng Xiaoping were key named entities.[6]

Keywords	[pope] [opinion] [savior] [cahill] [thomas] [paul] [john] [ii] [editor] [top] [catholic] [april] [saddened]
Time	[08-Apr-2005 , 08-Apr-2005] [02-Apr-2005 , 02-Apr-2005] [05-Apr-2005 , 05-Apr-2005] [03-Apr-2005 , 03-Apr-2005]
Locations	[YAGO:Florida] [YAGO:West_Palm_Beach,_Florida]
Entities	[YAGO:Pope_John_Paul_II] [YAGO:Thomas_Cahill] [YAGO:Kingdom_of_Italy] [YAGO:Camillo_Ruini] [YAGO:Catholic_Church] [YAGO:Judaism] [YAGO:Florida] [YAGO:West_Palm_Beach,_Florida]

Table 11.7: An event cluster for query *pope john paul ii*. The event described is that of his death on April 2, 2005.[7]

[13]https://en.wikipedia.org/wiki/Ryan_White

Keywords	[dr] [black] [hawking] [hole] [gilliam] [physicist] [time] [york] [caltech] [mr]
Time	[01-Nov-1998 , 30-Nov-1998] [01-Jan-1999 , 31-Dec-1999] [01-Jan-1900 , 01-Jan-1900] [03-Apr-1988 , 03-Apr-1988] [01-Jan-1988 , 31-Dec-1988]
Locations	[YAGO:University_of_Cambridge] [YAGO:Wildwood,_New_Jersey] [YAGO:Arizona]
Entities	[YAGO:Larry_Doyle_(writer)] [YAGO:University_of_Cambridge] [YAGO:Leonard_Susskind] [YAGO:Robert_Redford] [YAGO:William_Morris] [YAGO:A_Brief_History_of_Time] [YAGO:Wildwood,_New_Jersey] [YAGO:Arizona] [YAGO:Associated_Press] [YAGO:Random_House] [YAGO:Lou_Gehrig]

Table 11.8: An event cluster for query *stephen hawking*. The cluster shows the dates on which first edition of his book "A Brief History of Time" was released – 1988 and tenth anniversary edition of the book was released – 1998.[8]

Keywords	[soviet] [afghanistan] [war] [military] [beginning] [party] [forces] [union] [exhibition] [mixed]
Time	[01-Jan-1938 , 01-Jan-1938] [01-Jan-1980 , 01-Jan-1980] [29-Apr-1988 , 29-Apr-1988] [01-Jan-1979 , 01-Jan-1979] [01-Apr-1988 , 01-Apr-1988] [29-Jul-1987 , 29-Jul-1987] [01-Jan-1950 , 01-Jan-1950]
Locations	[YAGO:Soviet_Union] [YAGO:Afghanistan] [YAGO:Moscow] [YAGO:Kabul] [YAGO:United_States]
Entities	[YAGO:Soviet_Union] [YAGO:Afghanistan] [YAGO:Mohammad_Najibullah] [YAGO:Moscow] [YAGO:Bosniaks] [YAGO:Kabul] [YAGO:United_States]

Table 11.9: An event cluster for query *soviet afghanistan war*. It depicts the Soviet-Afghanistan conflict that lasted from 1979 to 1989.[9]

Keywords	[lord] [rings] [top] [movie] [motion] [opinion] [pictures] [article] [elvis] [jackson] [trilogy] [movies]
Time	[15-Dec-2002 , 15-Dec-2002] [01-Jan-1987 , 01-Jan-1987] [25-Jan-2004 , 25-Jan-2004] [12-Nov-2002 , 12-Nov-2002] [01-Jan-2003 , 31-Dec-2003] [01-Jan-1982 , 01-Jan-1982] [11-Jan-2004 , 11-Jan-2004] [28-Dec-2002 , 29-Dec-2002] [07-Sep-2003 , 07-Sep-2003] [01-Dec-2003 , 31-Dec-2003]
Locations	[YAGO:Weldon,_Northamptonshire] [YAGO:Wellington]
Entities	[YAGO:J._R._R._Tolkien] [YAGO:Weldon,_Northamptonshire] [YAGO:Wellington] [YAGO:Carol_Ann_Lee] [YAGO:Peter_Jackson]

Table 11.10: An event cluster for query *lord of the rings movie*. It captures the location where the movie was shot – Wellington and the author of the book on which the movie is based on – J. R. R. Tolkien.[10]

Keywords	[iraq] [states] [united] [war] [opinion] [top] [international] [relations] [defense] [armament] [president] [time] [fearful] [david]
Time	[13-Apr-2006 , 13-Apr-2006] [15-Jun-2005 , 15-Jun-2005] [16-Jul-2003 , 16-Jul-2003] [16-Oct-2003 , 16-Oct-2003] [30-Jun-2005 , 30-Jun-2005]
Locations	[YAGO:New_York_City] [YAGO:Port_Washington,_Wisconsin] [YAGO:Radcliff,_Kentucky] [YAGO:Iraq] [YAGO:United_States]
Entities	[YAGO:Iraq] [YAGO:United_States_Army] [YAGO:Donald_Rumsfeld] [YAGO:United_States_Department_of_Defense] [YAGO:George_W._Bush] [YAGO:Jim_Folsom] [YAGO:New_York_City] [YAGO:Port_Washington,_Wisconsin] [YAGO:Radcliff,_Kentucky] [YAGO:United_States]

Table 11.11: An event cluster for query *iraq war*. The cluster shows the start of Iraq War in 2003.[11]

Keywords	`[iraq] [iran] [war] [oil] [international] [top] [faw]`
	`[port] [east] [world] [delegate] [rafsanjani]`

Time	`[01-Mar-1986 , 31-Mar-1986] [01-Sep-1980 , 01-Sep-1980]`
	`[01-Sep-1980 , 30-Sep-1980] [01-Jan-1970 , 31-Dec-1970]`
	`[01-Jan-1980 , 01-Jan-1980] [23-Sep-2003 , 23-Sep-2003]`
	`[25-Jan-1991 , 25-Jan-1991] [01-Aug-1988 , 31-Aug-1988]`
	`[17-Mar-2006 , 17-Mar-2006] [01-Jan-1000 , 01-Jan-1000]`
	`[01-Jan-1988 , 31-Dec-1988] [02-Oct-2003 , 02-Oct-2003]`

Locations	`[YAGO:Iran] [YAGO:Iraq] [YAGO:Geneva]`

Entities	`[YAGO:Iran] [YAGO:Iraq] [YAGO:United_Nations]`
	`[YAGO:Akbar_Hashemi_Rafsanjani] [YAGO:Iranian_peoples] [YAGO:Gulf_War]`
	`[YAGO:Geneva] [YAGO:United_Nations_Security_Council] [YAGO:Fao_Landing]`
	`[YAGO:Western_world] [YAGO:Persian_people] [YAGO:Iran-Iraq_War]`
	`[YAGO:National_Iraqi_News_Agency]`

Table 11.12: An event cluster for query *iraq iran war*. It describes the conflict between Iran and Iraq that lasted from 1980 to 1988.[12]

Keywords	`[ryan] [school] [white] [senior] [friends] [mother] [kokomo] [edward]`
	`[president] [riley] [attendance] [family]`

Time	`[01-Jan-1984 , 01-Jan-1984] [01-Jan-1199 , 31-Dec-1199]`

Locations	`[YAGO:New_York] [YAGO:Kingston,_New_York] [YAGO:Cicero,_Illinois]`
	`[YAGO:Indianapolis] [YAGO:Kokomo,_Indiana]`

Entities	`[YAGO:Megan] [YAGO:Saint_Joseph] [YAGO:Edward_VI_of_England]`
	`[YAGO:New_York] [YAGO:Matt_Ryan] [YAGO:Ronald_Reagan] [YAGO:Nancy_Reagan]`
	`[YAGO:Hamilton_Heights_School_Corporation] [YAGO:Cicero,_Illinois]`
	`[YAGO:Indianapolis] [YAGO:Kokomo,_Indiana] [YAGO:Taco_Bell]`
	`[YAGO:Kelly_Ryan] [YAGO:Ryan_White] [YAGO:Kingston,_New_York]`
	`[YAGO:Kelly_Osbourne]`

Table 11.13: An event cluster for query *ryan white*. It depicts the event when Ryan White was not allowed to attend school due to health concerns related to his HIV/AIDS infection.[13]The time interval `[01-Jan-1199, 31-Dec-1199]` is mined due to erroneous annotation by the temporal annotator.

CHAPTER 12

JIGSAW: STRUCTURING TEXT INTO TABLES

• Dhruv Gupta and Klaus Berberich. JIGSAW: Structuring Text into Tables. In ICTIR 2019, pages 237–244.

12.1 Introduction

Tables are structured summaries obtained from multiple documents. Tables already present in documents or web tables are an important resource for tasks such as question answering, fact checking, and analytics. Manually generating tables is a laborious task. Teams of journalists often collaborate to curate tables using spreadsheet tools (e.g., Google Fusion Tables) [89]. To reduce this human effort, we need an information retrieval (IR) system that instead of presenting ten blue links, generates structured tables in response to queries.

⟨![*google*|*google inc.*|*google llc*], ![*acquired*|*takeover*|*bought*], {!ORG, ?TIME, ?MONEY}⟩

NO.	SCORE	ORG	TIME	MONEY
1.	0.721	*motorola mobility*	[2014, 2014]	[\$ 2.22×10^{8}, \$ 1.95×10^{10}]
2.	0.057	*boston dynamics*	[2013, 2013]	[\$ 1.20×10^{9}, \$ 3.60×10^{9}]
3.	0.036	*youtube*	[2006, 2006]	[\$ 1.00×10^{9}, \$ 1.50×10^{9}]
4.	0.014	*skybox imaging*	[2014, 2014]	[\$ 2.78×10^{8}, \$ 8.33×10^{8}]
5.	0.008	*redwood robotics*	[2004, 2004]	[\$ 2.00×10^{5}, \$ 4.00×10^{5}]
6.	0.007	*quickoffice*	[2012, 2012]	[\$$14.99 \times 10^{0}$, \$$14.99 \times 10^{0}$]
7.	0.006	*gecko design*	[2014, 2014]	[\$ 1.00×10^{9}, \$ 1.00×10^{9}]
8.	0.006	*imperium*	[2013, 2013]	[\$ 4.50×10^{6}, \$ 1.50×10^{7}]
9.	0.005	*makani power*	[2013, 2013]	[\$ 1.90×10^{10}, \$ 1.90×10^{10}]
10.	0.005	*nortel*	[2011, 2011]	[\$ 4.50×10^{9}, \$ 4.50×10^{9}]

Figure 12.1: Table generated by JIGSAW from the GDelt news archive for Google acquisitions. The structured query contains aliases for Google, paraphrases for the predicate "acquisitions" and contains bindings ORG, TIME, and MONEY that are to be filled in for each acquisition.

Results in the form of structured snippets [10], knowledge panels [115], and lists of related entities [119] are gaining prominence in search results. To create them, structured data in the form of knowledge graphs (KGs) [9, 68] and web tables [52, 212] are leveraged. These approaches are limited as they can only use fixed schema associated with individual web tables or KG schema in the form of ⟨S,P,O⟩ triples. To generate tables for user-defined schema from text: we need an expressive query language that can define arbitrary relationships between entities and an IR system that can retrieve concise text regions containing such relationships.

To generate tables from unstructured text collections, we leverage semantic annotations that natural language processing (NLP) tools can now provide accurately. Concretely, annotations in the form of part-of-speech (e.g., *google*⊕NNP), named entities (e.g., *larry page*⊕PERSON), temporal (e.g., *2020s*⊕[2020, 2029]), and numerical expressions (e.g., *a million dollars*⊕\$$1 \times 10^{6}$) help us impose a lexico-syntactic structure over unstructured text. Using this insight, we can perform structured search over large document collections and put together tables using redundant, partial, and paraphrased pieces of text spread across millions of documents.

Contributions and Outline. JIGSAW structures text into tables, for user-defined schema within seconds. To assemble tables, JIGSAW uses GYANI as an indexing infrastructure for structured search over large annotated document collections [98]. The key contributions this work makes are:

1. To speed up the retrieval from the inverted indexes, we describe a *greedy query optimizer* (Section 12.6). Thus, unlike open information extraction (open-IE) based approaches, that require an entire scan of the document collection, we can extract relevant annotated text regions for a query using indexes over annotated text.

2. To generate tables we describe three operator classes: QUERY, LINK, and ANALYZE. The operators in QUERY describe the table schema and help shape each retrieved annotated text region into a row for the table. The QUERY operators (Section 12.5) additionally link the row to its originating document, which helps us in estimating NULL values (i.e., values for which we can not spot annotation values) from its context. This is not possible with existing open-IE approaches, as no provenance information is kept for facts during extraction.
3. The LINK operators (Section 12.7) reconcile near-duplicate mentions of named entities, temporal, and numerical expressions by taking into account their semantics (e.g., *2020s* and *2025* should be considered similar).
4. Finally, the ANALYZE operators (Section 12.8) allow for aggregation and ranking over the linked rows in the table. Fig. 12.1 shows an example of a generated table by JIGSAW for acquisitions by Google.

12.2 Related Work

Annotated document collections have been leveraged by several studies [61, 98, 145] for mining valuable data. Li et al. [145] proposed key operators to analyze annotated text corpora using relational databases. Clarke et al. [61] and Gupta and Berberich [98] describe efficient algorithms to perform search in tagged text corpora using inverted index operations. However, none of the above systems support table generation capabilities that can aggregate redundant, partial, and paraphrased text evidences. A recent survey on the use of web tables [52], describes the impact web tables have had on commercial search engines. The authors also describe progress that has been made in terms of augmenting web tables from additional data sources such as KGs. Cannaviccio et al. [54] aim to link predicates from KGs to relations between attributes in web tables in order to understand their schema. Yang et al. [206] and Zhang and Balog [212] generate tables from KGs to answer keyword queries. The above approaches focus on leveraging existing structured resources (e.g., web tables and KGs) but not to generate tables from unstructured text. Knowledge graph population techniques rely on identifying salient extractions from documents or the Web [35]. Key works in this direction are [73, 160, 161, 166, 210]. Nakashole et al. [161] rely on distributed itemset mining for determining salient triples to be added to KGs. Niu et al. [166] and Zhang et al. [210] use Markov logic to reconcile and canonicalize triples. Dong et al. [73] verify the correctness of the extracted triples by using a combination of prior-knowledge learned using random-walks and neural-networks. The work by Mitchell et al. [160] uses a suite of machine learning methods including embedding-based methods to verify the quality of triples to be added to its KG. However, all these methods are bound to a fixed schema for extraction. Furthermore, the discussed methods solely rely on offline methods of pre-computing the KG. JIGSAW, on the other hand, allows table generation for user-defined schema efficiently and interactively in a query-driven manner.

12.3 Problem Definition

JIGSAW generates a table, given its schema, from large annotated document collections. As input, we are given a **structured query** $\mathcal{Q}$ that describes the table schema:

$$\text{Structured Query:} \quad \mathcal{Q} = \langle a_1, a_2, \ldots, a_N \rangle, \tag{12.1}$$

where, each **attribute** a can be specified with the help of word sequences (e.g., $\langle$*took over*$\rangle$), annotations (e.g., MONEY) or a combination of both word sequences and annotations (e.g., $\langle$*youtube*$\rangle\oplus$ORG). Let Σ_i denote the domain of values that attribute $a_i \in \mathcal{Q}$ can take. The **table schema** $\mathcal{R}$ is then defined by the query attributes. Concretely, a row $r \in \mathcal{T}$ has the following structure:

$$\text{Row Structure:} \quad r \subset 2^{\Sigma_1} \times 2^{\Sigma_2} \times \ldots \times 2^{\Sigma_N}. \tag{12.2}$$

In Equation 12.2, an attribute of query $a_i \in \mathcal{Q}$ can refer to multiple elements from its domain Σ. Thus, tables generated by our approach can contain cells with multiple values (zero normal form). To construct the **final table** $\mathcal{T}$ we proceed in two steps. First, we construct a **raw table** $\overline{\mathcal{T}}$. To construct $\overline{\mathcal{T}}$, we retrieve text regions that match the query template $\mathcal{Q}$. Using the retrieved text regions we create a raw table that is a collection of rows that contain document metadata $d_{\texttt{id}}$, text region span in document $\texttt{text}(\cdot)$, and value(s) $\overline{c}$ (for simplicity consider singular values in Equation 12.3) for the query template:

$$\text{Raw Table:} \quad \overline{\mathcal{T}} = \bigcup \overline{r} = \bigcup \big(\ d_{\texttt{id}}, \texttt{text}(\cdot), \overline{c}_1, \overline{c}_2, \ldots, \overline{c}_N \ \big). \tag{12.3}$$

In the raw table $\overline{\mathcal{T}}$, we allow for relaxed matches to the query template that will result in **unknown values** (**NULL**) as cell values. We resolve NULL values using two approaches: local and global resolution. Local resolution uses the document which establishes the provenance of the row to determine the missing values. Whereas, global resolution relies on other similar rows in the raw table (thereby leveraging cross-document evidences) to infer NULL values.

Second, having generated the raw table $\overline{\mathcal{T}}$, we link and aggregate rows that mention similar text, entities, relations, temporal or numerical values by leveraging the semantics of annotations in the cell values to arrive at the final table $\mathcal{T}$. Each aggregated row in the final table is assigned a $\texttt{score}(r)$ that reflects its prominence in the document collection. Furthermore, the rows in table $\mathcal{T}$ can be **ranked** using its originating context (provenance) $\texttt{rank}(r)$ and diversified amongst other rows in the table $\texttt{diversify}(r)$. Structure of the final table can be defined as:

$$\mathcal{T} = \bigcup r = \bigcup \big(\ \cup d_{\texttt{id}}, \cup \texttt{text}(\cdot), \texttt{score}(\cdot), c_1, c_2, \ldots, c_N \ \big).$$

12.4 Indexes Over Annotated Text

Annotated Text Model

Consider, a large document collection $\mathcal{D} = \{d_1, \ldots, d_{|\mathcal{D}|}\}$. Each document $d \in \mathcal{D}$ consists of sentences $d = \langle s_1, s_2, \ldots, s_{|d|}\rangle$ which further consist of a sequence of words $s = \langle w_1, w_2, \ldots, w_{|s|}\rangle$ drawn from the vocabulary of the collection $\Sigma_{\mathcal{V}}$. NLP tools can now deliver high-quality annotations over text in the form of parts-of-speech, named entities, numerical quantities, and temporal expressions. Consider, a NLP annotator $\mathcal{L}$ that further tags the sequence of words $\langle w_i \ldots w_j\rangle$ $(i \leq j)$ in documents with elements ℓ from its annotation alphabet $\Sigma_{\mathcal{L}}$. This way we obtain layers-of-annotation over text in documents (see Figure 12.2). Semantic annotations help us impose a lexico-syntactic structure over text. This in turn helps us perform structured search over annotated document collections. To do so, we leverage GYANI [98], as our backend indexing infrastructure. We briefly describe the indexes we use to support our query operators.

Inverted Indexes

Text Indexes help locate and compute statistics for text regions. To this end, we first create N-GRAM INDEXES and DICTIONARIES. N-GRAM indexes record unigrams, bigrams, and trigrams with their positional spans. While, the dictionaries record the document frequency (df) and collection frequency (cf) of n-grams. The n-grams are derived from the sentences in each document of $\mathcal{D}$. Furthermore, to speed up the retrieval of surface forms that are slight variations of a complete label (e.g., [*youtube video website* | *youtube website*]) we create SKIP-GRAM INDEXES and DICTIONARIES. Skip-gram indexes record ordered co-occurrences of words within a context of ten words. **Annotation Indexes** record for each annotation layer its element and positional span (e.g., MONEY with span $[8, 11]$ in Figure 12.2). **Annotated Text Indexes** record positional spans for pair-wise and ordered combinations of word sequences and annotations. For each such combination in a sentence of a document, we create two indexes: 2-FRAGMENT and 2-STITCH indexes. Where, 2-FRAGMENTS are annotated word sequences (e.g., $\langle$*Google*$\oplus$ ORG$\rangle$ in Figure 12.2) and 2-STITCHES are ordered co-occurrences of word sequences with other annotations in the sentences (e.g., $\langle$*YouTube*, DATE$\rangle$ in Figure 12.2).

Direct Index

The inverted indexes help us retrieve positional spans corresponding to text regions for a given structured query. However, to retrieve the resolved annotation values (e.g., the resolved annotation value $> \$1 \times 10^9$ for the phrase *over a billion dollars*) we need to access the different layers of annotation for a given text region. To this end, we store the documents with all annotation layers and sentence boundaries in a DIRECT index.

12.5 QUERY Operators

We next describe the operators that help define the table schema.

Binding Operator ($\boxed{!\ell}$ and $\boxed{?\ell}$) are unary operators that specify the annotations or word sequences that should be part of the table schema. The binding operator can be specified using annotations, e.g., ⟨ !PERSON, !ORG ⟩ or word sequences, e.g., !⟨*acquired*⟩. The $\boxed{!\ell}$ operator requires that the values *must* be present in the text regions being retrieved to populate a column in the table. Whereas, $\boxed{?\ell}$ operator can perform a relaxed query match. That is, the $\boxed{!\ell}$ operator can not result in NULL values, whereas the $\boxed{?\ell}$ can be relaxed to a NULL value. Semantics of $\boxed{!\ell}$ and $\boxed{?\ell}$ can be specified as:

$$\text{match}(\ell_{\langle i,j\rangle}, s) = \{s \in d \mid d \in \mathcal{D} \wedge \ell_{\langle i,j\rangle} \sqsubset s\},$$

where, $\ell_{\langle i,j\rangle}$ represents the text region $\langle w_i, \ldots, w_j\rangle$ tagged with annotation ℓ and $\sqsubset$ denotes that $\ell_{\langle i,j\rangle}$ is a contiguous subsequence in sentence s. Bindings can be decorated with markers to increase recall. These markers are: UNION, WILDCARD, and MULTIPLICITY.

UNION MARKER (|) helps to specify paraphrases for a word sequence binding, e.g., ![*acquired* | *takeover* | *buy out*]. The UNION marker applies the Boolean disjunctive semantics to the text regions matched for each of the paraphrases in the set.

WILDCARD MARKER ($\boxed{*}$) helps to indicate variable-length gaps in word sequences. With the $\boxed{*}$ marker the corresponding cell values for the binding contain the text regions that fill in the wildcard. An example query using $\boxed{*}$ marker is: ?⟨*obtained* $\boxed{*}$ *award*⟩.

MULTIPLICITY MARKER ($\boxed{\times\{m,n\}}$) helps to associate multiple annotation values in a cell value for the binding. The MULTIPLICITY marker specifies the minimum number m and maximum number n of annotation type ℓ a cell can contain for the binding. As an example query consider: ⟨!⟨*google* $\boxed{*}$ *acquired*⟩, !ORG $\boxed{\times\{1,3\}}$⟩.

Stack Operator ($\oplus$) is a binary operator that helps in attaching additional semantics to word sequences, e.g., !⟨*paris*$\oplus$LOCATION⟩. The semantics for the STACK $\oplus$ operator are:

$$\text{match}(w_{\langle i,j\rangle} \oplus \ell, s) = \{s \in d \mid d \in \mathcal{D} \wedge\ \ell_{\langle i,j\rangle} \sqsubset s \wedge w_{\langle i,j\rangle} \sqsubset s\},$$

where, the annotation ℓ and the word sequence $w_{\langle i,j\rangle}$ occupy the same positional span $[i,j]$ in the sentence of a document $s \in d$.

A structured query containing $\boxed{!\ell}$ or $\boxed{?\ell}$ will match annotated text regions that contain their arguments in an *ordered* sequence. A sequential order amongst the arguments of the query $a \in \mathcal{Q}$ helps to curate tables for **asymmetric relations**. For instance, in the table for the query, ⟨!ORG$_1$, !⟨*has acquired*⟩, !ORG$_2$⟩, the matches for ORG$_1$ and ORG$_2$ can not be exchanged.

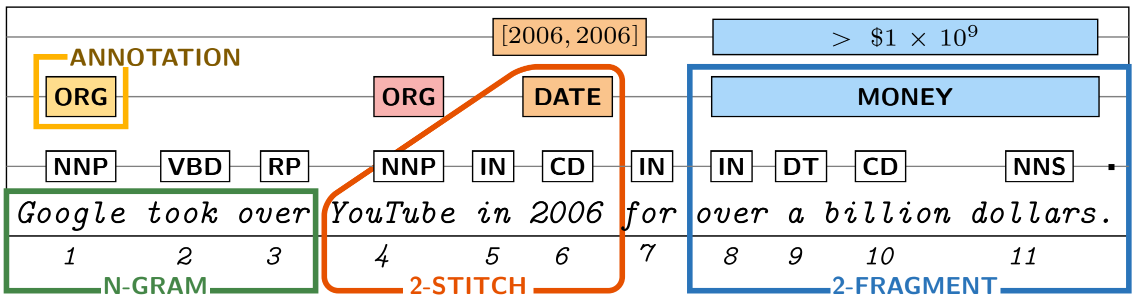

[2006, 2006]
> $1 × 10^9
ANNOTATION
ORG
ORG
DATE
MONEY
NNP VBD RP NNP IN CD IN IN DT CD NNS .
Google took over YouTube in 2006 for over a billion dollars.
1 2 3 4 5 6 7 8 9 10 11
N-GRAM
2-STITCH
2-FRAGMENT

Figure 12.2: Inverted indexes over the annotated text model.

Unorder Operator ($\boxed{\{\bullet\}}$) is a n-ary operator that allows matching text regions irrespective of the order in which its arguments are mentioned. For example, the query, ⟨!ORG, !⟨*has acquired*⟩, {!ORG, ?MONEY, ?DATE}⟩, treats the bindings for ORG, MONEY, and DATE in an unordered manner. The UNORDER operator is useful for creating tables for **symmetric relations** where the order amongst the bindings is not important, e.g., {!PERSON, !*married*, !PERSON}.

12.6 Query Processing

We next discuss how to derive a query execution plan for retrieval of annotated text regions for a structured query $\mathcal{Q}$.

Query Graph. The structured query $\mathcal{Q}$, represents a template to be matched against the annotated text model. This sequence of query operators and their arguments can be succinctly represented in a graph. Let, $\mathcal{G}(V, E)$ represent a directed graph corresponding to the query $\mathcal{Q}$, where the set of vertices V represents the arguments (e.g., word sequences or annotation types) and E be a set of directed edges that represents the query operator semantics. We associate a function $q(e)$ with each edge $e \in E$, that defines either sequential, stacking, unorder, or multiplicity semantics between the connecting vertices. Figure 12.3 shows an example of a query graph.

Sequential Semantics inherent in the query structure $\mathcal{Q} = \langle a_1, a_2, \ldots, a_k \rangle$ are represented by a directed edge between two of the query arguments. Sequential semantics can also be specified with the help of the BINDINGS operator. The sequential semantics $q(e) \equiv q(a_i \rightarrow a_j)$ ensure that the mention of the argument a_i is before that of a_j in the annotated text model:

$$q(e) \equiv q\big(a_i \rightarrow a_j\big) = \Big\{ s \in d | d \in \mathcal{D} \wedge \ell^i_{\langle m,n \rangle} \in s$$
$$\wedge \ell^j_{\langle p,q \rangle} \in s \wedge (m \leq n) \wedge (n < p) \wedge (p \leq q) \Big\},$$

where, $\ell^i_{\langle m,n \rangle}$ represents the annotation matching a_i and $\ell^j_{\langle p,q \rangle}$ represents the annotation matching a_j. The MULTIPLICITY marker constraints additionally imply that the number of argument value lie within bounds conveyed along with the marker.

Stacking Semantics specified by the STACK operator $q(e) \equiv q(a_i \xrightarrow{\oplus} a_j)$ conveys that the arguments to the $\oplus$ operator occupy the same positions in the sentence but adorn different annotation layers in the text model:

$$q(e) \equiv q\big(a_i \xrightarrow{\oplus} a_j\big) = \Big\{ s \in d | d \in \mathcal{D} \wedge \ell^i_{\langle m,n \rangle} \in s$$
$$\wedge \ell^j_{\langle p,q \rangle} \in s \wedge (m \leq n) \wedge (p \leq q) \wedge (m = p) \wedge (n = q) \Big\}.$$

Unorder Semantics specified by the $\boxed{\{a_j, \ldots, a_k\}}$ operator $q(e) \equiv q(a_i \xrightarrow{*} \{a_j, \ldots, a_k\})$ specifies that any of the arguments specified by $a_j, \ldots, a_k$ can follow a_i in the annotated text model (for simplicity, we show only a_j from $\{a_j, \ldots, a_k\}$ below):

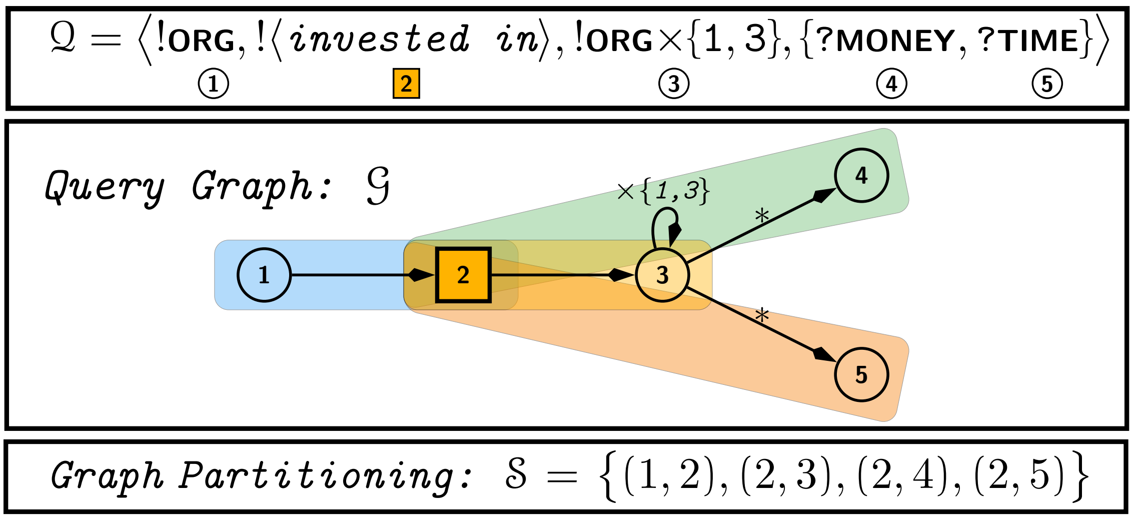
$$q(e) \equiv q\big(a_i \xrightarrow{*} a_j\big) = \Big\{ s \in d | d \in \mathcal{D} \wedge \ell^i_{\langle m,n \rangle} \in s$$
$$\vee \ell^j_{\langle p,q \rangle} \in s \wedge \big((m \leq n) \vee (p \leq q) \vee (n < p)\big) \Big\}.$$

12.6.1 Query Optimization

We next discuss how we can process the query graph.

Graph Partitions. We partition the graph $\mathcal{G}$ into a set of subgraphs $\mathcal{S}$ such that each subgraph $S \in \mathcal{S}$ either consists of a single vertex or a vertex pair (u, v) where vertex v is reachable from u. That way, each subgraph corresponds to an indexing unit for which we can retrieve its corresponding posting list from the five different indexes described in Section 12.4. Concretely, for a subgraph where the vertex pair consists of a word sequence and annotation, we retrieve their results using the 2-STITCH index (e.g., in Figure 12.3, subgraph $(1, 2)$). For a subgraph, where the vertex is an annotation, we can retrieve their results using the ANNOTATION indexes. For a vertex that contains the wildcard marker $\boxed{*}$, we can either directly lookup their posting lists from the SKIP-GRAM index or compute the resultant posting list using N-GRAM indexes. For a vertex that contains the stack operator, we can retrieve its posting list using the 2-FRAGMENT indexes.

Greedy Graph Partitioning and Optimization. In a graph partitioning, certain attribute combinations can be retrieved more quickly than others, e.g., (ORG $\rightarrow$ *acquired*) versus (ORG $\rightarrow$ MONEY). Since, there can exist multiple graph partitions, we opt for that one which contains subgraphs whose corresponding posting lists are shortest in the indexes. Thus, a naïve decomposition of the graph into subgraphs in which each vertex is adjacent to each other (e.g., $\mathcal{S} = \{(1, 2), (2, 3), (3, 4), (3, 5)\}$ in Figure 12.3) may not correspond to an efficient query execution plan. To speedup the query processing, we partition the graph in a *greedy manner*. Specifically, we seek **anchor vertices**, that correspond to bindings for n-grams (e.g., trigrams) and bindings for annotation types (e.g., MONEY) whose document frequency is less than other



$\mathcal{Q} = \langle !\text{ORG}, !\langle \textit{invested in} \rangle, !\text{ORG} \times \{1, 3\}, \{?\text{MONEY}, ?\text{TIME}\} \rangle$
1
2
3
4
5
Query Graph: $\mathcal{G}$
$\times\{1,3\}$
*
*
Graph Partitioning: $\mathcal{S} = \{(1,2),(2,3),(2,4),(2,5)\}$

Figure 12.3: An example query, its graph, its partitioning, and assembly. Square nodes correspond to anchor vertices. Each graph partition $S \in \mathcal{S}$ is highlighted in color in the query graph $\mathcal{G}$.

bindings in $\mathcal{G}$. Therefore, a subgraph (indexing unit) whose single vertex comprises of an anchor vertex shall have overall document frequency less than the anchor vertex by itself. Concretely, we first identify anchor vertices whose document frequencies are least amongst the bindings in the query graph $\mathcal{G}$ using dictionaries. Second, we compute subgraphs of vertex pairs: one of which is an anchor vertex and the other is reachable from the anchor vertex. Consider the query in Figure 12.3, where the anchor node corresponds to the bigram *invested in*. Using this anchor vertex, we can partition the query graph as $\mathcal{S} = \{(1, 2), (2, 3), (2, 4), (2, 5)\}$, where each vertex is reachable from the anchor vertex. **Direct Index** can be further used in conjunction with the inverted indexes to speed up the processing of the query graph. We do this by keeping track of the number of common documents when processing the subgraphs in the graph partitioning $\mathcal{S}$. When this number is small (e.g., ≤ 25), we can switch over to the DIRECT index to inspect the annotation layers for the remaining subgraphs in the partition.

12.6.2 Assembling the Puzzle (Raw Table)

Using the posting lists for the subgraphs in the partitioned graph, we assemble the complete text regions as evidences. This is done by computing the overlaps of the positional spans for the text regions in each document, with respect to the anchor vertex. This process of assembling the text regions as evidences is illustrated in Figure 12.3. The assembled text regions help us to generate the raw table $\overline{\mathcal{T}}$. The text regions gathered as evidences however contain only the positional spans. At this stage, we consider only those positional spans that are short and span a sentence. We prefer concise sentences as they yield semantically meaningful rows in the table. To do this, we first rank positional spans by increasing length. Then, we check that they lie within a sentence

`for a billion dollars`	$\equiv$	$\$10^9$	$\equiv$	$[10^9, 10^9]$
`for over a billion dollars`	$\equiv$	$> \$10^9$	$\equiv$	$[10^9, 10^9 + \Delta]$
`for under a billion dollars`	$\equiv$	$< \$10^9$	$\equiv$	$[10^9 - \Delta, 10^9]$
`for around a billion dollars`	$\equiv$	$\sim \$10^9$	$\equiv$	$[10^9 - \Delta, 10^9 + \Delta]$

Figure 12.4: Modeling uncertainty in numerical expressions.

using the sentence boundaries stored in the DIRECT index. These positional spans are next filled in with values for the various bindings in the structured query $\mathcal{Q}$. To fill in the values, we turn to the DIRECT index that stores within it the values for the annotations and the word sequences corresponding to the assembled positional spans. To generate the raw table, we instantiate a table with the number of columns equal to the number of bindings present in the structured query $\mathcal{Q}$. For each retrieved text region, we create a row in the table. Then, for each binding we lookup its cell value using the DIRECT index. At this step, we additionally verify the multiplicity constraints, if present. Also, if no value for a binding could be found, we fill its corresponding cell value as NULL. The NULL values are inferred from other near-duplicate rows using `LINK` and `ANALYSIS` operators.

12.7 Semantic `LINK` Operator

`LINK` operators group together near-duplicate mentions of text, entities, temporal, and numerical values in the raw table to generate the final table. The `LINK` operators provide functionality that is similar to that of deduplication in databases [76, 163]. However there the focus has been on linking records using only surface forms of attribute values. In contrast, our `LINK` operators take into account the context (or document) from which the row has been derived and collection-level statistics. Furthermore, we model the semantics behind the annotations that are part of the table schema when applying `LINK`. We model two kinds of semantics: text and numerical (Section 12.7.1 and Section 12.7.2). We model the semantics of text for the annotation types of: part-of-speech and named entities of types PERSON, ORGANIZATION, LOCATION, and MISC. We model the semantics of numbers for the annotation types: DATE, TIME, MONEY, PERCENT, and NUMBER. Additionally, we can locally resolve NULL values (local NULL resolution) (Section 12.7.3) by using the provenance of each row.

12.7.1 Semantic Model for Text

To link word sequences that refer to the same entity (PERSON, ORGANIZATION, and LOCATION) or concept (MISC) in different rows we rely on three similarity computations: surface, contextual, and global.

Surface Similarity

Surface similarity establishes similarity between two strings in cell values using traditional edit-distance based measures. To this end, we use the Jaro-Winkler similarity [203], to compute the similarity between two text cell values $\bar{c}_1$ and $\bar{c}_2$ containing text . We denote this surface level similarity measure by: $\text{sim}_{\texttt{surface}}(\bar{c}_1, \bar{c}_2)$. For example,

$$\underset{\texttt{surface}}{\text{sim}}(\textit{motorola}, \textit{motorola mobility}) = 0.91.$$

Contextual Similarity

Contextual similarity computes the similarity between the originating text regions of the cell values. For example, we can link together the word sequences, ***youtube*** and ***video sharing*** based on the similarity of their contexts, e.g., ***google acquired the video website youtube*** and ***google buys out video sharing platform, youtube***. Concretely, the local text similarity is defined below:

$$\underset{\texttt{context}}{\text{sim}}(\bar{c}_1, \bar{c}_2) = \frac{|\ \texttt{text}(\bar{c}_1) \cap \texttt{text}(\bar{c}_2)\ |}{|\ \texttt{text}(\bar{c}_1) \cup \texttt{text}(\bar{c}_2)\ |}. \tag{12.4}$$

Equation 12.4 captures the Jaccard coefficient between the bag of words for the matched text regions, $\texttt{text}(\bar{c}_1)$ and $\texttt{text}(\bar{c}_2)$, that help derive the cell values, $\bar{c}_1$ and $\bar{c}_2$.

Global Similarity

Global similarity computes text similarity by leveraging co-occurrence statistics aggregated over the entire document collection. For instance, we can link the entities referred by the phrases, ***youtube*** and ***video sharing*** based on the co-occurrence counts of $\{$***youtube***, ***video***$\}$ and $\{$***youtube***,***sharing***$\}$. To compute this similarity we leverage the SKIP-GRAM dictionaries that contain the document frequencies (df) of word pairs $\{w_1, w_2\}$. This global text similarity is defined below:

$$\underset{\texttt{global}}{\text{sim}}(\bar{c}_1, \bar{c}_2) = \frac{1}{Z} \cdot \sum_{w_1 \in \text{words}(\bar{c}_1)} \sum_{w_2 \in \text{words}(\bar{c}_2)} \frac{\text{df}(\{w_1, w_2\})}{|D|}, \tag{12.5}$$

where, $Z = |\text{words}(\bar{c}_1)| \cdot |\text{words}(\bar{c}_2)|$ is a normalization constant. The equation above captures the global similarity by computing the co-occurrence frequency of words in the cell values $\bar{c}_1$ and $\bar{c}_2$. The complete text similarity between two cell values $\bar{c}_1$ and $\bar{c}_2$ can now be defined as:

$$\underset{\texttt{text}}{\text{sim}}(\bar{c}_1, \bar{c}_2) = \frac{1}{3}\left[\underset{\texttt{surface}}{\text{sim}}(\bar{c}_1, \bar{c}_2) + \underset{\texttt{context}}{\text{sim}}(\bar{c}_1, \bar{c}_2) + \underset{\texttt{global}}{\text{sim}}(\bar{c}_1, \bar{c}_2)\right]. \tag{12.6}$$

We make the above design choice primarily for scalability reasons. Our method leverages pre-computed word co-occurrence statistics, which avoids computing transformed text representations (e.g., for neural embedding methods) at query time, thus speeding up the similarity computation.

12.7.2 Semantic Model for Numbers

Numerical values in the form of mentions of money, percentages, date, and time can be very vague and uncertain. For instance, the numerical mention in Figure 12.2 is disambiguated to $> \$1 \times 10^9$. This numerical expression can refer to an infinite number of uncertain intervals. Similarly, a temporal expression such as *the 60s* can refer to a multitude of time intervals. Therefore, it becomes essential that we model their uncertainty to compute similarity between numerical values when applying the LINK operator.

To incorporate uncertainty in numerical values, we model a numerical expression, whose values belong to a domain $\Sigma_{\mathcal{N}}$, by associating an interval with it: $[b, e]$, where b denotes the begin and e the end. A temporal expression (or date) can be converted to a numerical expression by representing the dates as UNIX epochs (number of milliseconds passed since 1970-01-01). We model the uncertainty based on the annotation type (e.g., date or numerical) and its value. Figure 12.4 shows how uncertainty in numerical expressions can be modeled. The uncertainty Δ for annotations of DATE and TIME is determined by the difference between two consecutive time elements at a given granularity (e.g., $\Delta =$ *1* year). The uncertainty Δ for the rest of the numerical annotations is equal to half of the value being modeled (e.g., for the PERCENT annotation value of *50%*, Δ is equal to *25%*). The similarity between two numerical cell values $\overline{c}_1$ and $\overline{c}_2$ is:

$$\underset{\texttt{number}}{\text{sim}}(\overline{c}_1, \overline{c}_2) = \frac{|\overline{c}_1 \cap \overline{c}_2|}{|\overline{c}_1 \cup \overline{c}_2|}. \tag{12.7}$$

The denominator in Equation 12.7 represents the number of numerical values at a fixed granularity that can be referenced by the union of the interval representation for $\overline{c}_1$ and $\overline{c}_1$. The numerator computes the extent of agreement in values between $\overline{c}_1$ and $\overline{c}_2$.

12.7.3 Local Resolution of NULL Values

NULL values in the raw table arise if the $\boxed{?\ell}$ operator is used to relax the match for the bindings. Unlike traditional imputation techniques in databases [38, 78] and open-IE approaches, JIGSAW can leverage the provenance of a raw row to infer or estimate the NULL value. To resolve NULL values, we make a *narrative assumption*: the provenance for the row bearing the NULL value is contained in a document that describes a narrative of related events or concepts. For instance, an acquisition made by Google for an undisclosed amount may be described by a news article by comparing it to related acquisitions. To resolve the NULL value locally we describe three methods: scoping, proximity, and semantic redundancy.

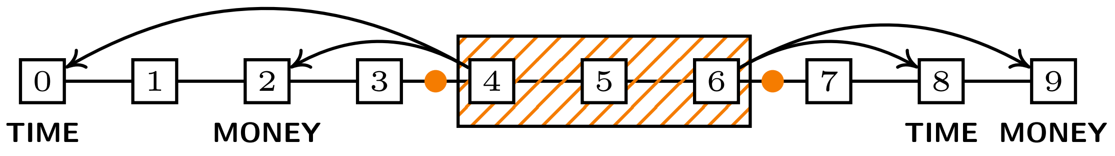


Figure 12.5: Inferring NULL values using the context surrounding the matched text region. Circular nodes represent sentence boundaries. Shaded region corresponds to the matched text region.

Scoping

Scoping for resolving NULL values relies on frequency for named entity annotation type in the document containing the evidence. While, for numbers and time, the NULL value is resolved by constructing an interval using the minimum and maximum values of the same annotation type in the document containing the evidence. For instance, in Figure 12.5 a NULL for annotation type of TIME can be estimated by constructing an interval using the annotation values present on position 0 and 8.

Proximity

Proximity for resolving NULL values considers only nearby annotation values for estimation. This way, we can restrict ourselves to few (e.g., three) nearby values for resolving the NULL values. For example, in Figure 12.5 we can resolve a NULL value for MONEY by looking at only the annotation at position 2 as the first nearest value.

Semantic Redundancy

Semantic redundancy for resolving NULL values considers frequency in semantic models for text or numbers. Thus, for estimating the NULL values for named entity types for PERSON, ORG, and LOC we consider the semantic similarity measures discussed in Section 12.7.1. For estimating the NULL values for annotation types of NUMBERS and TIME, we consider the similarity measures described in Section 12.7.2.

The user can select based on the application domain from the above three methods for filling in the NULL values to yield the best table. For instance, for entity-centric queries, the proximity method, works well (in a manner similar to that of co-reference resolution). For event-centric queries, the semantic redundancy method is more suitable for local resolution of NULL values.

12.7.4 LINK$(\overline{\mathcal{T}}, \{a_1, a_2, \ldots, a_n\}, \theta)$ Operator

The operator LINK $(\overline{\mathcal{T}}, \{a_1, a_2, \ldots, a_n\}, \theta)$ takes as an input the raw table $\overline{\mathcal{T}}$; an attribute set $\{a_1, a_2, \ldots, a_n\}$ to link by; and a threshold θ to determine the degree of similarity between rows. The LINK operator outputs sets of rows that are near-duplicates.

Linking of rows in the raw table $\overline{\mathcal{T}}$ by attributes is done as follows. First, each row in the raw table $\overline{\mathcal{T}}$ is considered related to every other row. That is, we model the raw table as a complete undirected graph. Each undirected edge in the graph is weighted by a similarity value that is computed attribute-wise. That is, corresponding attributes from both rows are compared using Equation 12.6 for text-based attributes and Equation 12.7 for numerical attributes:

$$\mathrm{sim}(\overline{r}_1, \overline{r}_2) = \frac{1}{N} \cdot \sum_{a \in \{a_1,\ldots,a_k\}} \mathrm{sim}\Big(\mathrm{value}(\overline{r}_1, a), \mathrm{value}(\overline{r}_2, a)\Big), \tag{12.8}$$

where, the function value$(\overline{r}, a)$ returns the cell value in the row $\overline{r}$ for the attribute a and N denotes the total number of attributes (or columns in the table). Our model for a row in a table $\overline{r} \in \overline{\mathcal{T}}$ (see Equation 12.2) allows for multiple cell values. To compute similarity between rows that contain multiple cell values for a single attribute, we compute their average pair-wise similarity:

$$\mathrm{sim}(\overline{r}_1, \overline{r}_2) = \frac{1}{N} \cdot \sum_{a \in \{a_1,\ldots,a_k\}} \frac{1}{Y} \cdot \sum_{\overline{c}_1 \in \mathrm{value}(\overline{r}_1, a)} \sum_{\overline{c}_2 \in \mathrm{value}(\overline{r}_2, a)} \mathrm{sim}\big(\overline{c}_1, \overline{c}_2\big),$$

where, $Y = |\mathrm{value}(\overline{r}_1, a)| \cdot |\mathrm{value}(\overline{r}_2, a)|$ is a normalization factor. Note that, the similarity of a cell value to a NULL value, that could not be resolved using local context, is defined to be zero:

$$\mathrm{sim}(\mathsf{NULL}, c) = 0. \tag{12.9}$$

Second, we find connected components in the weighted undirected graph representing $\overline{\mathcal{T}}$. A subgraph is considered connected if each edge in it has an edge weight greater than or equal to a threshold θ. Put another way, connected components can be found by removing all the edges in weighted graph that have weight less than the threshold θ. The remaining subgraphs are then clusters of related rows with respect to their attribute-wise similarity.

12.8 ANALYSIS Operators

The LINK operator groups together near-duplicate raw rows using semantics of text and numbers. The ANALYSIS operators work in conjunction with LINK operators to flatten the group of rows into a representative row and to assign a score to each representative row for ranking. Additionally, when flattening a group of rows we can infer NULL values, those which could not be resolved locally, from other near-duplicate rows (global NULL resolution). We next describe these three operators: FLAT, SCORE, and RANK.

Algorithm 10: FLAT and SCORE operators.

```
   Input   : Connected Component, $S = \{\overline{r}_1, \overline{r}_2, \ldots, \overline{r}_n\}$.
1  Function FLAT($S = \{\overline{r}_1, \overline{r}_2, \ldots, \overline{r}_n\}$)
2      $r_{\text{rep}} \leftarrow \varnothing$ // Create a new representative row for $S$.
3      for $a_i \in r_{rep}$ do
4          value $\leftarrow$ arg max(sim($c_i, \forall c_j \in S$.values($a_i$) $\setminus c_i$))
5          $r_{\text{rep}}$.put(a_i, value)
6      return $r_{\text{rep}}$

   Input   : The Final Table, $\mathcal{T} = \{r_1, r_2, \ldots, r_n\}$.
7  Function SCORE($\mathcal{T}$)
       // Compute support for each row in the final table.
8      for $r \in \mathcal{T}$ do
9          $r$.score = #raw rows forming $r$/#total rows in raw table $\overline{\mathcal{T}}$
10     Sort($\mathcal{T}$) // Sort the rows in table by descending support.
11     $\mathcal{T}_{\text{new}} \leftarrow \varnothing$
12     while $\mathcal{T}$ is not empty do
13         $\mathcal{T}_{\text{new}}$.append(arg max((1 − sim($r_i, \forall r_j \in \mathcal{T}_{\text{new}} \setminus r_i$)/$|\mathcal{T}|$)))
14         remove the row appended to $\mathcal{T}_{\text{new}}$ from $\mathcal{T}$
15     return $\mathcal{T}_{\text{new}}$
```

FLAT($S = \{\overline{r}_1, \overline{r}_2, \ldots, \overline{r}_k\}$) **Operator**. The final table $\mathcal{T}$ consists only of representative rows $r \in \mathcal{T}$ derived from each connected component $\{\overline{r}_1, \overline{r}_2, \ldots, \overline{r}_k\} \in \overline{\mathcal{T}}$ discovered by LINK. Each representative row $r \in \mathcal{T}$, consists of cell values from different rows in the connected component. The selection of the cell values for the representative row is done by computing the similarity of a cell value from a row to all the other cell values for that attribute in the set S. The cell value that is most similar to the others is chosen to be part of the representative row. In this step, the representative row can infer the value for NULLs, that could not be resolved using local context, from other rows' cell values; we refer to this as **global resolution of NULL values**. The global NULL resolution thus resolves NULLs using cross-document evidences. We make the above design choice primarily to resolve NULL values independently from other attributes. Alternative design choices (e.g., selecting the most similar row in its entirety from the group) are less helpful in global NULL resolution. The FLAT operator is described in Algorithm 10.

SCORE($S = \{\overline{r}_1, \overline{r}_2, \ldots, \overline{r}_k\}$) **Operator.** For each connected component $S \in \overline{\mathcal{T}}$ (or $r \in \mathcal{T}$), the SCORE operator assigns a value indicating: the size of S and the novelty of S amongst other connected components in $\overline{\mathcal{T}}$:

$$\texttt{score}(r \in \mathcal{T}) = \text{support}(r \in \mathcal{T}) \cdot \text{diversify}(r \in \mathcal{T}),$$

where, support$(r \in \mathcal{T}) \equiv$ support$(S \in \overline{\mathcal{T}}) = |S|/|\overline{\mathcal{T}}|$. To diversify the representative rows in the final table $\mathcal{T}$, the objective is to order rows in the final table such that a row is highly dissimilar to the rows above it. The SCORE operator is also described in Algorithm 10.

COLLECTION	SIZE (GB)	#DOCUMENTS	#WORDS	#SENTENCES
NYT	49.7	1,855,623	1,058,949,098	54,024,146
GIGAWORD	193.6	9,870,655	3,988,683,648	181,386,746
GDELT	296.2	14,320,457	6,371,451,092	297,861,511

COLLECTION	#PART-OF-SPEECH	#NAMED ENTITY	#TIME	#NUMBERS
NYT	1,058,949,098	107,745,696	15,411,681	21,720,437
GIGAWORD	3,988,683,648	517,420,195	72,247,124	102,299,554
GDELT	6,371,451,092	640,812,778	94,009,542	104,964,085

Table 12.1: Annotated document collection size and statistics.

`RANK` **Operator**. We can order the rows in the final table by two methods. First, we can simply rank $r \in \mathcal{T}$ by the scores generated by the `SCORE` operator: $\text{rank}_{\texttt{score}}$. Second, we can rank by the average length of the annotated text regions (`text`) supporting the row $r \in \mathcal{T}$. Rank by length can be defined as the average inverse length of the annotated text region span:

$$\underset{\texttt{length}}{\text{rank}}(r \in \mathcal{T}) \equiv \underset{\texttt{length}}{\text{rank}}(S \in \overline{\mathcal{T}}) = \frac{1}{|S|} \sum_{\overline{r} \in S} \frac{1}{|\ \texttt{text}(\overline{r})\ |}.$$

Thus, each row in the final table $r \in \mathcal{T}$, contains rows that have been pieced together from partial, redundant, and paraphrased information from the text regions obtained for a structured query $\mathcal{Q}$ from large annotated document collection.

12.9 Evaluation

In this section, we describe the evaluation setup of our experiments.

Annotated Document Collections and their Indexes

To evaluate JIGSAW, we considered three large news archives. The New York Times annotated corpus consists of around two million articles published during 1997-2007 [18]. The fifth edition of the English Gigaword consists of around ten million articles from seven different sources published during 1995-2010 [5]. GDelt news archive comprises of around fourteen million articles related to events available at the GDelt project website [7]. All three document collections, were preprocessed with the Stanford CoreNLP toolkit to obtain annotation for part-of-speech, named entity, temporal expressions, and numerical values. Statistics for the annotated document collections are displayed in Table 12.1. For each collection, we created the five indexes and the direct index (see Section 12.4). We store our indexes using HBase, a distributed and extensible record store. We list the indexes and their sizes for each document collection in Table 12.2.

INDEX TYPE	NYT	GIGAWORD	GDELT
DIRECT INDEX	18.80	52.40	82.30
N-GRAM DICTIONARIES	4.54	10.50	19.04
SKIP-GRAM DICTIONARY	14.40	21.30	29.30
SKIP-GRAM INDEX	56.10	203.60	289.00
N-GRAM INDEXES	45.90	154.40	234.80
ANNOTATION INDEXES	2.39	9.33	16.03
2-FRAGMENT INDEXES	6.30	24.16	36.84
2-STITCH INDEXES	141.00	542.40	677.10

Table 12.2: Index sizes in Gigabytes (GB).

CATEGORY	#ENTITIES	PREDICATE	BINDINGS	EXAMPLE ENTITY
OLYMPIANS	265	PARTICIPANT OF	!LOCATION, ?TIME	*Usain Bolt*
MARRIAGE	237	SPOUSE	!PERSON , ?TIME	*Bob Dylan*
FOOTBALLERS	101	SPORTS TEAM	!ORG , ?TIME	*Kaká*
CEOS	87	CEO	!PERSON , ?TIME	*Google*
ACQUISITIONS	58	ACQUIRE	!ORG , ?TIME, ?MONEY	*Takeda*

Table 12.3: Query testbed statistics.

Jigsaw Puzzles (Query Testbed)

JIGSAW can provide multiple answers in the form of a table for a query where many answers are correct (e.g., *google acquisitions*). To measure the quality of the generated table, we need to identify queries for which we need to link text, time, or numbers. To this end, we constructed a testbed of 748 tabular queries concerning acquisitions, CEOs, Olympians, footballers, and marriages for popular entities. We obtained companies and their acquisitions from CrunchBase [3]. These prominent 58 organizations were identified using Fortune 500 [6], large software [14] and manufacturing [12] companies lists. For the remaining categories, we constructed the tables from the Wikidata Knowledge Graph. Specifically to generate these queries, we restrict ourselves to prominent named entities in Wikidata. Where, the prominence of an entity in Wikidata is determined using the concept of *sitelinks* [21]. To instantiate the structured queries, we use the following query template: ⟨ENTITY, PREDICATE, UNORDERED BINDINGS⟩. The template is then filled with aliases of entities (e.g., [*google* | *search giant*]), paraphrases for the predicates (e.g., [*acquired* | *takeover*]) from Wikidata and bindings for the requisite category (e.g., ORG, TIME, and MONEY for acquisitions). For example, a query for ACQUISITIONS (see Table 12.3) is shown below:

$$\Big\langle ![\textit{google}\,|\,\textit{google inc.}\,|\,\textit{google llc}], ![\textit{acquires}\,|\,\textit{acquired}\,|\,\textit{acquisition}\,|\,\textit{takeover}\,|\,\textit{bought}\,|\,\textit{buys}\,|\,\textit{scoops up}\,|\,\textit{to buy}\,|\,\textit{slurps}], \{!\text{ORG}, ?\text{TIME}, ?\text{MONEY}\}\Big\rangle.$$

Quality of Generated Tables

Quality of the generated tables is measured by comparing against the ground truth extracted from Crunchbase and Wikidata. That is, for each row in the ground truth table $r_{\text{true}} \in \mathcal{T}_{\text{true}}$, we seek its equivalent row in the generated table and compare them in their respective semantic models of representation. We can therefore establish the notions of precision and recall using the semantic similarity measures discussed in Section 12.7.

Precision between the generated table $\mathcal{T}$ and the ground truth table $\mathcal{T}_{\text{true}}$ is computed by checking that each row in the generated table $r \in \mathcal{T}$ finds a corresponding equivalent row in the ground truth table $r_{\text{true}} \in \mathcal{T}_{\text{true}}$. This can be written as:

$$\text{precision}(\mathcal{T}, \mathcal{T}_{\text{true}}) = \frac{1}{|\mathcal{T}|} \sum_{r \in \mathcal{T}} \operatorname*{argmax}_{r_{\text{true}} \in \mathcal{T}_{\text{true}}} \text{sim}(r, r_{\text{true}}), \tag{12.10}$$

where, the similarity between two rows, $\text{sim}(r, r_{\text{true}})$, is computed using Equation 12.8 described in Section 12.7.4. For text based attributes, we seek the maximum similarity match between the generated cell attribute and possible aliases in the ground truth.

Recall between the generated table $\mathcal{T}$ and the ground truth table $\mathcal{T}_{\text{true}}$ measures how many rows from the ground truth table are found in the generated table. This can be stated as:

$$\text{recall}(\mathcal{T}, \mathcal{T}_{\text{true}}) = \frac{1}{|\mathcal{T}_{\text{true}}|} \sum_{r_{\text{true}} \in \mathcal{T}_{\text{true}}} \operatorname*{argmax}_{r \in \mathcal{T}} \text{sim}(r, r_{\text{true}}). \tag{12.11}$$

12.9.1 Setup

Currently, there exists no system that can generate tables for user-defined schema over annotated document collections. To evaluate JIGSAW, we consider alternative design choices using various building blocks in our system.

Baseline Basic resolves NULL values for temporal and numerical expressions using *scoping*. Linking of rows, relies on surface level similarity and redundancy. Flattening of connected components is done by considering the most frequent textual cell value; while for numbers and time, minimum and maximum of the cell values in the group is considered. For ranking, only support is considered.

Baseline Advanced resolves NULL values for temporal and numerical expressions using *proximity*. Baseline Advanced links rows based on text using surface and contextual similarity as discussed in Section 12.7.1. Whereas, linking for numerical expressions is based on simple interval overlaps. Flattening of connected components is done by considering the most frequent textual cell value, while for numbers and time the most frequent interval is chosen as representative. For ranking, here also only support is considered.

Systems JIGSAW and JIGSAW++, are our proposed systems. JIGSAW applies the semantic redundancy method for local NULL value resolution. JIGSAW++ considers the semantic redundancy method for event-centric queries (i.e., acquisitions and Olympians) and proximity method for entity-centric queries (i.e., CEOs, footballers, and marriages). For flattening a group of rows, JIGSAW considers semantic redundancy and relatedness for numerical values and frequency for text attributes. JIGSAW++ considers the same method for event-centric queries. For entity-centric queries JIGSAW++ considers the frequency based method for both text and numerical attributes. For ranking, both support and diversity are considered.

Significance Tests results between the baseline Basic and JIGSAW are shown by $\triangle$. Statistically significant results between baseline Advanced and JIGSAW are marked by $\blacktriangle$. The significance was computed using the two-tailed paired t-Test at $\alpha = 0.05$.

Hardware. The storage for the indexes is a cluster of twenty machines running Cloudera CDH 5.90 version of Hadoop and HBase. Each machine in the Hadoop cluster is equipped with up to a 24 core Intel Xeon CPU with 3.50 GHz processing speed, up to 128 GB of RAM, and up to eight 4 TB worth of secondary storage. We perform all evaluations on a high-memory compute node, with 96 core Intel Xeon CPU at 2.66 GHz processing speed and 1.48 TB of RAM.

12.9.2 Results

We executed the baselines and systems for all of the queries in each query category in Table 12.3. We considered three different values of similarity thresholds $\theta \in \{0.25, 0.50, 0.75\}$ for the LINK operator, to determine its best value. We then computed precision, recall, and F_1 (harmonic mean of precision and recall) at values of top-$k \in \{10, 25, 50\}$. The results for precision and recall averaged over all query categories for each collection are shown in Table 12.4 and 12.5. We summarize the F_1 values over all collections in Table 12.6. The best value of the measures is highlighted for the corresponding threshold value. For the collections NYT, Gigaword, and GDelt we were able to generate 344, 444, and 414 tables respectively. The number of tables vary per collection due to varying time periods of their reporting. For example, NYT does not contain any acquisitions for Twitter whereas Gigaword and GDelt do.

Effectiveness Results

In terms of precision (see Table 12.4), we observe that overall JIGSAW achieves the best performance. In terms of recall (see Table 12.5), we observe that JIGSAW performs at par or better when compared to the baselines. Note that perfect recall is not possible due to: temporal coverage of collections and KG incompleteness. When considering the F_1 score (see Table 12.6), JIGSAW provides a balanced performance of high precision and good recall as compared

New York Times									
TOP-k	**10**			**25**			**50**		
θ	**0.25**	**0.50**	**0.75**	**0.25**	**0.50**	**0.75**	**0.25**	**0.50**	**0.75**
BASIC	0.04	0.04	0.23	0.02	0.02	0.17	0.01	0.01	0.12
ADVANCED	0.04	0.17	0.23	0.01	0.10	0.18	0.01	0.06	0.14
JIGSAW	0.05 △▲	0.22 △▲	0.24	0.02 ▲	0.16 △▲	0.20 △▲	0.01	0.12 △▲	0.17 △▲
JIGSAW++	0.05 △▲	0.22 △▲	0.25	0.02 ▲	0.17 △▲	0.21 △▲	0.01	0.13 △▲	0.18 △▲

Gigaword									
TOP-k	**10**			**25**			**50**		
θ	**0.25**	**0.50**	**0.75**	**0.25**	**0.50**	**0.75**	**0.25**	**0.50**	**0.75**
BASIC	0.04	0.04	0.27	0.01	0.02	0.23	0.01	0.01	0.18
ADVANCED	0.04	0.22	0.29	0.02	0.15	0.25	0.01	0.09	0.21
JIGSAW	0.05 △▲	0.27 △▲	0.26	0.02 △	0.23 △▲	0.25	0.01	0.20 △▲	0.23 △▲
JIGSAW++	0.05 △▲	0.28 △▲	0.27	0.02 △	0.24 △▲	0.25	0.01	0.20 △▲	0.24 △▲

GDelt									
TOP-k	**10**			**25**			**50**		
θ	**0.25**	**0.50**	**0.75**	**0.25**	**0.50**	**0.75**	**0.25**	**0.50**	**0.75**
BASIC	0.04	0.05	0.27	0.02	0.02	0.19	0.01	0.01	0.15
ADVANCED	0.04	0.17	0.27	0.02	0.12	0.21	0.01	0.08	0.17
JIGSAW	0.06 △▲	0.24 △▲	0.30 △▲	0.02	0.19 △▲	0.26 △▲	0.01	0.15 △▲	0.24 △▲
JIGSAW++	0.06 △▲	0.25 △▲	0.31 △▲	0.02	0.20 △▲	0.27 △▲	0.01	0.15 △▲	0.25 △▲

Table 12.4: Precision over all categories in the testbed.

to the baselines that excel only in recall. The good performance of JIGSAW can be attributed to three key system design choices. First, the baselines are more sensitive to θ as they rely only on surface (Baseline Basic) or contextual (Baseline Advanced) similarities for text. Whereas, JIGSAW leverages surface, contextual, and global text similarities to link rows. Second, JIGSAW uses semantic redundancy for NULL value resolution that provides higher precision. However, the NULL resolution techniques utilized by the baselines produce less reliable estimates for temporal and numerical attributes. Third and finally, the techniques adopted by the baselines for flattening linked rows result in broader temporal and numerical representations that are less precise. In addition to this, JIGSAW is more robust and achieves higher precision at different similarity thresholds θ.

New York Times

TOP-k	10			25			50		
θ	**0.25**	**0.50**	**0.75**	**0.25**	**0.50**	**0.75**	**0.25**	**0.50**	**0.75**
BASIC	0.31	0.32	0.38	0.31	0.32	0.39	0.31	0.32	0.40
ADVANCED	0.29	0.35	0.36	0.29	0.36	0.38	0.29	0.36	0.39
JIGSAW	0.29	0.35 △	0.33	0.29	0.36 △	0.35	0.29	0.37 △▲	0.35
JIGSAW++	0.30 ▲	0.36 △▲	0.36	0.30 ▲	0.38 △▲	0.38	0.30 ▲	0.39 △▲	0.39

Gigaword

TOP-k	10			25			50		
θ	**0.25**	**0.50**	**0.75**	**0.25**	**0.50**	**0.75**	**0.25**	**0.50**	**0.75**
BASIC	0.29	0.30	0.40	0.29	0.30	0.43	0.29	0.30	0.44
ADVANCED	0.29	0.37	0.39	0.29	0.38	0.42	0.29	0.38	0.44
JIGSAW	0.29	0.37 △	0.35	0.29	0.40 △▲	0.37	0.29	0.42 △▲	0.39
JIGSAW++	0.30 ▲	0.39 △▲	0.37	0.30 ▲	0.42 △▲	0.41	0.30 △▲	0.44 △▲	0.43

GDelt

TOP-k	10			25			50		
θ	**0.25**	**0.50**	**0.75**	**0.25**	**0.50**	**0.75**	**0.25**	**0.50**	**0.75**
BASIC	0.32	0.32	0.40	0.32	0.32	0.42	0.32	0.32	0.43
ADVANCED	0.32	0.37	0.38	0.32	0.39	0.41	0.32	0.39	0.43
JIGSAW	0.31	0.38 △	0.37	0.31	0.40 △	0.39	0.31	0.42 △▲	0.41
JIGSAW++	0.32	0.39 △▲	0.40	0.32	0.42 △▲	0.43	0.32	0.44 △▲	0.44

Table 12.5: Recall over all categories in the testbed.

OVERALL

TOP-k	10			25			50		
θ	**0.25**	**0.50**	**0.75**	**0.25**	**0.50**	**0.75**	**0.25**	**0.50**	**0.75**
BASIC	0.07	0.08	0.30	0.03	0.03	0.25	0.02	0.02	0.21
ADVANCED	0.07	0.24	0.30	0.03	0.17	0.26	0.01	0.12	0.23
JIGSAW	0.09 △▲	0.28 △▲	0.30	0.04 △▲	0.24 △▲	0.28 △▲	0.02 ▲	0.21 △▲	0.27 △▲
JIGSAW++	0.09 △▲	0.29 △▲	0.31 △	0.04 △▲	0.25 △▲	0.29 △▲	0.02 ▲	0.22 △▲	0.28 △▲

Table 12.6: F_1 for all testbed categories and all collections.

NYT	GIGAWORD	GDELT
111.00	396.00	604.00

Table 12.7: Time in seconds taken to scan a collection on our cluster.

COLLECTION	BASIC	ADVANCED	JIGSAW	JIGSAW++
NYT	3.68 ± 6.16	3.86 ± 6.85	7.63 ± 15.63	7.95 ± 16.28
GIGAWORD	8.81 ± 11.12	9.09 ± 10.84	16.30 ± 20.85	16.45 ± 22.09
GDELT	17.33 ± 35.15	17.90 ± 36.22	28.37 ± 43.48	27.99 ± 42.56

Table 12.8: Run-times in seconds for our system JIGSAW.

Efficiency Results

We evaluated JIGSAW for efficiency by executing a sample of 100 queries from the query testbed three times in cold-cache setting for each collection. To simulate cold caches, we shuffle the queries in between rounds. To contrast the performance of our query-driven system with existing open-IE systems: we measure the time needed to scan the entire collection on our Hadoop cluster once. This thus simulates the minimum amount of time an open-IE system shall take to just retrieve all the sentences for a query in an embarrassingly parallel manner. The results for the scan baseline are shown in Table 12.7. The results for the end-to-end run-times for our system JIGSAW and the baselines Basic and Advanced are shown in Table 12.8. From Table 12.7 we see that scanning the entire collection for each query is in the order of minutes. From Table 12.8, we see that end-to-end run times for generating tables using the baselines or our system JIGSAW are significantly less than a simple scan. From Table 12.8, we observe that JIGSAW takes more time to generate tables than the baselines. This is because the baselines only leverage surface-level and contextual similarities that are quick to compute as they only require in-memory operations. On the other hand, JIGSAW additionally leverages global similarity, that relies on lookups from the dictionaries. This additional time however provides us improved linking of raw rows. Overall, we see that we gain at least an order of 13.96× speedup and at most an order of 44.95× speedup over a simple scan of the collections.

12.10 Conclusion

We presented JIGSAW, a system that generates tables from unstructured text for user-defined schema. To do so, we described `QUERY` operators to define the table schema. To speedup query processing we described a greedy query optimizer. To generate high-quality tables, we described `LINK` and `ANALYSIS` operators that leverage semantic models for text and numbers to group together near-duplicate rows. Our evaluation demonstrates that JIGSAW can generate high-quality tables from over 25 million documents efficiently.

12.11 Anecdotal Results

We next present tables, as anecdotal results, generated by JIGSAW for a few structured queries from our testbed.

NO.	SCORE	ORG	TIME	MONEY
1.	0.511	*micros systems*	$[2014, 2015]$	$[\$1.36 \times 10^8 , \$1.90 \times 10^{10}]$
2.	0.092	*cisco*	$[2010, 2011]$	$[\$3.00 \times 10^9 , \$9.00 \times 10^9]$
3.	0.021	*homestead technologies*	$[2012, 2013]$	$[\$3.50 \times 10^9 , \$3.50 \times 10^9]$
4.	0.016	*datalogix*	$[2014, 2015]$	$[\$4.68 \times 10^8 , \$1.40 \times 10^9]$
5.	0.012	*tekelec*	$[2013, 2014]$	$[\$3.00 \times 10^9 , \$9.00 \times 10^9]$
6.	0.012	*ericsson*	$[2004, 2005]$	$[\$1.80 \times 10^8 , \$2.70 \times 10^8]$
7.	0.009	*cnw group/manulife financial corporation*	$[2011, 2014]$	$[\$1.05 \times 10^{10} , \$1.05 \times 10^{10}]$
8.	0.008	*pillar data systems*	$[2011, 2012]$	$[\$8.04 \times 10^2 , \$8.04 \times 10^2]$
9.	0.008	*blue kai*	$[2010, 2011]$	$[\$3.10 \times 10^9 , \$3.10 \times 10^9]$
10.	0.008	*rightnow technologies*	$[2013, 2014]$	$[\$3.50 \times 10^7 , \$3.50 \times 10^7]$

Figure 12.6: Top-10 rows of the table generated by JIGSAW from the GDelt news archive for acquisitions made by Oracle. The structured query used to define the table schema was: $\langle$![*oracle corporation*|*oracle*], ![*acquires*|*acquired*|*acquisition*|*takeover*|*bought*|*buys*|*scoops up*|*to buy*|*slurps*], {!ORG, ?TIME, ?MONEY}$\rangle$.

NO.	SCORE	PERSON	TIME
1.	0.990	*mark zuckerberg*	$[2010, 2018]$
2.	0.003	*sheryl sandberg*	$[2013, 2014]$
3.	0.002	*obama*	$[2014, 2014]$
4.	0.001	*durov*	$[2013, 2013]$
5.	0.001	*benjamin joe*	$[2014, 2015]$
6.	0.001	*elon musk*	$[2013, 2014]$

Figure 12.7: Table generated by JIGSAW from the GDelt news archive for the CEO of Facebook. The structured query used to define the table schema was: $\langle$![*facebook inc.* |*facebook*|*fb*], ![*chief executive*|*ceo*|*executive director*], {!PERSON, ?TIME}$\rangle$. The table correctly lists Mark Zuckerberg as the CEO of Facebook. The table also lists Sheryl Sandberg, the COO of the company. Other rows that are erroneously listed have insignificant scores.

NO.	SCORE	PERSON	TIME
1.	0.851	*winnie madikizela*	$[2010, 2010]$
2.	0.029	*evelyn ntoko*	$[1957, 1995]$
3.	0.007	*thabo mbeki*	$[2003, 2003]$
4.	0.005	*zondi*	$[1955, 2005]$
5.	0.003	*nompumelelo ntuli zuma*	$[1973, 2010]$
6.	0.003	*nkosazana zuma*	$[1995, 1995]$
7.	0.002	*elita georgiades*	$[1998, 1998]$
8.	0.002	*philip*	$[1960, 1996]$
9.	0.002	*desmond tutu*	$[2010, 2010]$
10.	0.002	*kibaki*	$[2009, 2009]$

Figure 12.8: Top-10 rows of the table generated by JIGSAW from the Gigaword news archive for spouses of Nelson Mandela. The structured query used to define the table schema was: $\langle ![\textit{nelson mandela} | \textit{nelson rolihlahla mandela} | \textit{mandela} | \textit{madiba} | \textit{prisoner 46664 x vidios}], ![\textit{husband} | \textit{wife} | \textit{married to} | \textit{consort} | \textit{partner} | \textit{marry} | \textit{marriage partner} | \textit{married} | \textit{wedded to} | \textit{wed} | \textit{life partner}], \{!\text{PERSON}, ?\text{TIME}\} \rangle$. The generated table correctly lists two of his spouses in the first two rows: Winnie Madikizela and Evelyn Ntoko.

CHAPTER 13

DIGITALHISTORIAN: SEARCH AND ANALYTICS USING ANNOTATIONS

• Dhruv Gupta, Jannik Strötgen, and Klaus Berberich. DIGITALHISTORIAN: Search & Analytics Using Annotations. In HistoInformatics 2016 co-located with DH 2016, pages 5–10.

13.1 Introduction

Large born-digital document collections cannot be analyzed by manual human effort. However, they can be automatically annotated with temporal expressions and named entities for their navigation. Time has been found to be a very important part of generic Web queries; recent studies [130] estimate that around 17.1% of them are implicitly time-sensitive in nature. In the context of digital humanities, these figures can be expected to be much higher. The desiderata that we believe many commercial search engines do not currently meet and which will be useful for scholars in digital humanities are:

1. the ability to automatically suggest interesting time intervals for history-oriented queries;

2. the ability to diversify or re-rank documents using temporal expressions;
3. the ability to establish relationships between the time intervals of interest for query and other evidences in text such as named entities;
4. the ability to visually analyze the different relationships established between the annotations in documents.

In this chapter, we demonstrate DigitalHistorian, a system that leverages the semantic information in documents to retrieve better search results for history-oriented queries. We define history-oriented queries to consist of keywords that describe an entity (e.g., *george w. bush*) or an event (e.g., *economic depression*). Our search system analyzes temporal expressions in documents to identify interesting time intervals, and subsequently uses them for diversifying search results. The interesting time intervals can also be selected to expand the query to gather search results concerning that particular time interval. Furthermore, DigitalHistorian can construct visualizations that display frequent named entities in interesting time intervals identified for the history-oriented query.

Organization. The sections that are concerned with the DigitalHistorian system are as follows. We first put our system in context to related systems in Section 13.2. In Section 13.3, we give a brief description of the key methods applied for mining interesting time intervals, and how these are used for search result re- ranking and diversification. In Section 13.4, we describe the technical building blocks of DigitalHistorian. In Section 13.5, we describe how DigitalHistorian can be utilized to explore document collections. We summarize the contributions presented in this chapter in Section 13.6.

13.2 Related Work

There exists few demonstrations that utilize named entities and temporal expressions for search and analysis of documents. None of them put any emphasis on the understanding and analysis of temporal expressions in document contents as we have done in this chapter. Some of the challenges that we addressed, have been described in detail by Tahmesebi et al. from the digital humanities perspective [192]. Hoffart et al. [117] demonstrated a search system that utilized named entities in the Yago knowledge graph for retrieval of documents. To this end, they use named entities and their categorical types for auto-complete suggestion to queries. They, however, rank documents based on publication dates. In their subsequent work, Hoffart et al. [116] perform analytics by using a combination of document publication dates and the entities contained therein. Yeung and Jatowt [149] use LDA topics over time in text to assist historians in answering various queries. In contrast to these systems, we have looked at temporal expressions in document contents in order to generate a deeper analysis for search results. Similarly, Strötgen and Gertz [187] extract content temporal expressions and geographic locations to anchor news articles on a map, and Odjik et al. [169] present an interface to explore different document collections using temporal expressions and text. However unlike both these systems, we have utilized disambiguated named entities in a knowledge graph to contextualize interesting time intervals. Other systems such as WAHSP [121] use sentiment in text, and the system HISTOGRAPH uses *social relations* in photographic collections [167].

13.3 History by Algorithms

The underlying methods for temporal search in DIGITALHISTORIAN are derived from our prior research [41, 93, 94, 96]. We next give a brief description of these methods.

Understanding Time. Temporal expressions in documents can be highly uncertain, for example `1990s`. For such temporal expressions it is unclear how the time interval should be constructed for further analysis. In order to represent such ambiguities in time, we use the time model proposed by Berberich et al. [41] which allows for relaxations on the begin and end of time intervals. Thus, `1990s` may convey a time interval that can begin anywhere from `[1990,1999]` and end anywhere from `[1990, 1999]`. This new representation thus allows us to perform mathematical manipulations on uncertain and ambiguous temporal expressions.

Time Intervals of Interest to Queries. Given a history-oriented query such as *`george w. bush`*, our approach [93] can identify interesting time intervals (e.g., `[2000,2004], [2004,2008]`) by analyzing the temporal expressions in its pseudo-relevant set of documents. This is achieved in two steps. First, by counting the frequency of time intervals in the uncertainty-aware time model described earlier. Second, by weighting each frequent time interval with relevance of the document to the query. This is done recursively to generate interesting time intervals at year, month, and day level granularity.

Re-ranking Documents Using Time. The ranking of the initial set of pseudo-relevant documents can be refined by using one of the interesting time intervals for query expansion. Consider for example the query *`george w. bush`* reformulated with the time interval `[2000,2004]`. The documents which contain temporal expressions that can generate the time interval in the query more frequently and also have a higher textual relevance to the query will be promoted in the rankings [41]. Hence, all documents that are relevant to the time interval will be higher in the rankings.

Diversifying Documents Using Time. The time intervals of interest can be considered to reflect different temporal aspects underlying the query. The initial pseudo-relevant set of documents can then be diversified so as to contain at least one document relevant to the different temporal aspects [96]. The temporally diverse set of documents can thus be viewed as a biography of an entity or a timeline of an event.

Counting Frequent Named Entities. The time intervals of interest can further be used as a basis for aggregating different annotations in text. In particular, we aggregate the occurrence of unique named entities. For example, for the query *`george w. bush`* and the time interval `[2000,2004]`, we obtain the aggregate counts of co-occurring named entities such as AL GORE.

13.4 Architecture

The key building blocks of DIGITALHISTORIAN are: a document collection, semantic annotators, an information retrieval framework, a visualization engine, and the graphical user interface. We describe each of them briefly in the following paragraphs.

Document Collection. We used the The New York Times Annotated Corpus [18], a collection of news articles published in The New York Times between 1987 to 2007. It comprises of roughly two million news articles. As metadata, we used only the publication dates.

Semantic Annotators. For annotating temporal expressions we utilized the HeidelTime temporal tagger [188]. HeidelTime annotates implicit, explicit, and relative temporal expressions. For disambiguating and linking named entities to the Yago[190] knowledge graph, all documents were processed with Aida [118].

Information Retrieval Framework. All the documents along with their annotations were indexed with the ElasticSearch [4] framework. As a baseline retrieval model for pseudo-relevant documents we used the Okapi BM25 method implemented in ElasticSearch. All our methods for temporal search and aggregation were implemented in the Java language.

Visualization and GUI. For generating visualizations, we used the Brunel Visualization[1] API. The entire graphical user interface was programmed using Java's Swing API.

13.5 Demonstration

As outlined in the following, there are two key use cases that we demonstrate with DIGITALHISTORIAN. We also describe how the users will be able to interact with DIGITALHISTORIAN using illustrations for the different use cases.

Exploring Search Results. The foremost task that we address with DIGITALHISTORIAN is that of exploring the document collection using the interesting time intervals identified for the query. The main view (Figure 13.1) of the DIGITALHISTORIAN addresses this by providing the user with a search field to issue keyword queries. Subsequent to the search operation, various interesting time intervals are displayed in a list on the left hand-side of the interface. The list of time intervals are ordered by their interestingness, i.e.,

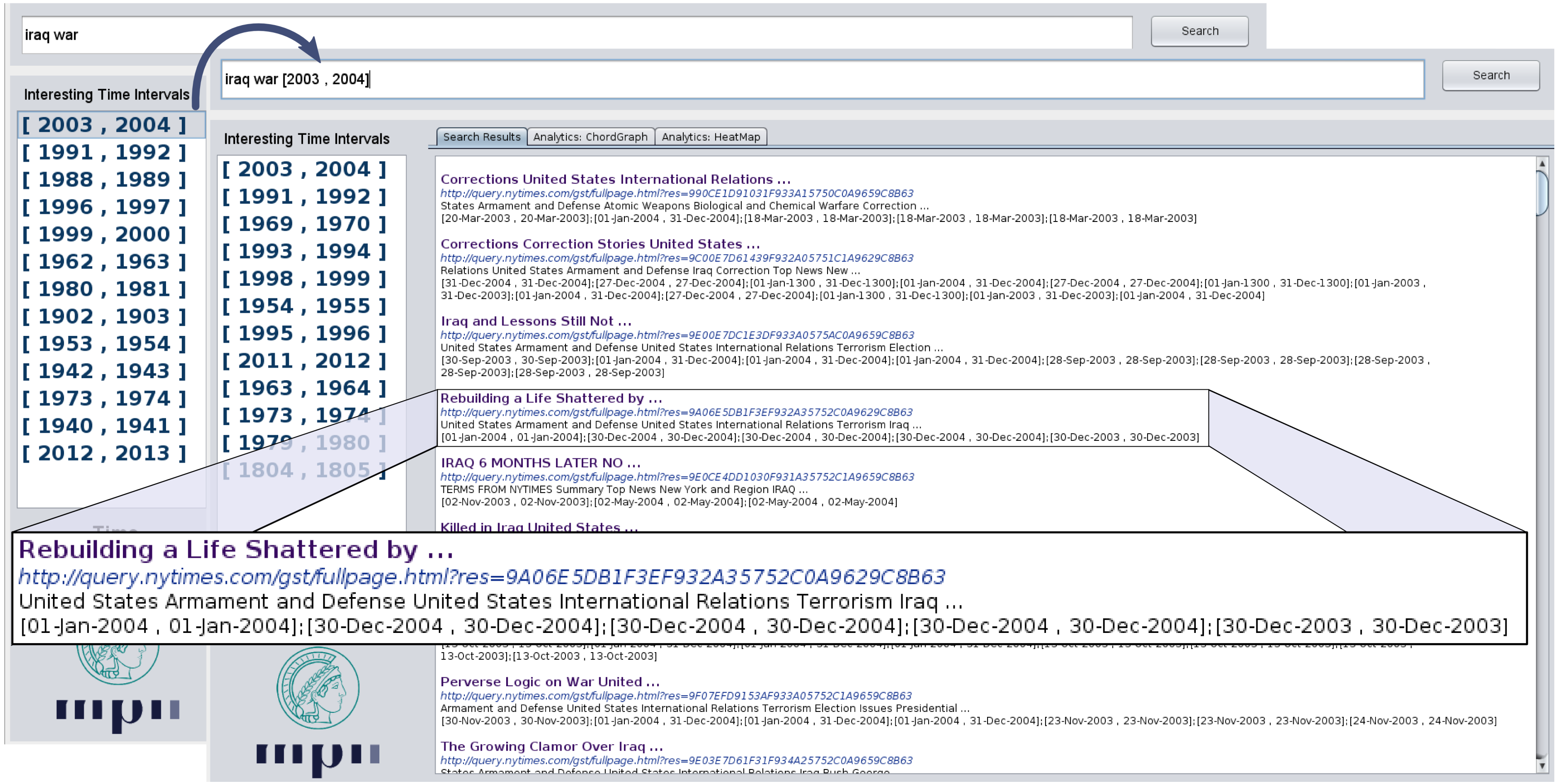
[1]`https://github.com/Brunel-Visualization/Brunel`

how frequently they are generated by the temporal expressions in document contents for the given query. A diversified set of documents is shown in the main display which gives a temporal overview of documents for the query. Each document in the list is depicted by its headline, its URL, a snippet from its contents, and the normalized temporal expressions in its contents. Furthermore, the users can double-click the various time intervals in the list to expand the query, so as to obtain more documents concerning it. Unlike many commercial search engines, all of this is done automatically, without imposing any sliders or check-boxes to manually specify relevant time intervals.

Analytical Visualizations. We further construct informative visualizations by contextualizing the interesting time intervals with co-occurring named entities. Currently, there are two analytical visualizations available. Both of them show the frequency of various named entities that occur in different time intervals. The first visualization is a chord diagram (Figure 13.2), where an arc is drawn between a time interval and a corresponding named entity that occurs in that time interval. The thickness of the arc is in proportion to the frequency of the named entity in that time interval. The user can also hover over to each individual chord in the graph to see the time interval and the entity it connects and the corresponding aggregate count. The second visualization is a heatmap diagram (Figure 13.3), where on the x-axis the different time points and on the y-axis different named entities are plotted. The intensity of the cell in the heatmap shows the frequency of that named entity in that time point. The user can further drill up from years to decades and drill down from years to days by scrolling on the time axis to inspect the different time intervals with their respective frequency of named entities.

13.6 Conclusion

In this chapter, we demonstrated DigitalHistorian, a system that is able to analyze temporal expressions in document content to generate interesting time intervals which are subsequently used to re-rank and diversify documents to give a historic overview for the issued query. It also offers capabilities to analyze frequent named entities in the Yago knowledge graph for informative visualizations. DigitalHistorian thus provides scholars in digital humanities an informative and innovative way of exploring semantically annotated document collections.



[2003 , 2004]
[1991 , 1992]
[1988 , 1989]
[1996 , 1997]
[1999 , 2000]
[1962 , 1963]
[1980 , 1981]
[1902 , 1903]
[1953 , 1954]
[1942 , 1943]
[1973 , 1974]
[1940 , 1941]
[2012 , 2013]

[2003 , 2004]
[1991 , 1992]
[1969 , 1970]
[1993 , 1994]
[1998 , 1999]
[1954 , 1955]
[1995 , 1996]
[2011 , 2012]
[1963 , 1964]
[1973 , 1974]
[1979 , 1980]
[1804 , 1805]

Corrections United States International Relations ...
http://query.nytimes.com/gst/fullpage.html?res=990CE1D91031F933A15750C0A9659C8B63
States Armament and Defense Atomic Weapons Biological and Chemical Warfare Correction ...
[20-Mar-2003 , 20-Mar-2003];[01-Jan-2004 , 31-Dec-2004];[18-Mar-2003 , 18-Mar-2003];[18-Mar-2003 , 18-Mar-2003];[18-Mar-2003 , 18-Mar-2003]

Corrections Correction Stories United States ...
http://query.nytimes.com/gst/fullpage.html?res=9C00E7D61439F932A05751C1A9629C8B63
Relations United States Armament and Defense Iraq Correction Top News New ...
[31-Dec-2004 , 31-Dec-2004];[27-Dec-2004 , 27-Dec-2004];[01-Jan-1300 , 31-Dec-1300];[01-Jan-2004 , 31-Dec-2004];[27-Dec-2004 , 27-Dec-2004];[01-Jan-1300 , 31-Dec-1300];[01-Jan-2003 , 31-Dec-2003];[01-Jan-2004 , 31-Dec-2004];[27-Dec-2004 , 27-Dec-2004];[01-Jan-1300 , 31-Dec-1300];[01-Jan-2003 , 31-Dec-2003];[01-Jan-2004 , 31-Dec-2004]

Iraq and Lessons Still Not ...
http://query.nytimes.com/gst/fullpage.html?res=9E00E7DC1E3DF933A0575AC0A9659C8B63
United States Armament and Defense United States International Relations Terrorism Election ...
[30-Sep-2003 , 30-Sep-2003];[01-Jan-2004 , 31-Dec-2004];[01-Jan-2004 , 31-Dec-2004];[01-Jan-2004 , 31-Dec-2004];[28-Sep-2003 , 28-Sep-2003];[28-Sep-2003 , 28-Sep-2003];[28-Sep-2003 , 28-Sep-2003];[28-Sep-2003 , 28-Sep-2003]

Rebuilding a Life Shattered by ...
http://query.nytimes.com/gst/fullpage.html?res=9A06E5DB1F3EF932A35752C0A9629C8B63
United States Armament and Defense United States International Relations Terrorism Iraq ...
[01-Jan-2004 , 01-Jan-2004];[30-Dec-2004 , 30-Dec-2004];[30-Dec-2004 , 30-Dec-2004];[30-Dec-2004 , 30-Dec-2004];[30-Dec-2003 , 30-Dec-2003]

IRAQ 6 MONTHS LATER NO ...
http://query.nytimes.com/gst/fullpage.html?res=9E0CE4DD1030F931A35752C1A9659C8B63
TERMS FROM NYTIMES Summary Top News New York and Region IRAQ ...
[02-Nov-2003 , 02-Nov-2003];[02-May-2004 , 02-May-2004];[02-May-2004 , 02-May-2004]

Killed in Iraq United States ...

13-Oct-2003];[13-Oct-2003 , 13-Oct-2003]

Perverse Logic on War United ...
http://query.nytimes.com/gst/fullpage.html?res=9F07EFD9153AF933A05752C1A9659C8B63
Armament and Defense United States International Relations Terrorism Election Issues Presidential ...
[30-Nov-2003 , 30-Nov-2003];[01-Jan-2004 , 31-Dec-2004];[01-Jan-2004 , 31-Dec-2004];[01-Jan-2004 , 31-Dec-2004];[23-Nov-2003 , 23-Nov-2003];[23-Nov-2003 , 23-Nov-2003];[24-Nov-2003 , 24-Nov-2003]

The Growing Clamor Over Iraq ...
http://query.nytimes.com/gst/fullpage.html?res=9E03E7D61F31F934A25752C0A9659C8B63

Rebuilding a Life Shattered by ...
http://query.nytimes.com/gst/fullpage.html?res=9A06E5DB1F3EF932A35752C0A9629C8B63
United States Armament and Defense United States International Relations Terrorism Iraq ...
[01-Jan-2004 , 01-Jan-2004];[30-Dec-2004 , 30-Dec-2004];[30-Dec-2004 , 30-Dec-2004];[30-Dec-2004 , 30-Dec-2004];[30-Dec-2003 , 30-Dec-2003]

Figure 13.1: The GUI of DigitalHistorian. Users can type in keyword queries in the search text field and DigitalHistorian will automatically determine interesting time intervals for it. The users can also double-click one of the many intervals in the list to expand the query and retrieve the search results with that time interval. In the illustration it's shown how the user selects the time interval [2003,2004] to expand the query *iraq war* and obtains search results for that particular time interval.

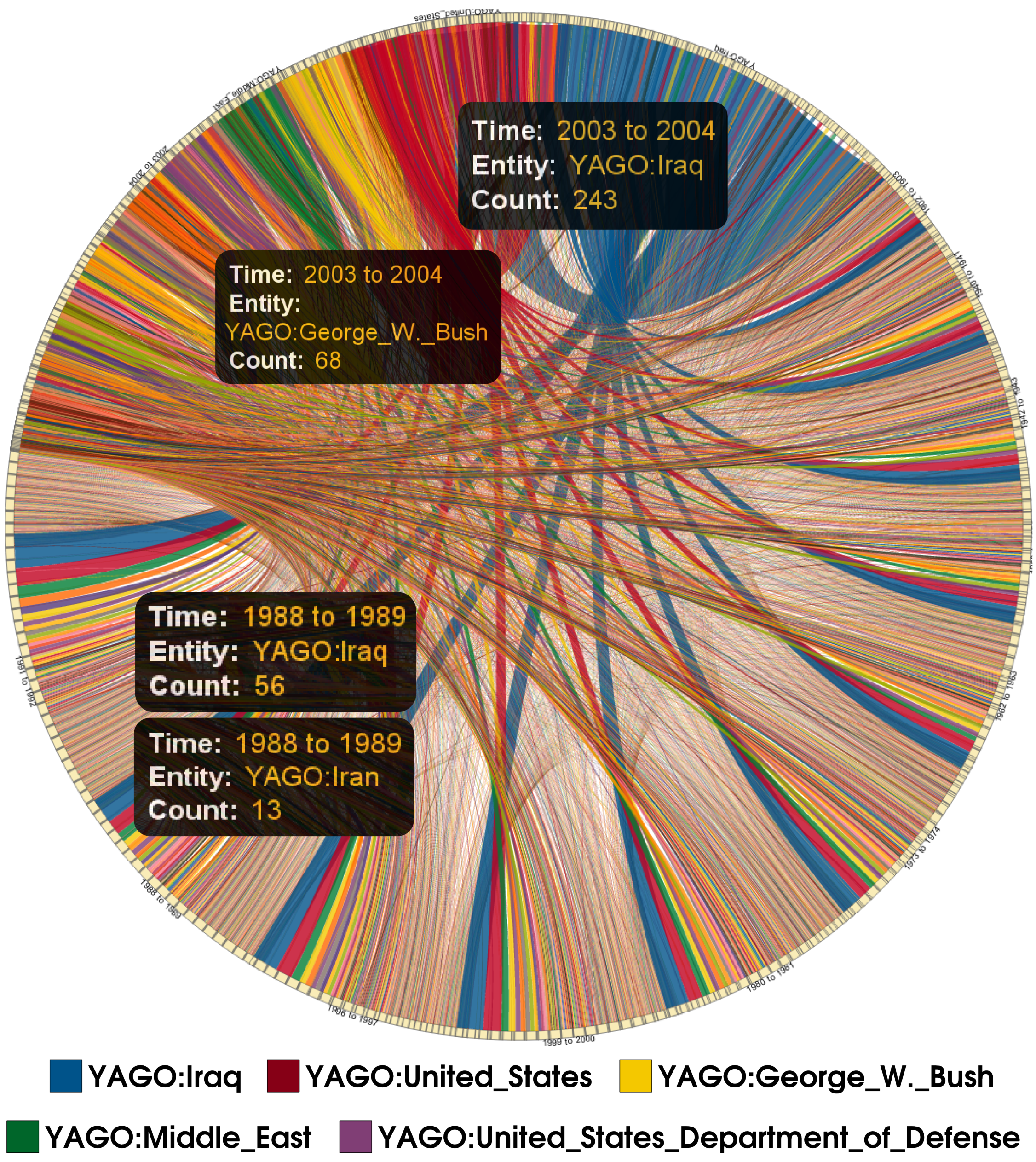


Figure 13.2: Chord diagram for the query *iraq war*. In the chord diagram, we can see the key entities by large chords corresponding to IRAQ, GEORGE W. BUSH, and IRAN. Iraq was involved in multiple conflicts which can be seen in the highlighted boxes, i.e., the American invasion of Iraq in 2003 and the Iran-Iraq war of 1980-1988. The most frequent entities are highlighted in the legend.

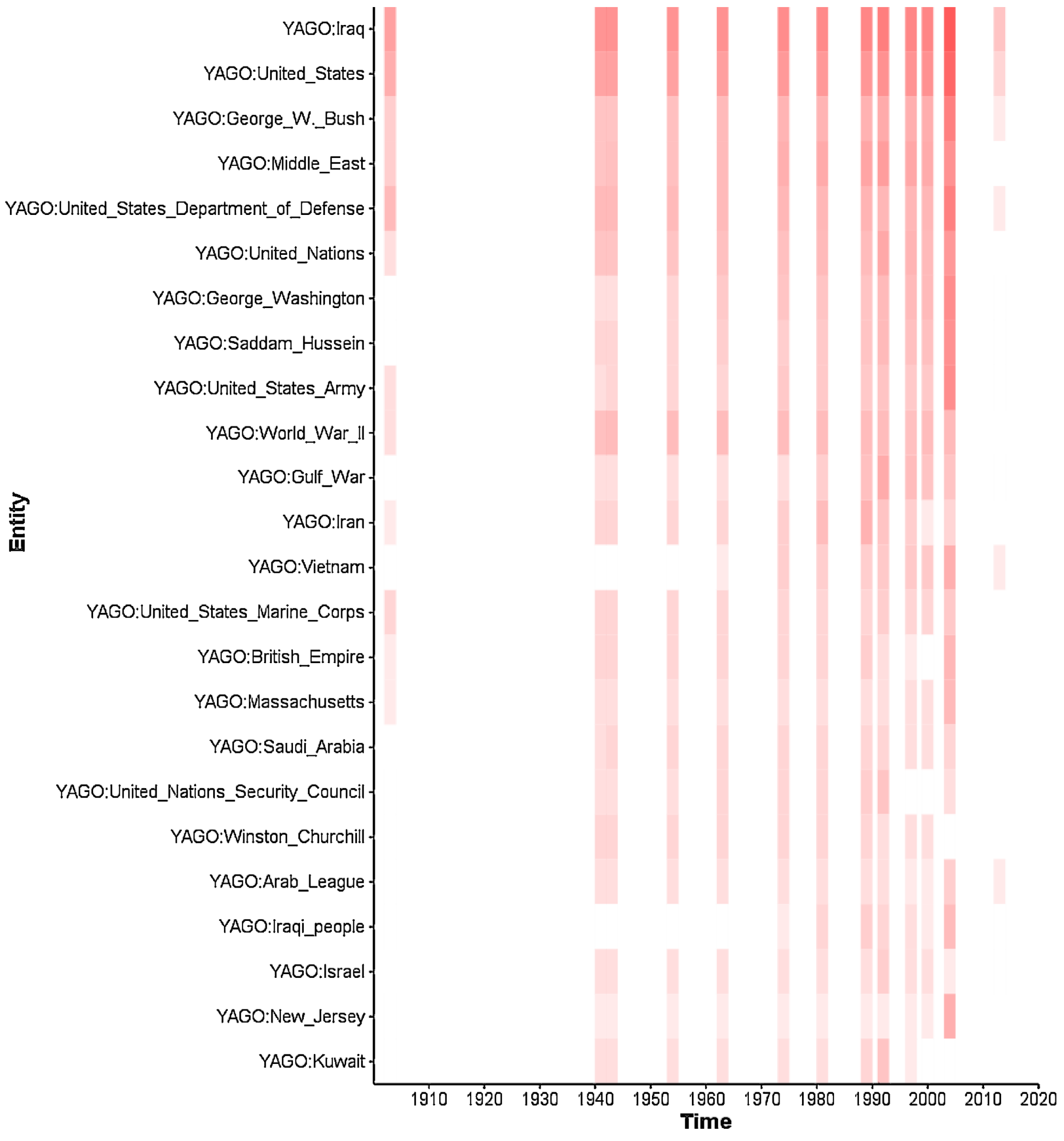


Figure 13.3: Heatmap diagram for the query *iraq war*. In the heatmap diagram, each cell shows the frequency of the named entity in the corresponding time point.

CHAPTER 14

CONCLUSION

Current information retrieval systems are ill-equipped to handle user's information needs expressed via short keyword queries. In order to serve the user's information need better, we turn to semantic annotations that natural language processing tools can now provide with great accuracy and reliability. Having large document collections tagged with semantic annotations in the form of part-of-speech tags, named entities, temporal expressions, and numerical values presents us with many unique opportunities for search and analytics. Concretely, to help in the search and analysis of large semantically annotated document collections, we described contributions along the following three research directions: indexing, querying, and mining of annotated document collections. We next summarize the contributions this thesis has made to the research community.

Indexing Annotated Document Collections

Knowledge-centric tasks such as information extraction, relationship extraction, and question answering rely on retrieval of text regions expressed via GREP-like patterns containing regular expressions, annotations, and word sequences. However, current information retrieval systems do not support GREP-like operators (e.g., regular expressions) to be expressed between word sequences and annotations. To assist, journalists and scholars in humanities for whom knowledge-centric tasks are of importance, we describe GYANI, an indexing infrastructure for annotated document collections. Our experiments show that GYANI returns text regions for knowledge-centric tasks with significant speedups: 95× for information extraction, 53× for question answering, and 12× for relationship extraction. GYANI thus truly enables knowledge acquisition at scale.

Knowledge graphs are rich information repositories as they encode relationships between real-world entities. However, due to limitations at the data modeling level, knowledge graphs can only encode facts in the form of ⟨SUBJECT, PREDICATE, OBJECT⟩ triples. In cases where the information does not exist in the knowledge graph, its is required that we can put such facts into context by spotting them in text documents. Natural language representations of knowledge graph facts can be modeled as a sequence phrase sets – which we refer to as hyper-phrase queries. To speed up the retrieval of text regions in support of such hyper-phrase queries, we showed how the combinatorial space of the phrases can be modeled using n-grams and skip-grams. Furthermore, to minimize the time needed to merge the posting lists corresponding to the phrases, we proposed a join-order optimization using dynamic programming. Our evaluation on document collections amounting to more than thirty million documents show that we can retrieve evidences for knowledge graph facts and subgraphs within seconds.

Querying Annotated Document Collections

Users' often express their information needs via short keyword queries. To narrow down their search, users reformulate their queries by adding words from the documents obtained from the previous query. This process can often be too cumbersome for the user. To simplify this process, we described methods that leverage semantic annotations to help the user navigate to their information need easily. First, we focused on temporal information needs, where the user's query has a temporal dimension to it. For instance, for the query *summer olympics*, the user will be glad to know the time intervals in which the Olympics took place and thereby narrow down her search to a particular time interval. To do so, we proposed methods to identify time intervals of interest to keyword queries. This was done by a computing the salience of temporal expressions, obtained from the pseudo-relevant set of documents, in an uncertainty-aware time model. Our experiments show that using the uncertainty-aware model for identifying time intervals of interest is more effective than a naïve method of simply counting frequent temporal expressions.

Second, we leverage the time intervals of interest for classification of temporal queries at different levels of granularities. For instance, for a query such as *summer olympics*, it is helpful to know that it is periodically recurring event at the year granularity, so that information concerning the next Olympics can be retrieved. To determine the temporal class of a query, we use Bayesian analysis of the generated time intervals of interest for multi-modality. Using our proposed features derived from this analysis of multi-modality, we show that we can indeed determine the temporal class of the query in a new taxonomy with greater precision and recall as compared to existing methods.

Third, we leverage the time intervals of interest for temporal diversification of search results. Concretely, we address the information needs for history-oriented queries that require retrieved results cover the timeline of events or sketch biography for an entity. To this end, the diversification objective is to identify a subset of documents from the pseudo-relevant set such that there exists at least one document that covers one of generated time intervals of interest for the history-oriented query. Our experiments show that using our temporal diversification objective is more useful for constructing historical-summaries as opposed to text-only based methods that reward textual novelty.

Fourth, we proposed a method on helping the user navigate large annotated document collections via the novel concept of semantic aspects. The semantic aspects generated in response to short and ambiguous keyword queries reflect salient annotations that co-occur frequently in the pseudo-relevant set of documents. The semantic aspects thus uplift the unstructured text into a structured representation, allowing the user to navigate them without the need to read their contents. Our experiments on news and web archives, amounting to more than 450 million documents, show that our approach leveraging semantic annotations can generate quality aspects as compared to simple text clustering techniques.

Mining Annotated Document Collections

Interesting insights can be mined by leveraging the saliency of annotations present in large document collections. We proposed four methods of leveraging annotations present in large document collections for mining insights. First, we proposed a method that identifies time intervals of interest for knowledge graph facts. To do so, our method leverages the GYANI indexing infrastructure for retrieving text regions corresponding to knowledge graph facts with temporal predicates. Our approach then computes a ranked list of temporal scopes for the input knowledge graph fact. Our experimental findings, show that using the uncertainty-aware model for determining the time intervals of interest for knowledge graph facts leads to higher precision as opposed to simply counting frequent temporal expressions.

Second, we proposed the EventMiner algorithm for mining events from a set of annotated documents. Concretely, given a keyword query, EventMiner retrieves a pseudo-relevant set of documents and clusters the annotations in event clusters. The clustering is done by leveraging similarity functions that embody semantic models for time, geography, and entities. Our evaluation shows that by considering the annotation semantics during clustering we can perform better than simply counting annotation frequency for generating event clusters.

Third, we presented jigsaw, an end-to-end query-driven system that pieces together unstructured text from large semantically annotated document collections into structured tables. To piece together text into tables, jigsaw relies on efficient query processing over the gyani indexing infrastructure that allows scalable extraction of annotated text regions in support of a structured query. To generate tables and describe their schema jigsaw provides three operator classes: `QUERY`, `LINK`, and `ANALYZE`. The `QUERY` operators shape text into rows. The `LINK` operators reconcile near-duplicate mentions of entities, predicates, temporal expressions, and numerical values in the table by considering their semantics. When a value doesn't exist (NULL), `LINK` estimates it, using the context or provenance of the extracted row. The `ANALYZE` operator allows the user to aggregate and rank the linked rows in the table for information consumption. We showed that we can generate high-quality tables from annotated collections amounting to more than twenty five million documents within seconds.

Fourth, we presented a complete time-sensitive search system, DigitalHistorian. Using DigitalHistorian, the user can enter keyword queries and obtain a ranked list of time intervals of interest to help her navigate the document collection. In addition to this, the documents presented are diversified along time, giving the user a historical overview of the query issued. DigitalHistorian also provides analytical visualizations that present a glimpse of the connections between the identified list of time intervals of interest and the disambiguated named entities present in the pseudo-relevant set of documents for the given history-oriented query.

14.1 Outlook

Limitations. We now discuss three shortcomings of the works proposed in this thesis. First, gyani efficiently supports scalable retrieval of text regions in support of structured queries consisting of word sequences, annotations, and regular expressions. However, our infrastructure does not take into account automatic synonym discovery for entities and predicates. To address this shortcoming an additional set of indexes and dictionaries are needed to be maintained that can automatically populate a structured query pattern with additional aliases to increase recall of retrieved text regions. This approach of search over structured resources (e.g., relational databases) has been proposed before [43, 111]. By additionally using this augmentation from prior work [43, 111], we can automate the query template generation.

Second, our probabilistic framework for time-sensitive search currently addresses the uncertainty behind temporal expressions, e.g., `1930s`, by associating an uniform likelihood to all possible time intervals. Contextual modifiers around temporal expressions can further help disambiguate the uncertainty associated with them. For instance, for the temporal expression `late 1930s`, we do not need to consider every possible time interval modeled by the four-tuple $\langle 1930{,}1939{,}1930{,}1939 \rangle$. Since, the context `late` conveys that there is higher likelihood of the time interval to lie in $[1935, 1939]$, we can further associate a heavy-tail distribution for this expression of time rather than considering a uniform likelihood for all intervals in $[1930, 1939]$. Leveraging the contextual semantics for modeling temporal expressions has been looked at before [124]. However, in [124] this was done in a limited scope without modeling the uncertainty behind temporal expressions. By combining both these approaches [41, 124] of modeling time, we expect a further increase in precision of our proposed approaches.

Third and finally, our event mining algorithm can only identify event clusters in isolation for a keyword query about an entity or event. Currently, our approach lacks the ability to weave together the event clusters into a story, as envisaged by the original *Topic Detection and Tracking* study. Event threading has been investigated before [79], however there no attention was paid to modeling of semantic annotations for event clustering. With such an ability built into our EventMiner algorithm, we can further see how the events evolved over time in the news archives for a given keyword query.

Future Work. The thesis presented ways to shape unstructured text into events, aspects, tables, and analytical visualizations for easy navigation of search results. Despite this, it is never clear as to what the user's information need is via simple keyword querying (too broad) or via more structured querying approaches (too narrow). Information needs are varied and a single answer to query is almost never enough. To this end, further research is needed in the direction of user interfaces and result presentation that builds on the work presented in this thesis.

REFERENCES

1. The ClueWeb09 Dataset.
   Last Accessed: July 1, 2019.
   URL: `http://lemurproject.org/clueweb09/`.
2. The ClueWeb12 Dataset.
   Last Accessed: July 1, 2019.
   URL: `http://lemurproject.org/clueweb12/`.
3. Crunchbase.
   Last Accessed: July 1, 2019.
   URL: `https://www.crunchbase.com/`.
4. Elastic — Revealing Insights from Data (Formerly Elasticsearch).
   Last Accessed: July 1, 2019.
   URL: `https://www.elastic.co/`.
5. English Gigaword Fifth Edition.
   Last Accessed: July 1, 2019.
   URL: `https://catalog.ldc.upenn.edu/LDC2011T07`.
6. Fortune.
   Last Accessed: July 1, 2019.
   URL: `http://fortune.com/fortune500/`.
7. The GDELT Project.
   Last Accessed: July 1, 2019.
   URL: `https://www.gdeltproject.org/`.
8. GNU Grep 3.0.
   Last Accessed: July 1, 2019.
   URL: `https://www.gnu.org/software/grep/manual/grep.html`.
9. Google Tables.
   Last Accessed: July 1, 2019.
   URL: `https://research.google.com/tables`.

10. Introducing Structured Snippets, Now a Part of Google Web Search.
Last Accessed: July 1, 2019.
URL: https://ai.googleblog.com/2014/09/introducing-structured-snippets-now.html.

11. JavaFastPFOR: A Simple Integer Compression Library in Java.
Last Accessed: July 1, 2019.
URL: https://github.com/lemire/JavaFastPFOR.

12. List of Largest Manufacturing Companies by Revenue - Wikipedia.
Last Accessed: July 1, 2019.
URL: https://en.wikipedia.org/wiki/List_of_largest_manufacturing_companies_by_revenue.

13. List of Lists of Lists.
Last Accessed: July 1, 2019.
URL: https://en.wikipedia.org/wiki/List_of_lists_of_lists.

14. List of the Largest Software Companies - Wikipedia.
Last Accessed: July 1, 2019.
URL: https://en.wikipedia.org/wiki/List_of_the_largest_software_companies.

15. Living Knowledge Corpus.
Last Accessed: July 1, 2019.
URL: https://web.archive.org/web/20171226132758/http://livingknowledge.europarchive.org/index.php.

16. George W. Bush — Wikipedia, The Free Encyclopedia, 2015.
Last Accessed: July 1, 2019.
URL: https://en.wikipedia.org/wiki/George_W._Bush.

17. Maria Sharapova.
Last Accessed: July 1, 2019.
URL: https://en.wikipedia.org/wiki/Maria_Sharapova.

18. The New York Times Annotated Corpus.
Last Accessed: July 1, 2019.
URL: https://catalog.ldc.upenn.edu/LDC2008T19.

19. The New York Times: On This Day.
Last Accessed: July 1, 2019.
URL: https://learning.blogs.nytimes.com/on-this-day/.

20. Project Gutenburg.
Last Accessed: July 1, 2019.
URL: https://www.gutenberg.org.

21. Wikidata: Sitelinks.
Last Accessed: July 1, 2019.
URL: https://www.wikidata.org/wiki/Help:Sitelinks.

22. Wikidata - The Free Knowledge Base.
Last Accessed: July 1, 2019.
URL: https://www.wikidata.org/.

23. Wikipedia: The Free Encyclopedia.
Last Accessed: July 1, 2019.
URL: https://www.wikipedia.org/.

24. Abujabal, A., and Berberich, K. Important Events in the Past, Present, and Future. In *Proceedings of the 24th International Conference on World Wide Web Companion, WWW 2015, Florence, Italy, May 18-22, 2015 - Companion Volume* (2015), pp. 1315–1320. URL: http://doi.acm.org/10.1145/2740908.2741692.

25. Agrawal, R., Gollapudi, S., Halverson, A., and Ieong, S. Diversifying Search Results. In *Proceedings of the Second International Conference on Web Search and Web Data Mining, WSDM 2009, Barcelona, Spain, February 9-11, 2009* (2009), pp. 5–14. URL: `http://doi.acm.org/10.1145/1498759.1498766`.

26. Agrawal, R., and Srikant, R. Fast Algorithms for Mining Association Rules in Large Databases. In *VLDB'94, Proceedings of 20th International Conference on Very Large Data Bases, September 12-15, 1994, Santiago de Chile, Chile* (1994), pp. 487–499.

27. Agrawal, S., Chakrabarti, K., Chaudhuri, S., and Ganti, V. Scalable Ad-Hoc Entity Extraction from Text Collections. *PVLDB 1*, 1 (2008), 945–957. URL: `http://www.vldb.org/pvldb/1/1453958.pdf`.

28. Allan, J., Ed. *Topic Detection and Tracking: Event-Based Information Organization.* Kluwer Academic Publishers, Norwell, MA, USA, 2002.

29. Alonso, O., Gertz, M., and Baeza-Yates, R. A. On the Value of Temporal Information in Information Retrieval. *SIGIR Forum 41*, 2 (2007), 35–41.

30. Alonso, O., Gertz, M., and Baeza-Yates, R. A. Clustering and Exploring Search Results using Timeline Constructions. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management, CIKM 2009, Hong Kong, China, November 2-6, 2009* (2009), pp. 97–106. URL: `http://doi.acm.org/10.1145/1645953.1645968`.

31. Antoniak, C. E. Mixtures of Dirichlet Processes with Applications to Bayesian Nonparametric Problems. *Ann. Statist. 2*, 6 (11 1974), 1152–1174. URL: `http://dx.doi.org/10.1214/aos/1176342871`.

32. Bader, J., Gog, S., and Petri, M. Practical Variable Length Gap Pattern Matching. In *Experimental Algorithms - 15th International Symposium, SEA 2016, St. Petersburg, Russia, June 5-8, 2016, Proceedings* (2016), pp. 1–16. URL: `https://doi.org/10.1007/978-3-319-38851-9_1`.

33. Baeza-Yates, R. Searching the future. In *SIGIR Workshop MF/IR* (2005).

34. Balog, K. *Entity-Oriented Search*, vol. 39 of *The Information Retrieval Series.* Springer, 2018. URL: `https://doi.org/10.1007/978-3-319-93935-3`.

35. Banko, M., Cafarella, M. J., Soderland, S., Broadhead, M., and Etzioni, O. Open Information Extraction from the Web. In *IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence, Hyderabad, India, January 6-12, 2007* (2007), pp. 2670–2676. URL: `http://ijcai.org/Proceedings/07/Papers/429.pdf`.

36. Bast, H., and Buchhold, B. An Index for Efficient Semantic Full-Text Search. In *22nd ACM International Conference on Information and Knowledge Management, CIKM'13, San Francisco, CA, USA, October 27 - November 1, 2013* (2013), pp. 369–378. URL: `http://doi.acm.org/10.1145/2505515.2505689`.

37. Bast, H., Buchhold, B., and Haussmann, E. Semantic Search on Text and Knowledge Bases. *Foundations and Trends in Information Retrieval 10*, 2-3 (2016), 119–271. URL: `http://dx.doi.org/10.1561/1500000032`.

38. Batini, C., Cappiello, C., Francalanci, C., and Maurino, A. Methodologies for Data Quality Assessment and Improvement. *ACM Comput. Surv. 41*, 3 (2009), 16:1–16:52. URL: `https://doi.org/10.1145/1541880.1541883`.

39. Ben-Yitzhak, O., Golbandi, N., Har'El, N., Lempel, R., Neumann, A., Ofek-Koifman, S., Sheinwald, D., Shekita, E. J., Sznajder, B., and Yogev, S. Beyond Basic Faceted Search. In *Proceedings of the International Conference on Web Search and Web Data Mining, WSDM 2008, Palo Alto, California, USA, February 11-12, 2008* (2008), pp. 33–44. URL: `http://doi.acm.org/10.1145/1341531.1341539`.

40. Berberich, K., and Bedathur, S. Temporal Diversification of Search Results. In *Proceedings of Workshop on Time-aware Information Access (TAIA 2013)* (2013).

41. Berberich, K., Bedathur, S. J., Alonso, O., and Weikum, G. A Language Modeling Approach for Temporal Information Needs. In *Advances in Information Retrieval, 32nd European Conference on IR Research, ECIR 2010, Milton Keynes, UK, March 28-31, 2010. Proceedings* (2010), pp. 13–25. URL: `http://dx.doi.org/10.1007/978-3-642-12275-0_5`.

42. Bhagavatula, C. S., Noraset, T., and Downey, D. *TabEL: Entity Linking in Web Tables.* Springer International Publishing, Cham, 2015, pp. 425–441.

43. Bhalotia, G., Hulgeri, A., Nakhe, C., Chakrabarti, S., and Sudarshan, S. Keyword Searching and Browsing in Databases using BANKS. In *Proceedings of the 18th International Conference on Data Engineering, San Jose, CA, USA, February 26 - March 1, 2002* (2002), pp. 431–440. URL: `https://doi.org/10.1109/ICDE.2002.994756`.

44. Bianchi, F., Palmonari, M., and Nozza, D. Towards Encoding Time in Text-Based Entity Embeddings. In *The Semantic Web - ISWC 2018 - 17th International Semantic Web Conference, Monterey, CA, USA, October 8-12, 2018, Proceedings, Part I* (2018), pp. 56–71. URL: `https://doi.org/10.1007/978-3-030-00671-6_4`.

45. Bille, P., Gørtz, I. L., Vildhøj, H. W., and Wind, D. K. String Matching with Variable Length Gaps. *Theor. Comput. Sci. 443* (2012), 25–34. URL: `https://doi.org/10.1016/j.tcs.2012.03.029`.

46. Blackwell, D., and MacQueen, J. B. Ferguson Distributions Via Polya Urn Schemes. *Ann. Statist. 1*, 2 (03 1973), 353–355. URL: `http://dx.doi.org/10.1214/aos/1176342372`.

47. Blei, D. M., Ng, A. Y., and Jordan, M. I. Latent Dirichlet Allocation. *J. Mach. Learn. Res. 3* (Mar. 2003), 993–1022. URL: `http://dl.acm.org/citation.cfm?id=944919.944937`.

48. Bordino, I., Lalmas, M., Mejova, Y., and Laere, O. V. Beyond Entities: Promoting Explorative Search with Bundles. *Inf. Retr. Journal 19*, 5 (2016), 447–486. URL: `http://dx.doi.org/10.1007/s10791-016-9283-5`.

49. Breiman, L., Friedman, J. H., Olshen, R. A., and Stone, C. J. *Classification and Regression Trees.* Wadsworth, 1984.

50. Büttcher, S., Clarke, C. L. A., and Cormack, G. V. *Information Retrieval - Implementing and Evaluating Search Engines.* MIT Press, 2010. URL: `http://mitpress.mit.edu/books/information-retrieval`.

51. Cafarella, M. J., and Etzioni, O. A Search Engine for Natural Language Applications. In *Proceedings of the 14th International Conference on World Wide Web, WWW 2005, Chiba, Japan, May 10-14, 2005* (2005), pp. 442–452. URL: `http://doi.acm.org/10.1145/1060745.1060811`.

52. Cafarella, M. J., Halevy, A. Y., Lee, H., Madhavan, J., Yu, C., Wang, D. Z., and Wu, E. Ten Years of WebTables. *PVLDB 11*, 12 (2018), 2140–2149. URL: `http://www.vldb.org/pvldb/vol11/p2140-cafarella.pdf`.

53. Campos, R., Dias, G., Jorge, A. M., and Jatowt, A. Survey of Temporal Information Retrieval and Related Applications. *ACM Comput. Surv. 47*, 2 (2014), 15:1–15:41. URL: `http://doi.acm.org/10.1145/2619088`.

54. Cannaviccio, M., Barbosa, D., and Merialdo, P. Towards Annotating Relational Data on the Web with Language Models. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web, WWW 2018, Lyon, France, April 23-27, 2018* (2018), pp. 1307–1316. URL: `http://doi.acm.org/10.1145/3178876.3186029`.

55. Carbonell, J., and Goldstein, J. The Use of MMR, Diversity-Based Reranking for Reordering Documents and Producing Summaries. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in*

Information Retrieval (New York, NY, USA, 1998), SIGIR '98, ACM, pp. 335–336. URL: http://doi.acm.org/10.1145/290941.291025.

56. Catena, M., Macdonald, C., and Ounis, I. On Inverted Index Compression for Search Engine Efficiency. In *Advances in Information Retrieval - 36th European Conference on IR Research, ECIR 2014, Amsterdam, The Netherlands, April 13-16, 2014. Proceedings* (2014), pp. 359–371. URL: https://doi.org/10.1007/978-3-319-06028-6_30.

57. Ceccarelli, D., Lucchese, C., Orlando, S., Perego, R., and Trani, S. Learning Relatedness Measures for Entity Linking. In *22nd ACM International Conference on Information and Knowledge Management, CIKM'13, San Francisco, CA, USA, October 27 - November 1, 2013* (2013), pp. 139–148. URL: https://doi.org/10.1145/2505515.2505711.

58. Chang, A. X., and Manning, C. D. SUTime: A Library for Recognizing and Normalizing Time Expressions. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC-2012), Istanbul, Turkey, May 23-25, 2012* (2012), pp. 3735–3740. URL: http://www.lrec-conf.org/proceedings/lrec2012/summaries/284.html.

59. Chang, F., Dean, J., Ghemawat, S., Hsieh, W. C., Wallach, D. A., Burrows, M., Chandra, T., Fikes, A., and Gruber, R. E. Bigtable: A Distributed Storage System for Structured Data. In *7th USENIX Symposium on Operating Systems Design and Implementation (OSDI)* (2006), pp. 205–218.

60. Cho, J., and Rajagopalan, S. A Fast Regular Expression Indexing Engine. In *Proceedings of the 18th International Conference on Data Engineering, San Jose, CA, USA, February 26 - March 1, 2002* (2002), pp. 419–430. URL: http://dx.doi.org/10.1109/ICDE.2002.994755.

61. Clarke, C. L. A., Cormack, G. V., and Burkowski, F. J. An Algebra for Structured Text Search and a Framework for its Implementation. *Comput. J. 38*, 1 (1995), 43–56. URL: http://dx.doi.org/10.1093/comjnl/38.1.43.

62. Clarke, C. L. A., Kolla, M., Cormack, G. V., Vechtomova, O., Ashkan, A., Büttcher, S., and MacKinnon, I. Novelty and Diversity in Information Retrieval Evaluation. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2008, Singapore, July 20-24, 2008* (2008), pp. 659–666. URL: http://doi.acm.org/10.1145/1390334.1390446.

63. Cohen, S., Li, C., Yang, J., and Yu, C. Computational Journalism: A Call to Arms to Database Researchers. In *CIDR 2011, Fifth Biennial Conference on Innovative Data Systems Research, Asilomar, CA, USA, January 9-12, 2011, Online Proceedings* (2011), pp. 148–151. URL: http://cidrdb.org/cidr2011/Papers/CIDR11_Paper17.pdf.

64. Consens, M. P., and Milo, T. Algebras for Querying Text Regions - Expressive Power and Optimization. *J. Comput. Syst. Sci. 57*, 3 (1998), 272–288. URL: http://linkinghub.elsevier.com/retrieve/pii/S0022000098915641.

65. Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. *Introduction to Algorithms, Second Edition.* The MIT Press and McGraw-Hill Book Company, 2001.

66. Cornacchia, R., Héman, S., Zukowski, M., de Vries, A. P., and Boncz, P. A. Flexible and Efficient IR Using Array Databases. *VLDB J. 17*, 1 (2008), 151–168. URL: http://dx.doi.org/10.1007/s00778-007-0071-0.

67. Crochemore, M., Iliopoulos, C. S., Makris, C., Rytter, W., Tsakalidis, A. K., and Tsichlas, T. Approximate String Matching with Gaps. *Nord. J. Comput. 9*, 1 (2002), 54–65.

68. Crow, D. Google Squared: Web Scale, Open Domain Information Extraction and Presentation. In *European Conference on Information Retrieval, Industry Day* (2010).

69. Culpepper, J. S., and Moffat, A. Efficient Set Intersection for Inverted Indexing. *ACM Trans. Inf. Syst. 29*, 1 (2010), 1:1–1:25. URL: `http://doi.acm.org/10.1145/1877766.1877767`.

70. Dakka, W., Gravano, L., and Ipeirotis, P. G. Answering General Time-Sensitive Queries. *IEEE Trans. Knowl. Data Eng. 24*, 2 (2012), 220–235. URL: `http://doi.ieeecomputersociety.org/10.1109/TKDE.2010.187`.

71. Davey, B. A., and Priestley, H. A. *Introduction to Lattices and Order*. Cambridge University Press, Cambridge, 1990. URL: `http://www.worldcat.org/search?qt=worldcat_org_all&q=0521367662`.

72. de Jong, F. M. G., Rode, H., and Hiemstra, D. Temporal Language Models for the Disclosure of Historical Text. In *Humanities, Computers and Cultural Heritage: Proceedings of the XVIth International Conference of the Association for History and Computing (AHC 2005), Amsterdam, The Netherlands* (Amsterdam, The Netherlands, September 2005), Royal Netherlands Academy of Arts and Sciences, pp. 161–168.

73. Dong, X., Gabrilovich, E., Heitz, G., Horn, W., Lao, N., Murphy, K., Strohmann, T., Sun, S., and Zhang, W. Knowledge vault: A Web-Scale Approach to Probabilistic Knowledge Fusion. In *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, New York, NY, USA - August 24 - 27, 2014* (2014), pp. 601–610. URL: `http://doi.acm.org/10.1145/2623330.2623623`.

74. Dou, Z., Hu, S., Luo, Y., Song, R., and Wen, J. Finding Dimensions for Queries. In *Proceedings of the 20th ACM Conference on Information and Knowledge Management, CIKM 2011, Glasgow, United Kingdom, October 24-28, 2011* (2011), pp. 1311–1320. URL: `http://doi.acm.org/10.1145/2063576.2063767`.

75. Elbassuoni, S., Hose, K., Metzger, S., and Schenkel, R. ROXXI: Reviving Witness Documents To Explore Extracted Information. *PVLDB 3*, 2 (2010), 1589–1592. URL: `http://www.comp.nus.edu.sg/~vldb2010/proceedings/files/papers/D19.pdf`.

76. Elmagarmid, A. K., Ipeirotis, P. G., and Verykios, V. S. Duplicate Record Detection: A Survey. *IEEE Trans. Knowl. Data Eng. 19*, 1 (2007), 1–16. URL: `https://doi.org/10.1109/TKDE.2007.250581`.

77. Evgeniy Gabrilovich, M. R., and Subramanya, A. FACC1: Freebase Annotation of ClueWeb Corpora, Version 1 (Release date 2013-06-26, Format version 1, Correction level 0), June 2013. `http://lemurproject.org/clueweb12/`.

78. Fellegi, I. P., and Holt, D. A Systematic Approach to Automatic Edit and Imputation. *Journal of the American Statistical Association 71*, 353 (1976), 17–35. URL: `http://www.jstor.org/stable/2285726`.

79. Feng, A., and Allan, J. Incident Threading for News Passages. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management, CIKM 2009, Hong Kong, China, November 2-6, 2009* (2009), pp. 1307–1316. URL: `https://doi.org/10.1145/1645953.1646118`.

80. Ferrucci, D., and Lally, A. UIMA: An Architectural Approach to Unstructured Information Processing in the Corporate Research Environment. *Nat. Lang. Eng. 10*, 3-4 (Sept. 2004), 327–348. URL: `http://dx.doi.org/10.1017/S1351324904003523`.

81. Fionda, V., and Pirrò, G. Fact Checking via Evidence Patterns. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden.* (2018), pp. 3755–3761. URL: `https://doi.org/10.24963/ijcai.2018/522`.

82. Fredriksson, K., and Grabowski, S. Efficient Algorithms for Pattern Matching with General Gaps, Character Classes, and Transposition Invariance. *Inf. Retr. 11*, 4 (2008), 335–357. URL: `https://doi.org/10.1007/s10791-008-9054-z`.

83. Frick, E., Schnober, C., and Banski, P. Evaluating Query Languages for a Corpus Processing System. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation, LREC 2012, Istanbul, Turkey, May 23-25, 2012* (2012), pp. 2286–2294. URL: `http://www.lrec-conf.org/proceedings/lrec2012/summaries/800.html`.

84. Gad-Elrab, M. H., Stepanova, D., Urbani, J., and Weikum, G. ExFaKT: A Framework for Explaining Facts over Knowledge Graphs and Text. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining, WSDM 2019, Melbourne, VIC, Australia, February 11-15, 2019* (2019), pp. 87–95. URL: `https://doi.org/10.1145/3289600.3290996`.

85. Gerber, D., Esteves, D., Lehmann, J., Bühmann, L., Usbeck, R., Ngomo, A. N., and Speck, R. DeFacto - Temporal and multilingual Deep Fact Validation. *J. Web Semant. 35* (2015), 85–101. URL: `https://doi.org/10.1016/j.websem.2015.08.001`.

86. Gey, F., Larson, R., Kando, N., Machado, J., and Sakai, T. NTCIR-GeoTime Overview: Evaluating Geographic and Temporal Search. In *Proc. NTCIR-8 Workshop Meeting* (2010), pp. 147–153.

87. Goldstein, J., Ramakrishnan, R., and Shaft, U. Compressing Relations and Indexes. In *Proceedings of the Fourteenth International Conference on Data Engineering, Orlando, Florida, USA, February 23-27, 1998* (1998), pp. 370–379. URL: `https://doi.org/10.1109/ICDE.1998.655800`.

88. Grau, B. C., Kharlamov, E., Marciuska, S., Zheleznyakov, D., and Arenas, M. SemFacet: Faceted Search over Ontology Enhanced Knowledge Graphs. In *Proceedings of the ISWC 2016 Posters & Demonstrations Track co-located with 15th International Semantic Web Conference (ISWC 2016), Kobe, Japan, October 19, 2016.* (2016). URL: `http://ceur-ws.org/Vol-1690/paper75.pdf`.

89. Gray, J., Bounegru, L., and Chambers, L. *The Data Journalism Handbook.* 08 2012.

90. Griffiths, T. L., and Steyvers, M. Finding Scientific Topics. *Proceedings of the National Academy of Sciences 101*, Suppl. 1 (April 2004), 5228–5235.

91. Guo, J., Xu, G., Cheng, X., and Li, H. Named Entity Recognition in Query. In *Proceedings of the 32nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2009, Boston, MA, USA, July 19-23, 2009* (2009), pp. 267–274.

92. Gupta, D. Event Search and Analytics: Detecting Events in Semantically Annotated Corpora for Search & Analytics. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining, San Francisco, CA, USA, February 22-25, 2016* (2016), p. 705. URL: `https://doi.org/10.1145/2835776.2855083`.

93. Gupta, D., and Berberich, K. Identifying Time Intervals of Interest to Queries. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management, CIKM 2014, Shanghai, China, November 3-7, 2014* (2014), pp. 1835–1838. URL: `https://doi.org/10.1145/2661829.2661927`.

94. Gupta, D., and Berberich, K. Temporal Query Classification at Different Granularities. In *String Processing and Information Retrieval - 22nd International Symposium, SPIRE 2015, London, UK, September 1-4, 2015, Proceedings* (2015), pp. 156–164. URL: `https://doi.org/10.1007/978-3-319-23826-5_16`.

95. Gupta, D., and Berberich, K. Diversifying Search Results Using Time. Tech. rep., Max Planck Institute for Informatics, 2016.

96. Gupta, D., and Berberich, K. Diversifying Search Results Using Time - An Information Retrieval Method for Historians. In *Advances in Information Retrieval - 38th European Conference on IR Research, ECIR 2016, Padua, Italy, March 20-23, 2016. Proceedings* (2016), pp. 789–795. URL: `https://doi.org/10.1007/978-3-319-30671-1_69`.

97. Gupta, D., and Berberich, K. A Probabilistic Framework for Time-Sensitive Search: MPII at the NTCIR-12 Temporalia-2 Task. In *Proceedings of the 12th NTCIR Conference on Evaluation of Information Access Technologies, National Center of Sciences, Tokyo, Japan, June 7-10, 2016* (2016). URL: `http://research.nii.ac.jp/ntcir/workshop/OnlineProceedings12/pdf/ntcir/TEMPORALIA/02-NTCIR12-TEMPORALIA-GuptaD.pdf`.

98. Gupta, D., and Berberich, K. GYANI: An Indexing Infrastructure for Knowledge-Centric Tasks. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management, CIKM 2018, Torino, Italy, October 22-26, 2018* (2018), pp. 487–496. URL: `https://doi.org/10.1145/3269206.3271745`.

99. Gupta, D., and Berberich, K. Identifying Time Intervals for Knowledge Graph Facts. In *Companion of the The Web Conference 2018 on The Web Conference 2018, WWW 2018, Lyon , France, April 23-27, 2018* (2018), pp. 37–38. URL: `https://doi.org/10.1145/3184558.3186917`.

100. Gupta, D., and Berberich, K. Efficient Retrieval of Knowledge Graph Fact Evidences. In *The Semantic Web: ESWC 2019 Satellite Events - ESWC 2019 Satellite Events, Portorož, Slovenia, June 2-6, 2019, Revised Selected Papers* (2019), pp. 90–94. URL: `https://doi.org/10.1007/978-3-030-32327-1\_18`.

101. Gupta, D., and Berberich, K. JIGSAW: Structuring Text into Tables. In *Proceedings of the 2019 ACM on International Conference on the Theory of Information Retrieval, ICTIR 2019, Santa Clara, CA, USA, October 2-5, 2019* (2019), pp. 237–244. URL: `https://doi.org/10.1145/3341981.3344228`.

102. Gupta, D., and Berberich, K. Optimizing Hyper-Phrase Queries. *Under Submission*.

103. Gupta, D., and Berberich, K. Structured Search in Annotated Document Collections. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining, WSDM 2019, Melbourne, VIC, Australia, February 11-15, 2019* (2019), pp. 794–797. URL: `https://doi.org/10.1145/3289600.3290618`.

104. Gupta, D., Berberich, K., Strötgen, J., and Zeinalipour-Yazti, D. Generating Semantic Aspects for Queries. In *Proceedings of the 18th ACM/IEEE on Joint Conference on Digital Libraries, JCDL 2018, Fort Worth, TX, USA, June 03-07, 2018* (2018), pp. 335–336. URL: `https://doi.org/10.1145/3197026.3203900`.

105. Gupta, D., Berberich, K., Strötgen, J., and Zeinalipour-Yazti, D. Generating Semantic Aspects for Queries. In *The Semantic Web - 16th International Conference, ESWC 2019, Portorož, Slovenia, June 2-6, 2019, Proceedings* (2019), pp. 162–178. URL: `https://doi.org/10.1007/978-3-030-21348-0_11`.

106. Gupta, D., Strötgen, J., and Berberich, K. DIGITALHISTORIAN: Search & Analytics Using Annotations. In *Proceedings of the 3rd HistoInformatics Workshop on Computational History (HistoInformatics 2016) co-located with Digital Humanities 2016 conference (DH 2016), Krakow, Poland, July 11, 2016.* (2016), pp. 5–10. URL: `http://ceur-ws.org/Vol-1632/paper_1.pdf`.

107. Gupta, D., Strötgen, J., and Berberich, K. EventMiner: Mining Events from Annotated Documents. In *Proceedings of the 2016 ACM on International Conference on the Theory of Information Retrieval, ICTIR 2016, Newark, DE, USA, September 12-16, 2016* (2016), pp. 261–270. URL: `https://doi.org/10.1145/2970398.2970411`.

108. Guttman, A. R-Trees: A Dynamic Index Structure for Spatial Searching. In *SIGMOD'84, Proceedings of Annual Meeting, Boston, Massachusetts, June 18-21, 1984* (1984), pp. 47–57. URL: `http://doi.acm.org/10.1145/602259.602266`.

109. Han, J., Kamber, M., and Pei, J. *Data Mining: Concepts and Techniques*, 3rd ed. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2011.

110. Härdle, W., Müller, M., Sperlich, S., and Werwatz, A. *Nonparametric and Semiparametric Models.* Springer Series in Statistics. Springer-Verlag, New York, 2004. URL: `http://dx.doi.org/10.1007/978-3-642-17146-8`.

111. He, H., Wang, H., Yang, J., and Yu, P. S. BLINKS: Ranked Keyword Searches on Graphs. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, Beijing, China, June 12-14, 2007* (2007), pp. 305–316. URL: https://doi.org/10.1145/1247480.1247516.

112. Hearst, M. A. Automatic Acquisition of Hyponyms from Large Text Corpora. In *Proceedings of the 14th Conference on Computational Linguistics - Volume 2* (Stroudsburg, PA, USA, 1992), COLING '92, Association for Computational Linguistics, pp. 539–545. URL: http://dx.doi.org/10.3115/992133.992154.

113. Hearst, M. A. *Search User Interfaces*, 1st ed. Cambridge University Press, New York, NY, USA, 2009.

114. Hearst, M. A., and Plaunt, C. Subtopic Structuring for Full-Length Document Access. In *Proceedings of the 16th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval. Pittsburgh, PA, USA, June 27 - July 1, 1993* (1993), pp. 59–68. URL: http://doi.acm.org/10.1145/160688.160695.

115. Henry, J. Providing Knowledge Panels With Search Results, May 2 2013. US Patent App. 13/566,489. URL: https://www.google.com/patents/US20130110825.

116. Hoffart, J., Milchevski, D., and Weikum, G. AESTHETICS: Analytics with Strings, Things, and Cats. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management, CIKM 2014, Shanghai, China, November 3-7, 2014* (2014), pp. 2018–2020. URL: http://doi.acm.org/10.1145/2661829.2661835.

117. Hoffart, J., Milchevski, D., and Weikum, G. STICS: Searching with Strings, Things, and Cats. In *The 37th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '14, Gold Coast , QLD, Australia - July 06 - 11, 2014* (2014), pp. 1247–1248. URL: http://doi.acm.org/10.1145/2600428.2611177.

118. Hoffart, J., Yosef, M. A., Bordino, I., Fürstenau, H., Pinkal, M., Spaniol, M., Taneva, B., Thater, S., and Weikum, G. Robust Disambiguation of Named Entities in Text. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing* (Stroudsburg, PA, USA, 2011), EMNLP '11, Association for Computational Linguistics, pp. 782–792. URL: http://dl.acm.org/citation.cfm?id=2145432.2145521.

119. Hong, P. J., Gupta, P. K., Gaylinn, N. J., Kazhiyur-Mannar, R., Goel, K. J., Bar-or, O., Menzel, J. W., Dhanaraj, C. R., Levy, J. L., Thakur, S. A., et al. Related entities, Feb. 15 2018. US Patent App. 15/798,175.

120. Hu, L., Li, J., Li, X., Shao, C., and Wang, X. TSDPMM: Incorporating Prior Topic Knowledge into Dirichlet Process Mixture Models for Text Clustering. In *EMNLP* (2015), L. Màrquez, C. Callison-Burch, J. Su, D. Pighin, and Y. Marton, Eds., The Association for Computational Linguistics, pp. 787–792. URL: http://dblp.uni-trier.de/db/conf/emnlp/emnlp2015.html#HuLLSW15.

121. Huijnen, P., Laan, F., de Rijke, M., and Pieters, T. A Digital Humanities Approach to the History of Science - Eugenics Revisited in Hidden Debates by Means of Semantic Text Mining. In *Social Informatics - SocInfo 2013 International Workshops, QMC and HISTOINFORMATICS, Kyoto, Japan, November 25, 2013, Revised Selected Papers* (2013), pp. 71–85. URL: http://dx.doi.org/10.1007/978-3-642-55285-4_6.

122. Ipeirotis, P. G., Agichtein, E., Jain, P., and Gravano, L. To Search or to Crawl?: Towards a Query Optimizer for Text-Centric Tasks. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, Chicago, Illinois, USA, June 27-29, 2006* (2006), pp. 265–276. URL: http://doi.acm.org/10.1145/1142473.1142504.

123. Jatowt, A., Au Yeung, C.-M., and Tanaka, K. Estimating Document Focus Time. In *Proceedings of the 22nd ACM International Conference on Conference on*

Information and Knowledge Management (New York, NY, USA, 2013), CIKM '13, ACM, pp. 2273–2278. URL: `http://doi.acm.org/10.1145/2505515.2505655`.

124. Jatowt, A., and man Au Yeung, C. Extracting Collective Expectations About the Future from Large Text Collections. In *Proceedings of the 20th ACM Conference on Information and Knowledge Management, CIKM 2011, Glasgow, United Kingdom, October 24-28, 2011* (2011), pp. 1259–1264. URL: `http://doi.acm.org/10.1145/2063576.2063759`.

125. Joho, H., Jatowt, A., and Blanco, R. NTCIR Temporalia: A Test Collection for Temporal Information Access Research. In *23rd International World Wide Web Conference, WWW '14, Seoul, Republic of Korea, April 7-11, 2014, Companion Volume* (2014), pp. 845–850. URL: `http://doi.acm.org/10.1145/2567948.2579044`.

126. Joho, H., Jatowt, A., Blanco, R., Yu, H., and Yamamoto, S. Overview of NTCIR-12 Temporal Information Access (Temporalia-2) Task. In *Proceedings of the 12th NTCIR Conference on Evaluation of Information Access Technologies* (2016).

127. Jones, R., and Diaz, F. Temporal Profiles of Queries. *ACM Trans. Inf. Syst. 25*, 3 (July 2007). URL: `http://doi.acm.org/10.1145/1247715.1247720`.

128. Jurafsky, D., and Martin, J. H. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition, 2nd Edition.* Prentice Hall series in artificial intelligence. Prentice Hall, Pearson Education International, 2009. URL: `http://www.worldcat.org/oclc/315913020`.

129. Kando, N. Overview of the Eighth NTCIR Workshop. xi–xviii. URL: `http://research.nii.ac.jp/ntcir/workshop/OnlineProceedings8/NTCIR/01-NTCIR8-OV-KandoN.pdf`.

130. Kanhabua, N., Blanco, R., and Nørvåg, K. Temporal Information Retrieval. *Foundations and Trends in Information Retrieval 9*, 2 (2015), 91–208. URL: `http://dx.doi.org/10.1561/1500000043`.

131. Kanhabua, N., Nguyen, T. N., and Nejdl, W. Learning to Detect Event-Related Queries for Web Search. *Proceedings of the 5th Temporal Web Analytics Workshop (TempWeb'2015) at WWW'2015* (2015).

132. Kanhabua, N., and Nørvåg, K. Determining Time of Queries for Re-Ranking Search Results. In *Research and Advanced Technology for Digital Libraries*, M. Lalmas, J. Jose, A. Rauber, F. Sebastiani, and I. Frommholz, Eds., vol. 6273 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2010, pp. 261–272. URL: `http://dx.doi.org/10.1007/978-3-642-15464-5_27`.

133. Kanhabua, N., and Nørvåg, K. Using Temporal Language Models for Document Dating. In *ECML/PKDD (2)* (2009), pp. 738–741.

134. Kong, W., and Allan, J. Extracting Query Facets from Search Results. In *The 36th International ACM SIGIR conference on research and development in Information Retrieval, SIGIR '13, Dublin, Ireland - July 28 - August 01, 2013* (2013), pp. 93–102. URL: `http://doi.acm.org/10.1145/2484028.2484097`.

135. Koutrika, G., Liu, L., and Simske, S. J. Generating Reading Orders Over Document Collections. In *31st IEEE International Conference on Data Engineering, ICDE 2015, Seoul, South Korea, April 13-17, 2015* (2015), pp. 507–518. URL: `http://dx.doi.org/10.1109/ICDE.2015.7113310`.

136. Krause, T., Leser, U., and Lüdeling, A. graphANNIS: A Fast Query Engine for Deeply Annotated Linguistic Corpora. *Corpus Linguistic Software Tools 31*, 1 (2016), 1–25.

137. Kuzey, E., Setty, V., Strötgen, J., and Weikum, G. As Time Goes By: Comprehensive Tagging of Textual Phrases with Temporal Scopes. In *Proceedings of the 25th International Conference on World Wide Web, WWW 2016, Montreal,*

Canada, April 11 - 15, 2016 (2016), pp. 915–925. URL: http://doi.acm.org/10.1145/2872427.2883055.

138. Kuzey, E., Vreeken, J., and Weikum, G. A Fresh Look on Knowledge Bases: Distilling Named Events from News. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management, CIKM 2014, Shanghai, China, November 3-7, 2014* (2014), pp. 1689–1698. URL: http://doi.acm.org/10.1145/2661829.2661984.

139. Lalmas, M. *XML Retrieval.* Synthesis Lectures on Information Concepts, Retrieval, and Services. Morgan & Claypool Publishers, 2009. URL: https://doi.org/10.2200/S00203ED1V01Y200907ICR007.

140. Lardon, J., Abdellaoui, R., Bellet, F., Asfari, H., Souvignet, J., Texier, N., Jaulent, M.-C., Beyens, M.-N., Burgun, A., and Bousquet, C. Adverse drug reaction identification and extraction in social media: a scoping review. *Journal of medical Internet research 17*, 7 (2015), e171.

141. Lehmann, J., Gerber, D., Morsey, M., and Ngomo, A. N. DeFacto - Deep Fact Validation. In *The Semantic Web - ISWC 2012 - 11th International Semantic Web Conference, Boston, MA, USA, November 11-15, 2012, Proceedings, Part I* (2012), pp. 312–327. URL: https://doi.org/10.1007/978-3-642-35176-1_20.

142. Li, C., Yan, N., Roy, S. B., Lisham, L., and Das, G. Facetedpedia: Dynamic Generation of Query-dependent Faceted Interfaces for Wikipedia. In *Proceedings of the 19th International Conference on World Wide Web* (New York, NY, USA, 2010), WWW '10, ACM, pp. 651–660. URL: http://doi.acm.org/10.1145/1772690.1772757.

143. Li, H. Data Extraction from Text Using Wild Card Queries. *Masters Abstracts International* (2006). URL: http://webdocs.cs.ualberta.ca/~drafiei/papers/vldb06.pdf.

144. Li, X., and Croft, W. B. Time-Based Language Models. In *Proceedings of the Twelfth International Conference on Information and Knowledge Management* (New York, NY, USA, 2003), CIKM '03, ACM, pp. 469–475. URL: http://doi.acm.org/10.1145/956863.956951.

145. Li, Y., Reiss, F., and Chiticariu, L. SystemT: A Declarative Information Extraction System. In *The 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Proceedings of the Conference, 19-24 June, 2011, Portland, Oregon, USA - System Demonstrations* (2011), pp. 109–114. URL: http://www.aclweb.org/anthology/P11-4019.

146. Lin, C.-Y. ROUGE: Recall-Oriented Understudy of Gisting Evaluation. A Software Package for Automated Evaluation of Summaries, 2015. URL: http://www.berouge.com/Pages/default.aspx.

147. Ling, X., and Weld, D. S. Temporal Information Extraction. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2010, Atlanta, Georgia, USA, July 11-15, 2010* (2010). URL: http://www.aaai.org/ocs/index.php/AAAI/AAAI10/paper/view/1805.

148. MacKay, D. J. C., and Peto, L. C. B. A Hierarchical Dirichlet Language Model. *Natural Language Engineering 1* (9 1995), 289–308. URL: http://journals.cambridge.org/article_S1351324900000218.

149. man Au Yeung, C., and Jatowt, A. Studying How the Past is Remembered: Towards Computational History through Large Scale Text Mining. In *CIKM* (2011), pp. 1231–1240.

150. Manning, C. D., Raghavan, P., and Schütze, H. *Introduction to Information Retrieval.* Cambridge University Press, New York, NY, USA, 2008.

151. Manning, C. D., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S. J., and McClosky, D. The Stanford CoreNLP Natural Language Processing Toolkit. In

Association for Computational Linguistics (ACL) System Demonstrations (2014), pp. 55–60. URL: `http://www.aclweb.org/anthology/P/P14/P14-5010`.

152. Massey, F. J. The Kolmogorov-Smirnov Test for Goodness of Fit. *Journal of the American Statistical Association 46*, 253 (1951), 68–78. URL: `http://www.jstor.org/stable/2280095`.

153. Mazur, P. P., and Dale, R. WikiWars: A New Corpus for Research on Temporal Expressions. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, EMNLP 2010, 9-11 October 2010, MIT Stata Center, Massachusetts, USA, A Meeting of SIGDAT, a Special Interest Group of the ACL* (2010), pp. 913–922. URL: `http://www.aclweb.org/anthology/D10-1089`.

154. Metzger, S., Elbassuoni, S., Hose, K., and Schenkel, R. S3K: Seeking Statement-Supporting Top-K Witnesses. In *Proceedings of the 20th ACM Conference on Information and Knowledge Management, CIKM 2011, Glasgow, United Kingdom, October 24-28, 2011* (2011), pp. 37–46. URL: `http://doi.acm.org/10.1145/2063576.2063587`.

155. Metzler, D., Jones, R., Peng, F., and Zhang, R. Improving Search Relevance for Implicitly Temporal Queries. In *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval* (New York, NY, USA, 2009), SIGIR '09, ACM, pp. 700–701. URL: `http://doi.acm.org/10.1145/1571941.1572085`.

156. Michel, J.-B., Shen, Y. K., Aiden, A. P., Veres, A., Gray, M. K., , Pickett, J. P., Hoiberg, D., Clancy, D., Norvig, P., Orwant, J., Pinker, S., Nowak, M. A., and Aiden, E. L. Quantitative Analysis of Culture Using Millions of Digitized Books. *Science 331*, 6014 (2011), 176–182. URL: `http://science.sciencemag.org/content/331/6014/176`, arXiv:http://science.sciencemag.org/content/331/6014/176.full.pdf.

157. Miller, R. C., and Myers, B. A. Lightweight Structured Text Processing. *USENIX Annual Technical Conference, General Track* (1999). URL: `http://dblp.org/rec/conf/usenix/MillerM99`.

158. Milne, D., and Witten, I. H. An Effective, Low-Cost Measure of Semantic Relatedness Obtained from Wikipedia Links. In *Proceeding of AAAI Workshop on Wikipedia and Artificial Intelligence: an Evolving Synergy* (July 2008), AAAI Press, pp. 25–30. URL: `https://www.aaai.org/Papers/Workshops/2008/WS-08-15/WS08-15-005.pdf`.

159. Min, F., Wu, X., and Lu, Z. Pattern Matching with Independent Wildcard Gaps. In *Eighth IEEE International Conference on Dependable, Autonomic and Secure Computing, DASC 2009, Chengdu, China, 12-14 December, 2009* (2009), pp. 194–199. URL: `https://doi.org/10.1109/DASC.2009.65`.

160. Mitchell, T. M., Cohen, W. W., Jr., E. R. H., Talukdar, P. P., Yang, B., Betteridge, J., Carlson, A., Mishra, B. D., Gardner, M., Kisiel, B., Krishnamurthy, J., Lao, N., Mazaitis, K., Mohamed, T., Nakashole, N., Platanios, E. A., Ritter, A., Samadi, M., Settles, B., Wang, R. C., Wijaya, D., Gupta, A., Chen, X., Saparov, A., Greaves, M., and Welling, J. Never-Ending Learning. *Commun. ACM 61*, 5 (2018), 103–115. URL: `http://doi.acm.org/10.1145/3191513`.

161. Nakashole, N., Theobald, M., and Weikum, G. Scalable Knowledge Harvesting with High Precision and High Recall. In *Proceedings of the Forth International Conference on Web Search and Web Data Mining, WSDM 2011, Hong Kong, China, February 9-12, 2011* (2011), pp. 227–236. URL: `http://doi.acm.org/10.1145/1935826.1935869`.

162. Nakashole, N., Weikum, G., and Suchanek, F. M. PATTY: A Taxonomy of Relational Patterns with Semantic Types. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, EMNLP-CoNLL 2012, July 12-14, 2012, Jeju Island, Korea* (2012), pp. 1135–1145. URL: `http://www.aclweb.org/anthology/D12-1104`.

163. Naumann, F., and Herschel, M. *An Introduction to Duplicate Detection.* Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2010. URL: https://doi.org/10.2200/S00262ED1V01Y201003DTM003.

164. Nguyen, T. N., and Kanhabua, N. Leveraging Dynamic Query Subtopics for Time-Aware Search Result Diversification. In *Advances in Information Retrieval - 36th European Conference on IR Research, ECIR 2014, Amsterdam, The Netherlands, April 13-16, 2014. Proceedings* (2014), pp. 222–234. URL: http://dx.doi.org/10.1007/978-3-319-06028-6_19.

165. Nguyen, T. N., Kanhabua, N., and Nejdl, W. Multiple Models for Recommending Temporal Aspects of Entities. In *The Semantic Web - 15th International Conference, ESWC 2018, Heraklion, Crete, Greece, June 3-7, 2018, Proceedings* (2018), pp. 462–480. URL: https://doi.org/10.1007/978-3-319-93417-4_30.

166. Niu, F., Zhang, C., Ré, C., and Shavlik, J. W. Elementary: Large-Scale Knowledge-Base Construction via Machine Learning and Statistical Inference. *Int. J. Semantic Web Inf. Syst. 8*, 3 (2012), 42–73. URL: https://doi.org/10.4018/jswis.2012070103.

167. Novak, J., Micheel, I., Melenhorst, M. S., Wieneke, L., Düring, M., Moron, J. G., Pasini, C., Tagliasacchi, M., and Fraternali, P. HistoGraph - A Visualization Tool for Collaborative Analysis of Networks from Historical Social Multimedia Collections. In *18th International Conference on Information Visualisation, IV 2014, Paris, France, July 16-18, 2014* (2014), pp. 241–250. URL: http://dx.doi.org/10.1109/IV.2014.47.

168. Nunes, S., Ribeiro, C., and David, G. Use of Temporal Expressions in Web Search. In *Advances in Information Retrieval*, C. Macdonald, I. Ounis, V. Plachouras, I. Ruthven, and R. White, Eds., vol. 4956 of *Lecture Notes in Computer Science.* Springer Berlin Heidelberg, 2008, pp. 580–584. URL: http://dx.doi.org/10.1007/978-3-540-78646-7_59.

169. Odijk, D., Gârbacea, C., Schoegje, T., Hollink, L., de Boer, V., Ribbens, K., and van Ossenbruggen, J. Supporting Exploration of Historical Perspectives Across Collections. In *Research and Advanced Technology for Digital Libraries - 19th International Conference on Theory and Practice of Digital Libraries, TPDL 2015, Poznań, Poland, September 14-18, 2015. Proceedings* (2015), pp. 238–251. URL: http://dx.doi.org/10.1007/978-3-319-24592-8_18.

170. Panev, K., and Berberich, K. Phrase Queries with Inverted + Direct Indexes. In *Web Information Systems Engineering - WISE 2014 - 15th International Conference, Thessaloniki, Greece, October 12-14, 2014, Proceedings, Part I* (2014), pp. 156–169. URL: http://dx.doi.org/10.1007/978-3-319-11749-2_13.

171. Paulheim, H. Knowledge Graph Refinement: A Survey of Approaches and Evaluation Methods. *Semantic Web 8*, 3 (2017), 489–508. URL: https://doi.org/10.3233/SW-160218.

172. Peetz, M., Meij, E., and de Rijke, M. Using Temporal Bursts for Query Modeling. *Inf. Retr. 17*, 1 (2014), 74–108. URL: https://doi.org/10.1007/s10791-013-9227-2.

173. Piscopo, A., Vougiouklis, P., Kaffee, L., Phethean, C., Hare, J. S., and Simperl, E. What do Wikidata and Wikipedia Have in Common?: An Analysis of their Use of External References. In *Proceedings of the 13th International Symposium on Open Collaboration, OpenSym 2017, Galway, Ireland, August 23-25, 2017* (2017), pp. 1:1–1:10. URL: https://doi.org/10.1145/3125433.3125445.

174. Qamra, A., Tseng, B. L., and Chang, E. Y. Mining Blog Stories Using Community-Based and Temporal Clustering. In *Proceedings of the 2006 ACM CIKM International Conference on Information and Knowledge Management, Arlington, Virginia, USA, November 6-11, 2006* (2006), pp. 58–67. URL: http://doi.acm.org/10.1145/1183614.1183627.

175. Radinsky, K., Davidovich, S., and Markovitch, S. Learning Causality for News Events Prediction. In *Proceedings of the 21st World Wide Web Conference 2012, WWW 2012, Lyon, France, April 16-20, 2012* (2012), pp. 909–918. URL: `https://doi.org/10.1145/2187836.2187958`.

176. Radinsky, K., Davidovich, S., and Markovitch, S. Learning to Predict from Textual Data. *J. Artif. Intell. Res. (JAIR) 45* (2012), 641–684. URL: `http://dx.doi.org/10.1613/jair.3865`.

177. Reinanda, R., Meij, E., and de Rijke, M. Mining, Ranking and Recommending Entity Aspects. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval, Santiago, Chile, August 9-13, 2015* (2015), pp. 263–272. URL: `http://doi.acm.org/10.1145/2766462.2767724`.

178. Robertson, S. E., and Jones, K. S. Relevance Weighting of Search Terms. *JASIS 27*, 3 (1976), 129–146. URL: `https://doi.org/10.1002/asi.4630270302`.

179. Salminen, A., and Tompa, F. W. PAT Expressions: An Algebra for Text Search. *Acta Linguistica Hungarica* (1994). URL: `http://users.jyu.fi/~airi/papers/COMPLEX-1992.pdf`.

180. Samet, H., Sankaranarayanan, J., Lieberman, M. D., Adelfio, M. D., Fruin, B. C., Lotkowski, J. M., Panozzo, D., Sperling, J., and Teitler, B. E. Reading News with Maps by Exploiting Spatial Synonyms. *Commun. ACM 57*, 10 (2014), 64–77. URL: `http://doi.acm.org/10.1145/2629572`.

181. Santos, R. L. T., Macdonald, C., and Ounis, I. Search Result Diversification. *Foundations and Trends® in Information Retrieval 9*, 1 (2015), 1–90. URL: `http://dx.doi.org/10.1561/1500000040`.

182. Sarker, A., Ginn, R., Nikfarjam, A., O'Connor, K., Smith, K., Jayaraman, S., Upadhaya, T., and Gonzalez, G. Utilizing Social Media Data for Pharmacovigilance: A Review. *Journal of Biomedical Informatics 54* (2015), 202 – 212. URL: `http://www.sciencedirect.com/science/article/pii/S1532046415000362`.

183. Savenkov, D., and Agichtein, E. When a Knowledge Base Is Not Enough: Question Answering over Knowledge Bases with External Text Data. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval, SIGIR 2016, Pisa, Italy, July 17-21, 2016* (2016), pp. 235–244. URL: `http://doi.acm.org/10.1145/2911451.2911536`.

184. Schuhmacher, M., Dietz, L., and Ponzetto, S. P. Ranking Entities for Web Queries Through Text and Knowledge. In *Proceedings of the 24th ACM International Conference on Information and Knowledge Management, CIKM 2015, Melbourne, VIC, Australia, October 19 - 23, 2015* (2015), pp. 1461–1470. URL: `http://doi.acm.org/10.1145/2806416.2806480`.

185. Selinger, P. G., Astrahan, M. M., Chamberlin, D. D., Lorie, R. A., and Price, T. G. Access Path Selection in a Relational Database Management System. In *Proceedings of the 1979 ACM SIGMOD International Conference on Management of Data, Boston, Massachusetts, USA, May 30 - June 1.* (1979), pp. 23–34. URL: `https://doi.org/10.1145/582095.582099`.

186. Strötgen, J., Alonso, O., and Gertz, M. Identification of Top Relevant Temporal Expressions in Documents. In *2nd Temporal Web Analytics Workshop, TempWeb '12, Lyon, France, April 16-17, 2012* (2012), pp. 33–40. URL: `https://doi.org/10.1145/2169095.2169102`.

187. Strötgen, J., and Gertz, M. Event-Centric Search and Exploration in Document Collections. In *Proceedings of the 12th ACM/IEEE-CS Joint Conference on Digital Libraries, JCDL '12, Washington, DC, USA, June 10-14, 2012* (2012), pp. 223–232. URL: `http://doi.acm.org/10.1145/2232817.2232859`.

188. Strötgen, J., and Gertz, M. Multilingual and Cross-domain Temporal Tagging. *Language Resources and Evaluation 47*, 2 (2013), 269–298.

189. Strötgen, J., and Gertz, M. Proximity2-Aware Ranking for Textual, Temporal, and Geographic Queries. In *Proceedings of the 22Nd ACM International Conference on Information & Knowledge Management* (New York, NY, USA, 2013), CIKM '13, ACM, pp. 739–744. URL: http://doi.acm.org/10.1145/2505515.2505640.

190. Suchanek, F. M., Kasneci, G., and Weikum, G. YAGO: A Large Ontology from Wikipedia and WordNet. *Web Semant. 6*, 3 (Sept. 2008), 203–217. URL: http://dx.doi.org/10.1016/j.websem.2008.06.001.

191. Swan, R. C., and Allan, J. Automatic Generation of Overview Timelines. In *SIGIR 2000: Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, July 24-28, 2000, Athens, Greece* (2000), pp. 49–56. URL: https://doi.org/10.1145/345508.345546.

192. Tahmasebi, N., Borin, L., Capannini, G., Dubhashi, D., Exner, P., Forsberg, M., Gossen, G., Johansson, F. D., Johansson, R., Kågebäck, M., Mogren, O., Nugues, P., and Risse, T. Visions and Open Challenges for a Knowledge-Based Culturomics. *International Journal on Digital Libraries 15*, 2 (2015), 169–187. URL: http://dx.doi.org/10.1007/s00799-015-0139-1.

193. Talukdar, P. P., Wijaya, D. T., and Mitchell, T. M. Coupled Temporal Scoping of Relational Facts. In *Proceedings of the Fifth International Conference on Web Search and Web Data Mining, WSDM 2012, Seattle, WA, USA, February 8-12, 2012* (2012), pp. 73–82. URL: http://doi.acm.org/10.1145/2124295.2124307.

194. Teh, Y. W. Dirichlet Process. In *Encyclopedia of Machine Learning*. 2010, pp. 280–287. URL: https://doi.org/10.1007/978-0-387-30164-8_219.

195. Teh, Y. W., Jordan, M. I., Beal, M. J., and Blei, D. M. Sharing Clusters among Related Groups: Hierarchical Dirichlet Processes. In *Advances in Neural Information Processing Systems 17 [Neural Information Processing Systems, NIPS 2004, December 13-18, 2004, Vancouver, British Columbia, Canada]* (2004), pp. 1385–1392. URL: http://papers.nips.cc/paper/2698-sharing-clusters-among-related-groups-hierarchical-dirichlet-processes.

196. Teh, Y. W., Jordan, M. I., Beal, M. J., and Blei, D. M. Hierarchical Dirichlet Processes. *Journal of the American Statistical Association 101*, 476 (2006), 1566–1581.

197. Therneau, T., Atkinson, B., and Ripley, B. *rpart: Recursive Partitioning and Regression Trees*, 2014. R package version 4.1-8. URL: http://CRAN.R-project.org/package=rpart.

198. Tran, N. K., Tran, T. A., and Niederée, C. Beyond Time: Dynamic Context-Aware Entity Recommendation. In *The Semantic Web - 14th International Conference, ESWC 2017, Portorož, Slovenia, May 28 - June 1, 2017, Proceedings, Part I* (2017), pp. 353–368. URL: https://doi.org/10.1007/978-3-319-58068-5_22.

199. Tylenda, T., Kondreddi, S. K., and Weikum, G. Spotting Knowledge Base Facts in Web Texts. In *Proceedings of the 4th Workshop on Automated Knowledge Base Construction* (2014), pp. 1–6.

200. Verhagen, M., Mani, I., Sauri, R., Littman, J., Knippen, R., Jang, S. B., Rumshisky, A., Phillips, J., and Pustejovsky, J. Automating Temporal Annotation with TARSQI. In *ACL 2005, 43rd Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference, 25-30 June 2005, University of Michigan, USA* (2005). URL: http://acl.ldc.upenn.edu/P/P05/P05-3021.pdf.

201. Wald, A., and Wolfowitz, J. On a Test Whether Two Samples are from the Same Population. *The Annals of Mathematical Statistics 11*, 2 (1940), pp. 147–162. URL: http://www.jstor.org/stable/2235872.

202. Williams, H. E., Zobel, J., and Bahle, D. Fast Phrase Querying with Combined Indexes. *ACM Trans. Inf. Syst. 22*, 4 (Oct. 2004), 573–594. URL: http://doi.acm.org/10.1145/1028099.1028102.

203. Winkler, W. E. String Comparator Metrics and Enhanced Decision Rules in the Fellegi-Sunter Model of Record Linkage.

204. Xu, L., Bedrick, E. J., Hanson, T., and Restrepo, C. A Comparison of Statistical Tools for Identifying Modality in Body Mass Distributions. *Journal of Data Science 12*, 1 (2014), 175–196.

205. Yahya, M., Barbosa, D., Berberich, K., Wang, Q., and Weikum, G. Relationship Queries on Extended Knowledge Graphs. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining, San Francisco, CA, USA, February 22-25, 2016* (2016), pp. 605–614. URL: http://doi.acm.org/10.1145/2835776.2835795.

206. Yang, M., Ding, B., Chaudhuri, S., and Chakrabarti, K. Finding Patterns in a Knowledge Base using Keywords to Compose Table Answers. *PVLDB 7*, 14 (2014), 1809–1820. URL: http://www.vldb.org/pvldb/vol7/p1809-yang.pdf.

207. Yeung, C. A., and Jatowt, A. Studying How the Past is Remembered: Towards Computational History through Large Scale Text Mining. In *Proceedings of the 20th ACM Conference on Information and Knowledge Management, CIKM 2011, Glasgow, United Kingdom, October 24-28, 2011* (2011), pp. 1231–1240. URL: http://doi.acm.org/10.1145/2063576.2063755.

208. yew Lin, C. Rouge: a Package for Automatic Evaluation of Summaries. In *Proceedings of the ACL Workshop: Text Summarization Braches Out 2004* (01 2004), pp. 25–26.

209. Zhai, C., and Massung, S. *Text Data Management and Analysis: A Practical Introduction to Information Retrieval and Text Mining.* Association for Computing Machinery and Morgan & Claypool, New York, NY, USA, 2016.

210. Zhang, C., Ré, C., Cafarella, M. J., Shin, J., Wang, F., and Wu, S. DeepDive: Declarative Knowledge Base Construction. *Commun. ACM 60*, 5 (2017), 93–102. URL: https://doi.org/10.1145/3060586.

211. Zhang, R., Konda, Y., Dong, A., Kolari, P., Chang, Y., and Zheng, Z. Learning Recurrent Event Queries for Web Search. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing* (Stroudsburg, PA, USA, 2010), EMNLP '10, Association for Computational Linguistics, pp. 1129–1139. URL: http://dl.acm.org/citation.cfm?id=1870658.1870768.

212. Zhang, S., and Balog, K. On-The-Fly Table Generation. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, SIGIR 2018, Ann Arbor, MI, USA, July 08-12, 2018* (2018), pp. 595–604. URL: http://doi.acm.org/10.1145/3209978.3209988.

213. Zhang, V., Rey, B., Stipp, E., and Jones, R. Geomodification in Query Rewriting. In *GIR '06: Proceedings of the Workshop on Geographic Information Retrieval, SIGIR 2006* (2006).

214. Zhou, M., Cheng, T., and Chang, K. C.-C. Data-oriented Content Query System - Searching for Data into Text on the Web. *Proceedings of the Third International Conference on Web Search and Web Data Mining, WSDM 2010, New York, NY, USA, February 4-6, 2010* (2010), 121–130. URL: http://portal.acm.org/citation.cfm?doid=1718487.1718503.

215. Zhu, X., Ghahramani, Z., and Lafferty, J. Time-Sensitive Dirichlet Process Mixture Models. Tech. rep., DTIC Document, 2005.

216. Zukowski, M., Heman, S., Nes, N., and Boncz, P. Super-Scalar RAM-CPU Cache Compression. In *Proceedings of the 22nd International Conference on Data Engineering* (Washington, DC, USA, 2006), ICDE '06, IEEE Computer Society, pp. 59–59. URL: http://dx.doi.org/10.1109/ICDE.2006.150.