Methods and Tools for Temporal Knowledge Harvesting

Dissertation
zur Erlangung des Grades
Doktor der Ingenieurwissenschaften (Dr.-Ing.)
der Naturwissenschaftlich-Technischen Fakultät I
der Universität des Saarlandes

Yafang Wang
Max-Planck-Institut für Informatik

Saarbrücken
2013

Dekan der Naturwissenschaftlich-Technischen Fakultät I	Univ.-Prof. Mark Groves
Vorsitzender der Prüfungskommission	Prof. Dr. Dietrich Klakow
Berichterstatter	Prof. Dr.-Ing. Gerhard Weikum
Berichterstatter	Prof. Dr.-Ing. Martin Theobald
Beisitzer	Dr.-Ing. Klaus Berberich
Tag des Promotionskollquiums	25.February.2013

Eidesstattliche Versicherung

Hiermit versichere ich an Eides statt, dass ich die vorliegende Arbeit selbstständig und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe.

Die aus anderen Quellen oder indirekt übernommenen Daten und Konzepte sind unter Angabe der Quelle gekennzeichnet.

Die Arbeit wurde bisher weder im In- noch im Ausland in gleicher oder ähnlicher Form in einem Verfahren zur Erlangung eines akademischen Grades vorgelegt.

Saarbrücken, den 15.01.2013

(Yafang Wang)

To my Mum, Chuanwei Zhu
To my Dad, Juyuan Wang

读万卷书
行万里路
交四方友

Acknowledgment
I would like to express my profound appreciation to my advisor, Prof. Dr.-Ing. Gerhard Weikum, for his extreme patience and support, and encouragement throughout my doctoral studies. Gerhard gave me the freedom to explore on my own, and at the same time offered the guidance to keep me on track. I am also greatly thankful for his careful reading and commenting on the manuscript. His excellent and friendly advice was essential to the completion of this dissertation.

I am also deeply indebted to my colleagues with whom I worked on many publications that I published throughout the procedure of this dissertation. I would like to thank Prof. Dr.-Ing. Martin Theobald and Maximilian Dylla, for the stimulating discussions, for the sleepless nights we were working together before deadlines. I am grateful to Dr. Bin Yang and Lizhen Qu, for both the help of technical details of the work, and the fun we have had together in Saarbrücken. I wish to express my gratitude to Dr. Mingjie Zhu, Dr. Marc Spaniol, Dr. Spyros Zoupanos, Zhaochun Ren and Mohamed Yahya, for their help in many details of the work. I appreciate Kim Pecina and Maximilian Dylla, who were also very generous in helping me to translate the abstract and summary of this dissertation into German.

I am very thankful to all my colleagues at MPI and all my friends, with whom I have had lots of inspiring discussions and made great memories lasting forever. I also very much appreciate the help from Petra Schaaf and Andrea Ruffing, who were always ready for my questions about both my life in Germany and business issues. Last but not least, I would like to thank my family and in particular my parents, who have been a constant source of love, concern, support and strength all these years.

Abstract

To extend the traditional knowledge base with temporal dimension, this thesis offers methods and tools for harvesting temporal facts from both semi-structured and textual sources. Our contributions are briefly summarized as follows.

1. **Timely YAGO:** A temporal knowledge base called Timely YAGO (T-YAGO) which extends YAGO with temporal attributes is built. We define a simple RDF-style data model to support temporal knowledge.

2. **PRAVDA:** To be able to harvest as many temporal facts from free-text as possible, we develop a system PRAVDA. It utilizes a graph-based semi-supervised learning algorithm to extract fact observations, which are further cleaned up by an Integer Linear Program based constraint solver. We also attempt to harvest spatio-temporal facts to track a person's trajectory.

3. **PRAVDA-live:** A user-centric interactive knowledge harvesting system, called PRAVDA-live, is developed for extracting facts from natural language free-text. It is built on the framework of PRAVDA. It supports fact extraction of user-defined relations from ad-hoc selected text documents and ready-to-use RDF exports.

4. **T-URDF:** We present a simple and efficient representation model for time-dependent uncertainty in combination with first-order inference rules and recursive queries over RDF-like knowledge bases. We adopt the common possible-worlds semantics known from probabilistic databases and extend it towards histogram-like confidence distributions that capture the validity of facts across time.

All of these components are fully implemented systems, which together form an integrative architecture. PRAVDA and PRAVDA-live aim at gathering new facts (particularly temporal facts), and then T-URDF reconciles them. Finally these facts are stored in a (temporal) knowledge base, called T-YAGO. A SPARQL-like time-aware querying language, together with a visualization tool, are designed for T-YAGO. Temporal knowledge can also be applied for document summarization.

Abstract

Diese Dissertation zeigt Methoden und Werkzeuge auf, um traditionelle Wissensbasen um zeitliche Fakten aus semi-strukturierten Quellen und Textquellen zu erweitern. Unsere Arbeit lässt sich wie folgt zusammenfassen.

1. **Timely YAGO:** Wir konstruieren eine Wissensbasis, genannt 'Timely YAGO' (T-YAGO), die YAGO um temporale Attribute erweitert. Zusätzlich definieren wir ein einfaches RDF-ähnliches Datenmodell, das temporales Wissen unterstützt.

2. **PRAVDA:** Um eine möglichst große Anzahl von temporalen Fakten aus Freitext extrahieren zu können, haben wir das PRAVDA-System entwickelt. Es verwendet einen auf Graphen basierenden halbüberwachten Lernalgorithmus, um Feststellungen über Fakten zu extrahieren, die von einem Constraint-Solver, der auf einem ganzzahligen linearen Programm beruht, bereinigt werden. Wir versuchen zudem räumlich-temporale Fakten zu extrahieren, um die Bewegungen einer Person zu verfolgen.

3. **PRAVDA-live:** Wir entwickeln ein benutzerorientiertes, interaktives Wissensextrahiersystem namens PRAVDA-live, das Fakten aus freier, natürlicher Sprache extrahiert. Es baut auf dem PRAVDA-Framework auf. PRAVDA-live unterstützt die Erkennung von benutzerdefinierten Relationen aus ad-hoc ausgewählten Textdokumenten und den Export der Daten im RDF-Format.

4. **T-URDF:** Wir stellen ein einfaches und effizientes Repräsentationsmodell für zeitabhängige Ungewissheit in Verbindung mit Deduktionsregeln in Prädikatenlogik erster Stufe und rekursive Anfragen über RDF-ähnliche Wissensbasen vor. Wir übernehmen die gebräuchliche Mögliche-Welten-Semantik, bekannt durch probabilistische Datenbanken

und erweitern sie in Richtung histogrammähnlicher Konfidenzverteilungen, die die Gültigkeit von Fakten über die Zeit betrachtet darstellen.

Alle Komponenten sind vollständig implementierte Systeme, die zusammen eine integrative Architektur bilden. PRAVDA und PRAVDA-live zielen darauf ab, neue Fakten (insbesondere zeitliche Fakten) zu sammeln, und T-URDF gleicht sie ab. Abschließend speichern wir diese Fakten in einer (zeitlichen) Wissensbasis namens T-YAGO ab. Eine SPARQL-ähnliche zeitunterstützende Anfragesprache wird zusammen mit einem Visualisierungswerkzeug für T-YAGO entwickelt. Temporales Wissen kann auch zur Dokumentzusammenfassung genutzt werden.

Summary

To extend the traditional knowledge base with temporal dimension, this thesis offers methods and tools for harvesting temporal facts from both semi-structured and textual sources. Our contributions are briefly summarized as follows.

1. **Timely YAGO:** A temporal knowledge base called Timely YAGO (T-YAGO) which extends YAGO with temporal attributes is built. We define a simple RDF-style data model to support temporal knowledge. For temporal facts valid at a time point, we use the relation *on* to describe the validity time; for those valid during a time period, we use the relation *since* for the begin time point, and the relation *until* for the end time point. Most of the temporal facts are extracted from semi-structured data (e.g., infoboxes, categories and lists of Wikipedia articles) via a rule-based method. A SPARQL-like time aware query language is provided for querying. The results are shown via a timeline visualization tool.

2. **PRAVDA:** To be able to harvest as many temporal facts from free-text as possible, we develop a system PRAVDA. It utilizes a graph-based semi-supervised learning algorithm, which guarantees a high recall and comparable precision with limited human created seeds. The graph is composed of fact candidate nodes and pattern nodes. Edges between fact candidate nodes and pattern nodes are introduced whenever a fact candidate appears with a pattern. Their weight is derived from the co-occurrence frequency. Edges between pattern nodes are created if their type pair is identical and the degree of their similarity delivers the edge's weight. We use one label for each observation type (*begin*, *during*, and *end*) of each relation. Then label propagation is exploited to determine the label of each node. Once a fact candidate is labelled with a particular relation, it is called a *valid fact observation*. Afterwards, an Integer Linear

Program is used to clean out false hypotheses that violate temporal constraints. We also attempt to harvest spatio-temporal facts to track a person's trajectory.

3. **PRAVDA-live:** A user-centric interactive knowledge harvesting system, called PRAVDA-live, is developed for extracting facts from natural language free-text. It is built on the framework of PRAVDA. It supports fact extraction of user-defined relations from ad-hoc selected text documents and ready-to-use RDF exports. Further features include support for temporally annotated relations, custom or mined extraction-patterns, and a constraint solver able to clean extracted fact observations by inter-fact constraints.

4. **T-URDF:** We present a simple and efficient representation model for time-dependent uncertainty in combination with first-order inference rules and recursive queries over RDF-like knowledge bases. We adopt the common possible-worlds semantics known from probabilistic databases and extend it towards histogram-like confidence distributions that capture the validity of facts across time. An algorithm is designed to reconcile multiple (potentially inconsistent) observations of temporal facts into a concise histogram. The *begin*, *during* and *end* fact observations are aggregated into histograms respectively. Then we perform *forward aggregation* for the *begin* histogram, *backward aggregation* for the *end* histogram, and merge them with the *during* histogram. Query processing is done via a Datalog-like, rule-based inference engine, which employs the *lineage* of derived facts for confidence computations to remain consistent with the possible-worlds model.

All of these components are fully implemented systems, which together form an integrative architecture. PRAVDA and PRAVDA-live aim at gathering new facts (particularly temporal facts), and then T-URDF reconciles them. Finally these facts are stored in a (temporal) knowledge base, called T-YAGO. A SPARQL-like time-aware querying language, together with a visualization tool, are designed for T-YAGO. Temporal knowledge can also be applied for document summarization.

Zusammenfassung

Diese Dissertation zeigt Methoden und Werkzeuge auf, um traditionelle Wissensbasen um zeitliche Fakten aus semi-strukturierten Quellen und Textquellen zu erweitern. Unsere Arbeit lässt sich wie folgt zusammenfassen.

1. **Timely YAGO:** Wir konstruieren eine Wissensbasis, genannt 'Timely YAGO' (T-YAGO), die YAGO um temporale Attribute erweitert. Zusätzlich definieren wir ein einfaches RDF-ähnliches Datenmodell, das temporales Wissen unterstützt. Die Relation *on* beschreibt zeitliche Fakten, die zu einem bestimmten Zeitpunkt gültig sind; Fakten, die über eine längere Zeitspanne gültig sind, werden durch die Relationen *since* und *until* ausgedrückt, die jeweils den Startpunkt und den Endpunkt des Zeitraums beschreiben. Der Großteil der temporalen Fakten wird aus semi-strukturierten Daten (z.B. Infoboxen, Kategorien und Listen von Wikipedia-Artikeln) durch regelbasierte Methoden extrahiert. Wir stellen eine SPARQL-ähnliche Abfragesprache mit Unterstützung der Zeitkomponente vor. Die Ergebnisse werden mit Hilfe eines Zeitleisten-Visualisierungstools präsentiert.

2. **PRAVDA:** Um eine möglichst große Anzahl von temporalen Fakten aus Freitext extrahieren zu können, haben wir das PRAVDA-System entwickelt. Es verwendet einen auf Graphen basierenden halbüberwachten Lernalgorithmus, welcher hohe Ausbeute bei vergleichbarer Präzision garantiert und mit einer geringen Anzahl von manuellen Annotationen zurechtkommt. Die Knoten des Graphen unterteilen sich in Muster-Knoten und Faktkandidat-Knoten. Kanten zwischen Muster-Knoten und Faktkandidat-Knoten werden in den Graphen eingefügt, wenn ein Faktkandidat innerhalb eines Musters im Freitext auftritt. Das Gewicht dieser Kanten wird durch die Frequenz des gemeinsamen Auftretens bestimmt. Des Weiteren fügen wir Kanten

zwischen Muster-Knoten in den Graphen ein, wenn deren Typenpaar identisch ist, wobei das Kantengewicht durch die Ähnlichkeit bestimmt wird. Für jede Relation benutzen wir drei Annotationen, welche den Beobachtungstypen (*begin*, *during* und *end*) entsprechen. Danach wird Annotationsweitergabe verwendet um die Annotation jedes Knoten im Graphen zu bestimmen. Sobald ein Faktkandidat mit einer Relation annotiert ist, nennen wir ihn *valide Faktbeobachtung*. Anschließend wird ein ganzzahliges lineares Programm verwendet, um die falschen Hypothesen mit Hilfe von temporalen Bedingungen auszufiltern. Wir versuchen außerdem räumlich-temporale Fakten zu extrahieren, um die Bewegungen einer Person zu verfolgen.

3. **PRAVDA-live:** Wir entwickeln ein benutzerorientiertes, interaktives Wissensextrahiersystem namens PRAVDA-live, das Fakten aus freier, natürlicher Sprache extrahiert. Es baut auf dem PRAVDA-Framework auf. PRAVDA-live unterstützt die Erkennung von benutzerdefinierten Relationen aus ad-hoc ausgewählten Textdokumenten und den Export der Daten im RDF-Format. Weitere Merkmale beinhalten die Unterstützung von zeitlich annotierten Relationen, maßgeschneiderten oder gewonnenen Extrahierungsmustern und einen Constraint-Solver, der extrahierte Faktbeobachtungen durch zwischenfaktische Beschränkungen bereinigt.

4. **T-URDF:** Wir stellen ein einfaches und effizientes Repräsentationsmodell für zeitabhängige Ungewissheit in Verbindung mit Deduktionsregeln in Prädikatenlogik erster Stufe und rekursive Anfragen über RDF-ähnliche Wissensbasen vor. Wir übernehmen die gebräuchliche Mögliche-Welten-Semantik, bekannt durch probabilistische Datenbanken und erweitern sie in Richtung histogrammähnlicher Konfidenzverteilungen, die die Gültigkeit von Fakten über die Zeit betrachtet darstellen. Des Weiteren designen wir einen Algorithmus, der mehrere (potentiell inkonsistente) Beobachtungen von zeitlichen Fakten in präzise Histogramme überführt. Die Fakten *begin*, *during* und *end* werden jeweils in Histogramme aggregiert. Danach führen wir eine *Vorwärtsaggregierung* für die *begin* Histogramme, eine *Rückwärtsaggregierung* für die *end* Histogramme durch und fusionieren

beide mit dem *during* Histogramm. Anfragen werden mithilfe eines datalogähnlichen, regelbasierten Deduktionssystems beantwortet, welches die *Abstammung* von hergeleiteten Fakten für Konfidenzberechnungen einbezieht, um im Einklang mit dem Mögliche-Welten-Modell zu bleiben.

Alle Komponenten sind vollständig implementierte Systeme, die zusammen eine integrative Architektur bilden. PRAVDA und PRAVDA-live zielen darauf ab, neue Fakten (insbesondere zeitliche Fakten) zu sammeln, und T-URDF gleicht sie ab. Abschließend speichern wir diese Fakten in einer (zeitlichen) Wissensbasis namens T-YAGO. Eine SPARQL-ähnliche zeitunterstützende Anfragesprache wird zusammen mit einem Visualisierungswerkzeug für T-YAGO entwickelt. Temporales Wissen kann auch zur Dokumentzusammenfassung genutzt werden.

Contents

Chapter 1. Introduction

1.1. Motivation

In recent years, automated fact extraction from Web contents has seen significant progress with the emergence of freely available knowledge bases, such as DBpedia [3], YAGO [62], KnowItAll [21]/TextRunner [89, 20], Kylin/KOG [86] or ReadTheWeb [13, 12]. These knowledge bases are constantly growing and contain currently (by example of DBpedia) several million entities and half a billion facts about them. This wealth of data supports the information needs of advanced Internet users by raising queries from keywords to entities. This enables queries like “Who is married to Prince Charles?” or “Who are the teammates of FIFA World Footballer Lionel Messi at FC Barcelona?”.

However, factual knowledge is highly ephemeral: people get married and divorced, politicians hold positions only for a limited time, and soccer players transfer from one club to another. Let us take YAGO as an example to show the purpose of this thesis. YAGO knows the facts in Table 1.1, which shows an excerpt from the YAGO knowledge base about David Beckham and Ronaldo. Obviously, these facts are time-agnostic. What if we want to know the timespan of Beckham and Ronaldo’s career in Real Madrid? Who is older? Beckham or Ronaldo? Have they ever been teammates? For the purpose of answering these questions, we aim at creating *time-aware* knowledge with validity time intervals of facts. For the example we could obtain the temporal knowledge in Table 1.2.

Clearly the temporal expressions are at diverse granularities. Some may even not be accurate especially when temporal information is retrieved from noisy

Subject	**Relation**	**Object**
`David_Beckham`	`isBornIn`	`London`
`David_Beckham`	`playsForClub`	`Real_Madrid_C.F.`
`David_Beckham`	`playsForClub`	`Los_Angeles_Galaxy`
`David_Beckham`	`winsAward`	`ESPY_Award`
`David_Beckham`	`winsAward`	`FIFA_100`
`David_Beckham`	`winsAward`	`La_Liga`
`Ronaldo`	`isBornIn`	`Rio_de_Janeiro`
`Ronaldo`	`playsForClub`	`Real_Madrid_C.F.`
`Ronaldo`	`playsForClub`	`A.C._Milan`
`Ronaldo`	`winsAward`	`FIFA_100`
`Ronaldo`	`winsAward`	`Bravo_Award`
`Ronaldo`	`winsAward`	`La_Liga`

Table 1.1.: An excerpt from the YAGO knowledge base about David Beckham and Ronaldo.

sources. The inaccuracy comes from imperfect extraction methods, careless authors of information sources, etc. For example, we might extract multiple start time points for David_Beckham playing for Los_Angeles_Galaxy, such as 2007, 2006-Dec-15 and 2007-Jan. When is the exact start time point?

Some interesting facts can be inferred with the temporal facts in Table 1.2. For example, "Beckham won La_Liga during the time he was in Real_Madrid". The teammate relationship between Beckham and Ronaldo can be inferred as follows:

`David_Beckham` playsForClub `Real_Madrid_C.F.` from `2003` to `2007`
`David_Beckham` playsForClub `Los_Angeles_Galaxy` from `2007` to `NOW`
`Ronaldo` playsForClub `Real_Madrid_C.F.` from `2002` to `2007`
`Ronaldo` playsForClub `A.C._Milan` from `2007` to `2008`

Therefore, Beckham and Ronaldo were teammates of Real_Madrid during the time from 2003 to 2007. Also they won La_Liga together for Real_Madrid by considering the additional facts:

`David_Beckham` winsCup `La_Liga` in `2007`
`Ronaldo` winsCup `La_Liga` in `2007`

Subject	Relation	Object	Start	End
David_Beckham	isBornIn	London	1975-05-02	1975-05-02
David_Beckham	playsForClub	Real_Madrid_C.F.	2003	2007
David_Beckham	playsForClub	Los_Angeles_Galaxy	2007-01	NOW
David_Beckham	winsAward	ESPY_Award	2004	2004
David_Beckham	winsAward	ESPY_Award	2008	2008
David_Beckham	winsAward	FIFA_100	2004-03-04	2004-03-04
David_Beckham	winsCup	La_Liga	2007	2007
Ronaldo	isBornIn	Rio_de_Janeiro	1976-09-18	1976-09-18
Ronaldo	playsForClub	Real_Madrid_C.F.	2002-01	2007
Ronaldo	playsForClub	A.C._Milan	2007	2008
Ronaldo	winsAward	FIFA_100	2004	2004
Ronaldo	winsAward	Bravo_Award	1995	1995
Ronaldo	winsAward	Bravo_Award	1998	1998
Ronaldo	winsCup	La_Liga	2007	2007

Table 1.2.: An excerpt from the YAGO knowledge base with temporal knowledge about David Beckham and Ronaldo.

We may further obtain the additional facts from a variety of sources:

`David_Beckham` isBornIn `London` on `1975-05-02`

`David_Beckham` isBornIn `Paris` on `1975-05-02`

`David_Beckham` isBornIn `Berlin` on `1975-05-02`

Beckham can only be born in one place on one day, so what is the birth place of Beckham? Obviously, this needs some form of data cleaning or reasoning on the uncertain information.

Relations have different flavors with respect to time aspect. Facts in relations like *isBornIn*, *winsAward* and *winsCup* are valid on one *time point*, whereas other relations like *playsForClub* are valid within a *time period* between the start and end time point.

The construction of a temporal knowledge base is a non-trivial task. We summarize the challenges in the next section.

1.2. Research Challenges

The automatic construction of a temporal knowledge base is challenging for various reasons.

Diverse Data Sources. Data sources are in various formats: structured, semi-structured and unstructured. Semi-structured and structured text are easier to be handled by regular expressions. However, the irregularities and ambiguities of unstructured text, typically free natural language text, is difficult to define rules.

Reconciliation of Facts. The inconsistency of different sources and imperfect extraction methods cause noises in extracted facts. It is thus necessary to remove these noises and derive reasonable new facts. In particular, each temporal fact requires inferring the valid time interval from its multiple observations which might be in different granularities: year, month and day. It is thus desirable to design a comprehensive model to reconcile all the temporal fact observations, especially to deal with the different kinds of relations which are valid on a time point or during a time period. Afterwards, the valid time interval is helpful for reasoning new temporal facts.

High Precision vs. High Recall. The aggressive extraction methodology usually results in a large amount of facts but lower precision; while the conservative method ensures a better precision but lower recall. The trade off between the precision and recall is thus a tricky problem. On the other hand, increasing input manually labelled seed facts is able to increase recall. However, it is highly desirable to introduce as little human effort as possible.

Interactive Knowledge Harvesting. Acquiring high-quality (temporal) facts for knowledge bases is a labor-intensive process. Although there has been recent progress in the area of semi-supervised fact extraction, these approaches still have limitations, including a restricted corpus, a fixed set of relations to be extracted or a lack of assessment capabilities. It is thus necessary to provide users with tools that allow them to interactively explore knowledge from their own documents.

Querying & Visualization. SPARQL is used for querying the time invariant knowledge bases. It is desirable to design a querying language for temporal knowledge bases, and provide a proper visualization tool to illustrate temporal knowledge.

1.3. Contributions

In this thesis, methods and tools for harvesting temporal facts from both semi-structured and textual sources are presented. Our system addresses the issues pointed out in the previous section. Our contributions are briefly summarized as follows.

1. **Timely YAGO:** A temporal knowledge base called Timely YAGO (T-YAGO) which extends YAGO with temporal attributes is built. We define a simple RDF-style data model to support temporal knowledge. Most of the temporal facts are extracted from semi-structured data (e.g., infoboxes, categories and lists of Wikipedia articles) via rule-based method. A SPARQL-like time aware query language is provided for querying. The results are shown via a timeline visualization tool. Our Timely YAGO knowledge base system was published in the proceedings of the 13th International Conference on Extending Database Technology (EDBT 2010) [85].

2. **PRAVDA:** To be able to harvest as many temporal facts from free-text as possible, we develop a system PRAVDA. It utilizes a graph-based semi-supervised learning algorithm, which guarantees a high recall and comparable precision with limited human created seeds. Afterwards, an Integer Linear Program is used to clean out false hypotheses that violate temporal constraints. We also attempt to harvest spatio-temporal facts to track a person's trajectory. Our PRAVDA temporal knowledge harvesting system was published in the proceedings of the 20th ACM Conference on Information and Knowledge Management (CIKM 2011) [83], the proceedings of the conference of the 50th Annual Meeting of the Association for Computational Linguistics (ACL 2012) [81] and the proceedings of the 20th International Conference on World Wide Web (WWW 2011) [84].

3. **PRAVDA-live:** A user-centric interactive knowledge harvesting system, called PRAVDA-live, is developed for extracting facts from natural language free-text. It supports fact extraction of user-defined relations from ad-hoc selected text documents and ready-to-use RDF exports.

Further features include support for temporally annotated relations, custom or mined extraction-patterns, and a constraint solver able to clean extracted facts by inter-fact constraints. Our interactive knowledge harvesting system was published in the proceedings of the 21th ACM Conference on Information and Knowledge Management (CIKM 2012) [80].

4. **T-URDF:** We present a simple and efficient representation model for time-dependent uncertainty in combination with first-order inference rules and recursive queries over RDF-like knowledge bases. We adopt the common possible-worlds semantics known from probabilistic databases and extend it towards histogram-like confidence distributions that capture the validity of facts across time. Query processing is done via a Datalog-like, rule-based inference engine, which employs the *lineage* of derived facts for confidence computations to remain consistent with the possible-worlds model. Our T-URDF temporal reasoning system was published in the proceedings of the Fourth International VLDB workshop on Management of Uncertain Data (MUD 2010) in conjunction with VLDB 2010 [82].

All of these components are fully implemented systems, which together form an integrative architecture. PRAVDA and PRAVDA-live aim at gathering new facts (particularly temporal facts), and then T-URDF reconciles them. Finally these facts are stored in a (temporal) knowledge base, called T-YAGO. A SPARQL-like time-aware querying language, together with a visualization tool, are designed for T-YAGO. Temporal knowledge can also be applied for document summarization.

1.4. Thesis Outline

We present temporal knowledge harvesting from both semi-structured data and free-text in Chapter 4. Chapter 5 describes how to clean and reason on the extracted temporal facts. Chapter 6 demonstrates the user-centric interactive knowledge harvesting system – PRAVDA-live. Finally Chapter 7 shows applications of temporal knowledge bases, including a time-aware

SPARQL-like query language for temporal knowledge and a document summarization system. The four main contributions, introduced in Section 1.3, are presented throughout these chapters. T-YAGO is presented in both Section 4.2 and Section 7.1.1. Section 4.3 and Section 5.2 discuss the whole framework of PRAVDA — t-fact observation extraction and cleaning. The entire Chapter 6 demonstrates PRAVDA-live. T-URDF is described in Section 5.3. We summarize our findings and highlight future research directions in Chapter 8.

Chapter 2.

Related Work

This chapter briefly reviews the literature on knowledge harvesting, temporal information extraction, label propagation, temporal and probabilistic databases, interactive systems, and summarization.

2.1. Knowledge Harvesting

Several projects have addressed the automatic construction of large knowledge bases.

Semi-structured approaches. Projects such as *DBpedia* [3] and *YAGO* [62, 29] automatically extract facts from semi-structured elements of Wikipedia, like infoboxes and categories. DBpedia uses attribute names in infoboxes as relation names instead of defining accurate relations with ranges and domains. Different relation names may refer to the same relation (e.g., length, length-in-km, length-km). YAGO defines relations with ranges and domains, and represents them in a canonical manner. YAGO combines the high quality of facts extracted from Wikipedia with the clean taxonomy of the concepts from WordNet.

Free-text approaches. Relation extraction from text aims at detecting and classifying semantic relationships between entities. Many machine learning methods have been devised to address this task. Supervised learning methods, such as [17, 95], suffer from the need for large amounts of manually labelled training data. Semi-supervised methods, such as [94], require less labelled data.

The method of [15] first attempted to apply graph-based label propagation to relation extraction. Recently distant supervision for relation extraction, such as [47, 88], has become popular. Instead of using a labelled corpus, the input of distant supervision only requires an existing knowledge base of seed facts. These advances in information extraction and the success of knowledge communities like Wikipedia have enabled the largely automated construction of fairly comprehensive knowledge bases. The KnowItAll project [21] focuses on automating the unary or binary fact extraction task on a very large scale manner. The ReadTheWeb project proposes an approach towards fact extraction based on coupled semi-supervised learning for information extraction (IE) : NELL [12, 13]. It exploits the ensemble-learning method with coupled pattern learners for retrieving both unary and binary facts. TextRunner project [20, 86] targets at fact extraction in an open-domain way, which aims at retrieving instances of all meaningful relations without pre-specification. However, all of these projects do not disambiguate entity mentions to entities. Kylin/KOG [86] aims at building a knowledge base from not only infoboxes and categories of Wikipedia articles, but also free-text. PROSPERA [49] designs an n-gram-itemsets based patterns, and applies frequent itemset mining to associate confidence to each pattern. Then the MaxSat-based reasoning is exploited to decide the true fact candidates. SOFIE [63] unifies fact extraction and cleaning into one framework. This framework includes word disambiguation, pattern matching and rule-based reasoning on the ontology. However all of the above mentioned projects do not consider temporal knowledge harvesting.

None of the aforementioned knowledge bases has a temporal dimension. They are all designed for time-invariant knowledge and built as snapshots in time. For example, while several of them know that people can have multiple spouses, none of them capture the time intervals for the various marriages. Birth and death dates of people are available, though, as they can be extracted at the level of base-facts such as `Diego_Maradona` *isBornOn* `30-October-1960`. This is simpler than the kind of temporal facts that are tackled in this thesis.

2.2. Temporal Information Extraction

Temporal knowledge as a first-class citizen in knowledge bases has been addressed solely by few prior projects: TOB [92], TIE [43], and CoTS [68]. TOB [92] focuses on extracting business-related temporal facts such as terms of CEOs. It uses an NLP machinery, with deep parsing of every sentence, and machine-learning methods for classifiers for specifically interesting relations. It workes reasonably well, but is computationally expensive, requires extensive training, and cannot be easily generalized to other relations. TIE [43] uses training data with fine-grained annotations to learn an inference model based on Markov Logic. This involves using consistency constraints on the relative ordering of events. This machinery is computationally expensive and cannot easily be scaled up. CoTS [68] determines the begin and end dates of temporal facts via a classifier. Afterwards an Integer Linear Program (ILP) infers the validity-intervals of facts based on intra- and cross-relation temporal constraints. The work by [45, 65] studies the properties of relations. [45] studies the temporal bounds of a certain relation. For instance, it can learn that a person cannot start a job at age two or get married before his/her birth. [65] analyzes whether a relation is time-dependent. For example, the *isBornIn* relation is time-invariant since one's birth place never changes. However, these work do not focus on temporal fact extraction.

The NLP community has event extraction tasks in its SemEval workshop series [74], using representations such as TimeML and reference corpora such as Timebank [10]. The TARSQI toolkit [75] provides a suite of techniques for detecting events and temporal expressions, based on a dictionary of time-related adverbial phrases and regular-expression matching. These work mainly focus on detecting events (e.g., "her birthday"), relative expressions (e.g., "last Sunday"), and time-order relations (before, after, during, etc.). [87] uses a small set of seeds to learn pattern rules for various complex relations, such as joining (leaving) a job. The algorithm works on parsed input data, which requires preprocessing by a dependency parser. Moreover, temporal mentions associated with these relations are not considered. [40] proposes a new approach to parse the temporal dependency structures among all the events in a text. These basic tasks are still far from actual fact extraction.

Temporal facts or temporal information are also considered in other fields. For instance, there is recent awareness of temporal IR: ranking of search results for keyword queries with temporal phrases [7, 50].

2.3. Label Propagation

Label propagation (LP) belongs to a family of graph-based semi-supervised learning methods. The first LP method [96] assumed that initial labels are fully correct. This assumption is inappropriate for our setting, as the seed patterns are not fully correct, but only associated with a confidence value. The Adsorption algorithm [4] copes with noisy initial labels, and is successfully used in video recommendation. The MAD algorithm [66] optimizes the Adsorption algorithm. The MADDL algorithm [66] incorporates possible dependencies on different labels into the optimization function. However, these dependencies must be symmetric, in contrast to asymmetric inclusion constraints introduced in Section 4.3 of this thesis. In a recent survey [67], MAD is regarded as the most effective label propagation algorithm across various data sets and experimental conditions. The nature of label propagation, whose solution can be iteratively approximated by the Jacobi method, makes it nicely scalable [4]. [52] describes a scalable implementation of label propagation on MapReduce to deal with Web-scale graphs.

2.4. Temporal and Probabilistic Databases

Temporal reasoning has a fairly long history through works in logics and AI, most notably in the seminal work by Allen et al. [23]. Work on temporal databases dates back to the early 1980's [33]. Different semantics for associating time with facts have been defined. In the context of this thesis, we use the valid-time semantics, where the time recorded for facts in a database captures the reality of the world being modeled by the database. Extensions for traditional data models have been explored to accommodate temporal data in an efficient manner, both in terms of storage space and query processing. There has also been an extensive effort to develop query languages for querying temporal data. Most of these efforts were attempts to modify SQL to reduce the complexity of temporal queries [69]. There is a wealth of research on probabilistic databases and the management of uncertain data. The book Probabilistic Databases [64] discusses the state of the art in representation formalisms and query processing techniques for probabilistic data. [32] is a state-of-the-art probabilistic database management system. [6, 57] present a framework for dealing with uncertain data and data lineage. This approach allows for the decoupling of data and confidence computations when processing queries over uncertain data [57], allowing for a wider range of query plans to be used while still maintaining the correctness of confidence computations. For dealing with probabilistic reasoning in the context of information retrieval, [24] presents a probabilistic version of Datalog. [25, 31, 73] present probabilistic extensions of RDF and discuss how SPARQL queries over such an extension can be supported. However, no notion of temporal reasoning has been considered in these contexts.

2.5. Interactive Systems

Freebase [11] is very related to our work, since users are encouraged to add facts. Nevertheless, there is no support for extraction from text documents. Crowdsourcing techniques have been applied to knowledge acquisition. Open Mind Common Sense (OMCS) [60] project acquires common sense knowledge from volunteers. Verbosity [76] achieves the same purpose in the form of an enjoyable game. [61] connects the result data from Verbosity to the OMCS project. CrowdER [79] develops a hybrid human-machine approach for entity resolution, where the machine passes over the data to generate the most likely matching pairs for people to verify. [35] exploits the wisdom of crowds to derive taxonomies. CrowdMap [56] studies the use of crowdsourcing for ontology alignment. [91] empirically studies the impact on precision and recall of changing the size of two types of resources: labelled data from distant supervision and crowdsourcing. The crowd is employed on collecting seed facts by sampling the training data. There are other crowdsourcing activities for data annotation such as KiWi [58] or Semantic Wiki [41]. However, those efforts do not aim at fact extraction for knowledge bases in terms of clarity and scale. Especially, none of the systems is designed for users performing fact extraction on any relations of interest.

2.6. Summarization

The two main approaches of document summarization strategies are abstractive and extractive summarization techniques [27]. Most of the existing work in abstractive document summarization usually considers advanced language generation techniques to produce a grammatical summary [18]. Some employ a form of Nature Language Generation (NLG) [46]. Some other approaches construct abstractive summaries by using the Cut-And-Paste algorithm [34], which compresses sentences by removing redundant information. Similar work use corpus-based methods for sentence compression [39]. In recent years, several parsing-free abstractive summarization strategies have been proposed. Opinosis [26] is proposed to generate summarization from redundant data sources by building a single graph-based representation. Similarly, a graph-based multi-sentence compression approach has been proposed in [22].

Extractive document summarization frameworks, which usually extract summaries based on machine learning methods, can be categorized into unsupervised methods and supervised methods. For the unsupervised methods, MEAD [51] is proposed based on a centroid-clustering-based strategy. After the PageRank method was proposed, several Markov Random Walk based methods [19, 77] were proposed to rank sentences by their saliency. A HITS-based ranking algorithm [90] has also been proposed for document summarization.

After topic models [9] were proposed, LDA has also been applied to document summarization [2, 78] based on various probabilistic methods for topic detection and clustering. Other related methods [14] consider the summarization task as a prediction problem based on a two-step hybrid generative model. Topic models could deal with sentence understanding at the topic-level by extracting latent semantic components. However, for a query-biased summarization task, usually those latent topics could not represent what "topic" the user exactly intends to search. For the supervised methods, HMM-based [16] and CRF-based algorithms [59] have proved to be effective in extractive document summarization. Meanwhile, structured SVM-based classifiers have also been applied [42] to document summarization, with the aim of enhancing the diversity, coverage and balance for the summary result.

All these work focus on documents in natural language text, which is different with our strategy by applying summarization for both semi-structured and unstructured contents on the web. Moreover, these extractive methods only extract sentences from unstructured data, however, our approach represents the semantics from both unstructured and semi-structured data.

There are also some works that aim to summarize factual information from a knowledge base. [93] defines the "RDF sentence" to summarize an ontology by ranking the ontology graph. MING [38] extracts an "informative" subgraph for given entity nodes in a knowledge base. [72] retrieves the salient type properties for a certain entity. These work only consider the existing knowledge base, and the "summary" is in the form of a subgraph or type properties, while our work automatically harvests the knowledge from data sources and represents the summary of queried relations in a nicely (chronologically) ordered set of natural-language sentences.

Chapter 3.

Knowledge Representation

3.1. Relation and Fact Types

We introduce the distinction between *base relations* (without temporality) and *temporal relations* (including temporality). *Base relations* indicate the relationships between two entities without any temporal information. A base relation R_b is commonly described with a type signature, $\mathtt{sig}(R_b) = (\mathit{TYPE}_1, \mathit{TYPE}_2)$, meaning that the relation R_b only holds its semantic meaning for two entities whose types are TYPE_1 and TYPE_2, respectively. A base relation contains a set of base facts. Each base fact is in the form of (e_i, e_j), where e_i and e_j indicate two entities whose types are consistent with the corresponding base relation's type signature. For example, *worksForClub* is a base relation with signature (*PERSON*, *CLUB*). The base fact `(Lionel_Messi, FC_Barcelona)` holds for the *worksForClub* relation, where `Lionel_Messi` is a *PERSON* entity and `FC_Barcelona` is a *CLUB* entity.

Temporal relations indicate the relationships between two entities at specific time points or during particular timespans. Similar to base relations, a temporal relation is always associated with a type signature indicating the pertinent entity types. A temporal relation contains a set of temporal facts in the form of (e_i, e_j)@t_k, where e_i and e_j are two entities with pertinent types, and t_k indicates a temporal mention (either a time point or a timespan). For example, *joinsClubTemp* is a temporal relation with type signature (*PERSON*, *CLUB*). The temporal fact `(David_Beckham, Real_Madrid)@2005` holds for the *joinsClubTemp* relation, indicating that `David_Beckham` joined `Real_Madrid` in 2005.

A *fact candidate* is a pair of entities that occurs in a textual source (plus a time stamp in the case of temporal fact candidates), where the pair of entities must satisfy the type signature of any of the relations of interest. Once a fact candidate is labelled with a particular relation R, it is called a *valid observation* and added to the set of *fact observations* that are returned as the result of the knowledge harvesting phase. In particular, we distinguish the *base fact observation* (b-observation) with the *temporal fact observation* (t-observation). After the *knowledge cleaning* stage (see Chapter 5), noisy fact observations are filtered out. The remaining clean fact observations are called true *facts*. Also we distinguish the *base fact* (b-fact) with the *temporal fact* (t-fact) respectively.

3.2. Time Points, Intervals, and Histograms

A time point t denotes the smallest time unit of fixed granularity. We have a discrete series of ordered time points $0, \ldots, N$ (with a special designator N which marks the end of the time range we consider). These time points could represent any desired—but fixed—granularity (e.g., years, days, or seconds, or even transaction-based counters).

A *validity interval* is represented as a left-closed, right-open or right-closed, interval, which is bounded by two time points (e.g., $[1990, 2010)$), thus denoting a discrete and finite set of time points. This way, we are able to support both *range-queries* (i.e., *"Is this fact valid in the range of [1999, 2006)?"*) and *snapshot queries* (i.e., *"Is this fact valid at time point 2006?"*). A snapshot query can then simply be seen as a special-case range query by using an interval consisting of just a single time point (e.g., $[2006, 2006]$). Every interval has a corresponding *confidence value* associated with it, which denotes the probability of the fact being valid for the given interval. Multiple, non-overlapping intervals can be concatenated to form a *time histogram* H. Intervals in a time histogram do not necessarily have to be contiguous. A gap between two consecutive intervals is equivalent to an interval with a confidence value of 0.

3.3. Event and State Relations

For t-facts, we further distinguish between *event* and *state* relations. In an event relation, a fact is valid at *exactly one* time point. For example, *Nicolas_Sarkozy isIn Beijing* is valid on 25-Aug-2011 (or 2011, if the time point is available at a coarser granularity only). Actually, President Sarkozy visits China frequently. This results in multiple different facts of an event relation, each associated with different time points. By default, facts in an event relation are thus associated with a validity interval consisting of only one time point. For capturing uncertainty, however, validity intervals (and entire histograms) may cover more than one time point, as in the following example:

$$\mathsf{winsCupForClub}(\mathsf{Beckham}, \mathsf{Champions\_League})[1999, 2001):0.8$$

For simplicity, we assume a *uniform* distribution for the probability of a fact within the interval in this case. For example, for the interval $[1999, 2001)$, which covers 2 time points with a confidence of 0.8, each time point in the interval would have a probability of 0.4. The confidences of all intervals (and implicitly also the confidences of the corresponding time points) must form a proper probability distribution, i.e., the sum of all intervals' confidences may be at most 1.

For a state relation, a fact is valid at *every time point* of an interval. Hence, all time points in the interval are (implicitly) associated with the probability of the interval, as in the following example:

$$\mathsf{playsForClub}(\mathsf{Beckham}, \mathsf{Manchester\_United})[1992, 2003):0.3; [2003, 2007):0.4$$

Here, for the interval $[1992, 2003)$, which covers 12 time points with a confidence of 0.3, the fact is valid at each time point with probability of 0.3; and for the interval $[2003, 2007)$, the fact is valid at each time point with probability of 0.4. For facts in a state relation, the confidences of all intervals must form a proper probability distribution.

For both event and state relations, the sum p of confidences for the intervals in a histogram may be less than 1. In general, a fact is *invalid* for all time points outside the range of time points captured by the histogram with probability

$1 - p$. Moreover, different operations for slicing and coalescing intervals apply, depending on whether a fact belongs to either an event or a state relation.

The extraction of time periods for state facts is challenging, because there are typically only few occurrences of facts in input sentences with explicit time intervals. Ideally, there are sentences like "Maradona had a contract with FC Barcelona from July 1982 to June 1984". However, such explicit sentences are rare in both news and web sources. Instead, we can find cues that refer to the *begin*, *end*, or some time point *during* the desired interval. For example, news articles would often mention sentences such as "Maradona did not play well in the match against Arsenal London" with a publication date of 15-May-1983 (a time point which is supposed to be contained within the corresponding state fact's interval). Thus, we also focus on time points (*begin*, *during* and *end* events) for the extraction step and later aggregate these into intervals of state-oriented t-facts.

3.4. Temporal Knowledge Representation

A widely used formalism in knowledge representation is OWL. However, it is computationally expensive; instead we use a simpler RDF-style model. As in OWL and RDF, all objects are represented as entities in the YAGO data model. Two entities can stand in a relation, and we call this relational instance a fact. All facts are represented by unary or binary relations. Unary relations capture memberships in semantic types such as: `David_Beckham` *instanceOf* `Soccer_Player`. Binary relations like *isBornOn*, *isBornIn*, *isMarriedTo*, or *winsAward* hold between entities of specific types; an example is:

`David_Beckham` *winsAward* `UEFA_Club_Player_Of_The_Year`

This fairly simple model has proven to be very valuable for knowledge exploration and querying. However, facts actually have associated time information. For example, "David Beckham has won the UEFA Club Player of the Year" in "1999". Therefore, in T-YAGO, we introduce the concept of temporal facts. A temporal fact is a relation with an associated validity time. The fact may be valid at a time point or within a time interval. In the current YAGO ontology, temporal facts cannot be directly represented. Facts are limited to binary relations while temporal facts have more than two arguments.

To support temporal facts in a binary relation model, we decompose the n-ary fact into a primary fact and several associated facts. We assign a fact identifier to the primary fact, and then the associated facts are represented as the relation between the identifier and the remaining arguments. For example, for the ternary relation between `David_Beckham`, the `UEFA_Club_Player_Of_The_Year`, and 1999, we use the original time-agnostic fact as a primary fact with identifier #1. For temporal facts valid at a time point, we use the relation *on* to describe the validity time. Then we represent the temporal property of the primary fact as: #1 *on* 1999. For temporal facts that are valid during a time period, we use two relations to represent the interval: the relation *since* for the begin time point, and the relation *until* for the end time point. For example, the fact that Beckham played for Real Madrid from 2003 to 2007 is represented as:

#2 : `David_Beckham` $\mathtt{playsForClub}$ `Real_Madrid`

#2 *since* 2003

#2 *until* 2007

Sometimes it is impossible to extract accurate time points, say the exact day, and sometimes we may know only the begin or the end of a fact's validity interval but not both. For these situations, we introduce a data type for time points with the earliest and latest possible time to constrain the range of the true time point. For example, if we know that Beckham started playing for Real Madrid in July 2003 and terminated his contract in 2007, we would add the temporal facts:

#2 *since* $[1-\mathtt{July}-2003,\ 31-\mathtt{July}-2003]$

#2 *until* $[1-\mathtt{January}-2007,\ 31-\mathtt{December}-2007]$

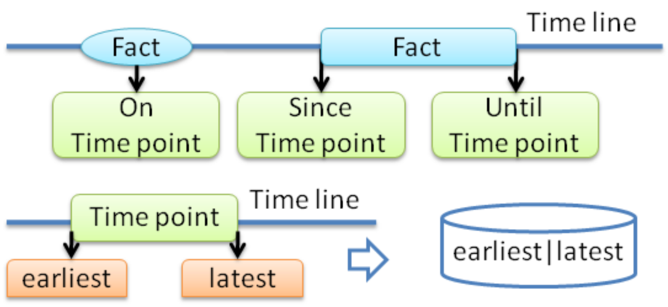


Figure 3.1.: Representation of temporal facts.

If we later learn that his contract with the team Los Angeles Galaxy started on July 1, 2007, and assuming a constraint that one must not play for two clubs at the same time, we could refine the *until* relation for #2 into:

#2 *until* $[1 - January - 2007,\ 30 - June - 2007]$
Figure 3.1 illustrates our representation of validity times of facts.

Chapter 4.

Temporal Knowledge Harvesting

The world is dynamic: periodic events like sports competitions need to be interpreted with their respective time points, and facts such as coaching a sports team, holding political or business positions, and even marriages do not hold forever and should be augmented by their respective timespans. This chapter addresses the problem of automatically harvesting temporal facts with such extended time-awareness. We describe how we gather temporal facts from semi-structured and free-text sources.

4.1. Problem Statement and Contributions

4.1.1. Motivation

Who were the teammates of the legendary Argentinean player Diego Maradona? How often has the pop singer Madonna been married, and to whom? Questions like these often arise from the information needs of Internet users, but are not supported by keyword-based search engines. Instead, answering them requires an understanding of entities and relationships embedded in Web contents, or even better, a knowledge base that contains facts about people, organizations, locations, etc. In the last few years, such knowledge bases have indeed been built and made publicly accessible, for example: *DBpedia* [3], *YAGO* [62], *TextRunner* [20], *ReadTheWeb* [12], the commercial knowledge portals *freebase.com* and *trueknowledge.com*, the computational knowledge engine *wolframalpha.com*, the entity search engine *entitycube.research.microsoft.com*, and others. Most of these have been automatically compiled, either by integrating structured sources (e.g., geospecies.org, geonames.org, world factbook, etc.), by harvesting semi-structured sources such as Wikipedia infoboxes, lists, and categories, or by information extraction from Web pages. Some of them involve a substantial amount of manual curation for quality assurance. Moreover, there is a strong momentum towards semantically inter-linking many knowledge sources into the *Linked Data* cloud [28], see *linkeddata.org*.

However, the world is highly dynamic and nothing lasts forever! Knowledge evolves over time, and many facts are fairly ephemeral, e.g., winners of sports competitions, and occasionally even CEOs and spouses. In addition, many information needs by advanced users require *temporal knowledge*. For example, consider the following variations of the above example questions: Who were the teammates of Diego Maradona during the 1990 FIFA World Cup? When did Madonna get married, when did she get divorced? None of these questions are supported by existing knowledge bases. The problem that we tackle in this thesis is to automatically distill, from news articles and biography-style texts such as Wikipedia, *temporal facts* for a given set of relations. By this we mean instances of the relations with additional time annotations that denote the validity point or span of a relational fact. For example, for the *winsAward* relation between people and awards, we want to augment facts with the time points of the respective events; and for the *worksForClub* relation between athletes and sports clubs, we would add the timespan during which the fact holds. This can be seen as a specific task of extracting ternary relations, which is much harder than the usual information extraction issues considered in prior work.

4.1.2. Contributions

Knowledge harvesting enables the automatic construction of large knowledge bases. In this chapter, we present the methods of harvesting temporal facts from semi-structured and free-text sources. Also we made a first attempt to harvest spatio-temporal knowledge from news archives to construct trajectories of individual entities for spatio-temporal entity tracking.

In summary, the work in this chapter makes the following contributions:

- handcrafted rules for extracting temporal facts from semi-structured textual sources (see Section 4.2),

- a new model for distilling temporal facts from textual sources like news or biographies (see Section 4.3),
- an algorithm for extended label propagation with consideration of inclusion constraints (see Section 4.3.3),
- an algorithm for distilling spatio-temporal knowledge from textual sources (see Section 4.4),
- experimental studies for knowledge harvesting from free-text sources, comparing the extraction of base facts (without time) against a state-of-the-art baseline [49], and demonstrating the effectiveness of harvesting temporal facts for a variety of relations from the domains of soccer and celebrities (see Section 4.3.4),
- the evaluation for the spatio-temporal knowledge harvesting on the 20 years' New York Times news article corpus shows that our methods are effective and scalable (see Section 4.4.3).

4.2. Temporal Knowledge Harvesting from Semi-structured Text

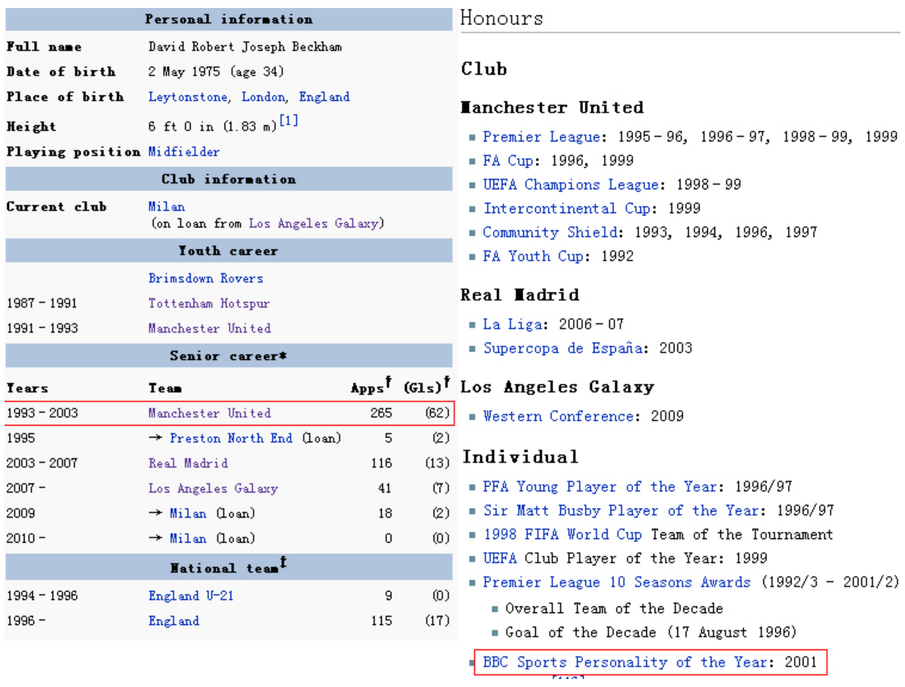
This section presents methods for constructing a temporal extension of the YAGO knowledge base [62]. Currently YAGO is built upon facts extracted from semi-structured infoboxes and category information in Wikipedia, and it is unified with the taxonomic class system of WordNet. Infoboxes are based on templates which are re-used for important types of entities such as countries, companies, sportsmen, politicians, etc. This allows YAGO to apply rule-based extraction for popular infobox attributes (the attributes in the "long tail" are typically skipped as they are highly diverse and noisy). Like YAGO, our approach to temporal fact extraction also focuses on the semi-structured elements in Wikipedia. However, as we need to detect also the validity time of each fact, our rules are more sophisticated than those of the original YAGO. We use regular expression matching for this purpose. For higher coverage, we also analyze additional elements in Wikipedia articles, most notably lists such as awards which contain many meaningful temporal facts. Here again, regular

Example Text	Regular Expression	Type
1993-2003	`[\d]{4}[-]{1}[\d]{4}`	Time Period
1000 BC	`(-?\d{1,10})[\W_&&[^-]]*+ (?:BC|B\.C\.|BCE|AC|A\.C\.)`	Time Period
2 May 1975	`(\d{1,2})[a-z]{0,2} [\W_&&[^-]]*+MONTH(\d\d) [\W_&&[^-]]*+(-?\d{1,10})`	Time Point
[[Manchester United F.C. |Manchester United]]	`\[\[([^\\|\]]+)(?:\\| ([^\]]+))?\]\]`	Entity

Table 4.1.: Examples of regular expressions for identifying temporal expressions and entities.

expressions tailored to specific types of lists yield high return. Table 4.1 shows some regular expressions for identifying temporal expressions and entities from the source code of Wikipedia articles. In addition, our rule-based techniques can also bootstrap learning-based methods applied on the natural-language text of the full articles. For the latter, the lack of manually labelled training data is often the bottleneck if not a showstopper. Our rich collection of temporal facts in T-YAGO can serve as training data and reduce the need for expensive hand-labeling.

For illustration, Figure 4.1 lists the infobox and the honors list of David Beckham. For example, "Senior career" attributes are extracted as *playsForSeniorClub* facts, yielding the facts "`David_Beckham` *playsForSeniorClub* `Manchester_United`" and "`David_Beckham` *playsForSeniorClub* `Real_Madrid`". YAGO can accept only one of these two fact candidates, as it enforces consistency constraints like a functional dependency from SoccerPlayer to SoccerClub (a player cannot simultaneously play for two clubs). The reason for this deficiency is that YAGO lacks temporal information. In T-YAGO, we can accept both facts if we can validate that they refer to disjoint time periods. We identify time points or intervals like 1993-2003 by pattern matching. This yields temporal facts like "#3 : `David_Beckham` *playsForSeniorClub* `Manchester_United`",



Personal information			
Full name	David Robert Joseph Beckham		
Date of birth	2 May 1975 (age 34)		
Place of birth	Leytonstone, London, England		
Height	6 ft 0 in (1.83 m)[1]		
Playing position	Midfielder		
Club information			
Current club	Milan (on loan from Los Angeles Galaxy)		
Youth career			
	Brimsdown Rovers		
1987 – 1991	Tottenham Hotspur		
1991 – 1993	Manchester United		
Senior career*			
Years	**Team**	**Apps†**	**(Gls)†**
1993 – 2003	Manchester United	265	(62)
1995	→ Preston North End (loan)	5	(2)
2003 – 2007	Real Madrid	116	(13)
2007 –	Los Angeles Galaxy	41	(7)
2009	→ Milan (loan)	18	(2)
2010 –	→ Milan (loan)	0	(0)
National team‡			
1994 – 1996	England U-21	9	(0)
1996 –	England	115	(17)

Honours

Club

Manchester United

- Premier League: 1995 – 96, 1996 – 97, 1998 – 99, 1999
- FA Cup: 1996, 1999
- UEFA Champions League: 1998 – 99
- Intercontinental Cup: 1999
- Community Shield: 1993, 1994, 1996, 1997
- FA Youth Cup: 1992

Real Madrid

- La Liga: 2006 – 07
- Supercopa de España: 2003

Los Angeles Galaxy

- Western Conference: 2009

Individual

- PFA Young Player of the Year: 1996/97
- Sir Matt Busby Player of the Year: 1996/97
- 1998 FIFA World Cup Team of the Tournament
- UEFA Club Player of the Year: 1999
- Premier League 10 Seasons Awards (1992/3 – 2001/2)
  - Overall Team of the Decade
  - Goal of the Decade (17 August 1996)
- BBC Sports Personality of the Year: 2001

Figure 4.1.: Temporal facts in infobox and honors list.

"#3 *since* 1993", and "#3 *until* 2003" (with simplified notation, leaving out the [earliest, latest] part for brevity). Other time-annotated attributes are extracted in a similar way. From the honors lists we also obtain valuable knowledge such as "#4 : `David_Beckham` *winsAward* `BBC_Sports_Personality_of_Year`" and "#4 *on* 2001".

The category systems for Wikipedia articles also contain temporal facts like those highlighted in Figure 4.2. We identify time points in category names and map them onto the timeline. The remaining part of the category name is treated as the entity. For example, from the first category in Figure 4.2 we extract the temporal fact "#5 : `David_Beckham` *playsAs* `FIFA_World_Cup_players`", "#5 *on* 1998".

T-YAGO contains around 200,000 temporal facts from the sports domain. Among them, about 70,000 facts have been extracted from Wikipedia categories and lists embedded in articles. For evaluation, we randomly sample 50 facts for each relation. The precision of t-facts in relations of *worksForTeam* (including *playsForYouthClub*, *playsForSeniorClub*, *playsForNationalTeam*, and *managesTeam*), *participatedIn*, *isBornIn* and *diedIn* is above 90%. The precision of t-facts in *winsAward* is around 80%, as the facts are from lists which are not clean and

Categories: 1998 FIFA World Cup players | 2002 FIFA World Cup players | 2006 FIFA World Cup players | 20th-century English people | 21st-century English people | A.C. Milan players | BBC Sports Personality of the Year winners | British expatriate sportspeople in the United States | ... | People from Leytonstone | Premier League players | Preston North End F.C. players | Real Madrid C.F. players | Serie A footballers | The Football League players | UEFA Euro 2000 players | UEFA Euro 2004 players | 1975 births | Living people

Figure 4.2.: Temporal facts from categories of David Beckham.

difficult to apply regular expressions.

4.3. Temporal Knowledge Harvesting from Natural Language Free-text

4.3.1. Framework and System Overview

This section gives an overview on the system architecture for temporal knowledge harvesting from free-text. We introduce our system called PRAVDA (label PRopagated fAct extraction on Very large DAta) for gathering fact candidates and distilling facts with their temporal extent based on a new form of *label propagation (LP)*. This is a family of graph-based semi-supervised learning methods, applied to (in our setting) a similarity graph of fact candidates and textual patterns. LP algorithms start with a small number of manually labelled seeds, correct facts in our case, and spread labels to neighbors based on a *graph regularized objective function* which we aim to minimize. The outcome is an assignment of labels to nodes which can be interpreted as a per-node probability distribution over labels. In our scenario, the labels denote relations to which the fact in a correspondingly labelled node belongs.

We adopt the specific algorithm of [66], coined MAD (Modified Adsorption), with an objective function that combines the quadratic loss between initial labels (from seeds) and estimated labels of vertices with a data-induced graph regularizer and an L2 regularizer. The graph regularizer is also known as the un-normalized graph Laplacian, which penalizes changes of labels between vertices that are close. We develop substantial extensions, and show how to judiciously construct a suitable graph structure and objective function. Notably, we consider inclusion constraints between different relation labels for the same node in the graph. For example, we may exploit that a relation like *joinsClub* (with time points) is a sub-relation of *worksForClub* (with timespans).

System Architecture. Figure 4.3 shows the architecture of our system for fact harvesting from Web sources. The system consists of five components: *candidate gathering*, *pattern analysis*, *graph construction*, *label propagation*, and *noise cleaning*. The components are invoked in five phases. This chapter focuses on the first four phases; the noise cleaning phase is presentend in Chapter 5.

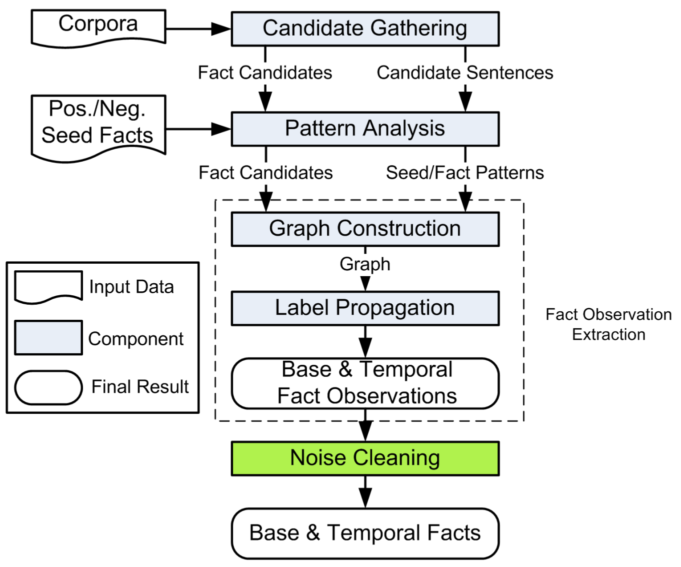


Figure 4.3.: System overview.

1. *Candidate Gathering*: This phase serves to collect potentially relevant sentences. A sentence is interesting if it contains entities in relevant types for a target relation of interest. We employ a test for the potential presence

of a base fact in a sentence, by checking for the existence of two noun phrases (denoting two entities) in the same sentence. In addition, we test for the existence of a temporal expression (currently only explicit date) in the sentence, thus producing raw input also for temporal fact candidates. This is discussed in Section 4.3.2.

2. *Pattern Analysis*: Based on a (small, manually crafted) set of seed facts for a particular relation (either base or temporal), seed patterns in the form of sets of word-level n-grams are extracted from the interesting sentences. For each target relation that we aim to extract, a statistical prior is then computed based on how strongly fact candidates have evidence in the form of seed patterns. This is discussed in Section 4.3.2.

3. *Graph Construction*: To facilitate the knowledge harvesting procedure, the fact candidates and pattern candidates are represented as vertices of a graph. Edges represent the evidence connection between a pattern and a fact candidate, as well as the similarity between two patterns. The graph construction is presented in Section 4.3.2.

4. *Label Propagation*: Finally, a graph-based semi-supervised learning method is employed to harvest both base facts and temporal facts. To this end, we have developed an extended label propagation algorithm which is presented in Section 4.3.3.

4.3.2. Patterns and Graph Model

In this section, we first describe our candidate gathering approach, raising surface (text) strings to entities. Then, we introduce our pattern analysis method, which is based on “basic”, “seed” and “fact” patterns. Basic patterns are (entity) type-aware word-level n-grams of surface strings. Based on statistical analytics, those patterns yielding in high quality results for a certain relation become seed patterns. Fact patterns are finally derived from sentences where the similarity of an observed basic pattern and seed patterns exceeds a pre-defined threshold. Finally, we will explain the underlying (pattern-candidate) graph model between patterns and candidate sentences. To facilitate the following discussions, we use bold capital letters to indicate matrices, and normal lower-case letters to indicate scalar values.

Candidate Gathering

Our approach is driven by target relations for which we would like to gather instances. The set of relations of interest is denoted as $\mathcal{R}$={R_1, R_2, ..., R_m}. Each is associated with a type signature and is either a base relation or a temporal relation. $Type(\mathcal{R})$ contains all possible entity types in the type signatures of the relations in $\mathcal{R}$.

We assume that there exists a knowledge base with typed entities which contains the individual entities whose types are in $Type(\mathcal{R})$. In this section, we consider relations of interest from the domains of soccer and celebrities, and use YAGO [62] as knowledge base. The YAGO knowledge base contains almost all the pertinent entities harvested from Wikipedia, such as persons (both sports people and celebrities), clubs, locations, awards, universities, and even a large fraction of people's spouses. YAGO also provides us with a fairly complete dictionary of surface names for entities, including abbreviations (e.g., "ManU" for the entity `Manchester_United`). Thus, detecting mentions of entities in an input text is straightforward.

We consider a textual corpus $\mathcal{C}$, e.g., a set of Wikipedia articles, news articles, or Web pages, as input for candidate gathering. The corpus is pre-processed to generate meaningful segments. In this section, we consider sentence-level segments: the corpus is decomposed into a set of sentences where each sentence is a sequence of tokens.

Given a sentence, an entity recognition and disambiguation module first checks whether at least two entities (and a temporal mention for temporal relations) are mentioned in the same sentence. This is primarily based on YAGO's "means" relation, where a string means an entity (e.g., "ManU" meaning `Manchester_United`). The entity recognition engine works in four stages:

1. Based on a sliding window technique "interesting" strings are extracted. A text string becomes interesting when it matches the subject of a YAGO fact with the means relation. That means, the string can be mapped to at least one YAGO entity.

2. Entities that can be mapped unambiguously (by plain string matching between the textual input and the means relation of YAGO) are identified.

These entities serve as "ground-truth entities" for the following disambiguation steps.

3. The "ground-truth entities" and Wikipedia anchor links are used to build a graph for disambiguation. The graph consists of edges between those (entity-) nodes where mutual anchor links among pages exist. For each "interesting string" it is now checked if an entity exists that is directly linked with a single "ground-truth entity". If more than two links to "ground-truth entities" exist, the analyzed entity becomes - due to its strong relation to this context a "ground-truth" entity, too.

4. Disambiguation of each "interesting" string to a single (highest ranked) entity based on a distance function applied to the before constructed graph.

We then check whether the entity types are compatible with the type signature of some target relation R_m. If so, a candidate fact is generated, and the sentence is added to the set of interesting sentences. For example, the sentence "`David_Beckham` played for `Real_Madrid` and `LA_Galaxy`" will create the candidate facts (`David_Beckham`, `Real_Madrid`) and (`David_Beckham`, `LA_Galaxy`), but <u>not</u> (`Real_Madrid`, `LA_Galaxy`) as there is no relation of interest between two entities of type *CLUB*.

Pattern Analysis

Basic Patterns. Our notion of patterns builds upon the prior work of [49], where sets of word-level n-grams from sentences are used as patterns. We extend this approach by exploiting the type signatures of the target relations for which we perform fact harvesting.

Consider a fact candidate (e_i, e_j) and its corresponding candidate sentence, written in the form x e_i y e_j z where x, y and z are surface string surrounding the entities. We consider the *context surface string* y as an (initial) indicator of the relationship with fact (e_i, e_j). For example, we observe a fact candidate (`David_Beckham`, `LA_Galaxy`) in the sentence s = "`David_Beckham` finally moved from Real_Madrid before his recent joining `LA_Galaxy` in 2007.", the context surface string "finally moved from Real_Madrid before his recent joining" is the evidence for the *joinsClubTemp* relationship between `David_Beckham` and `LA_Galaxy`.

Context surface strings are usually long and over specific. It is extremely rare to observe other entity pairs sharing exactly the same context surface strings. To overcome this problem, we generalize the pattern representation. One possible solution is to adopt word level n-grams to represent patterns. Before generating the n-grams, we first do compression and lifting on context surface strings.

In order to avoid getting misguided by problems induced from natural language processing, we convert each surface string into a *compressed surface string*. These contain only - preserving their original order - nouns, verbs, and prepositions after stop words removal based on Wikipedia's "List of English Stop Words" [1]. Nouns, verbs and prepositions are identified by LingPipe Part-of-Speech tagger [2]. All verbs are transformed into present tense based on the verb dictionary provided by Assert (Automatic Statistical SEmantic Role Tagger) [3]. Nouns are stemmed with PlingStemmer [4].

The compressed surface strings are further generalized by considering the set of types that constitute the type signatures of our target relations. This is done by replacing entities by their types (e.g. *PERSON*, *CITY*, *CLUB*, and *DATE*). Thus, the corresponding *lifted surface string* of sentence s is {"move from *CLUB* before join"}. For a lifted surface string, we generate word-level n-grams as its *basic pattern* (with n typically set to 2). For example, the basic pattern of s w.r.t. the entity pair (`David_Beckham`, `LA_Galaxy`) is denoted as $BP(s,$ `David_Beckham`, `LA_Galaxy`$)$ = ("move from", "from *CLUB*", "*CLUB* before", "before join").

Seed Patterns. Given a set of positive and negative samples for the facts in a specific relation, we are able to measure how good a pattern is for a particular relation. The best patterns are then regarded as *seed patterns* for this relation. The overall procedure of identifying seed patterns for a specific relation is similar to [49].

A positive seed fact (e_i, e_j) (or $(e_i, e_j)@t_k$) for a base relation R_b (or a temporal

[1]Stop Words, `http://en.wikipedia.org/wiki/Stop_words`

[2]LingPipe POS tagger, `http://alias-i.com/lingpipe/web/demo-pos.html`

[3]ASSERT, `http://cemantix.org/assert.html`

[4]PlingStemmer, `http://www.mpi-inf.mpg.de/yago-naga/javatools/`

relation R_t), is a valid fact of the relation R_b (or R_t). A negative seed fact (e_i, e_j) (or $(e_i, e_j)@t_k$) for R_b (or R_t) is an entity pair (with temporal mention) that is not a fact of R_b (or R_t). $PF(R_i)$ and $NF(R_i)$ denote the manually crafted positive and negative seed facts for R_i.

For each base relation R_b we identify the sentences that contain two entities from a positive seed in $PF(R_i)$ and the sentences that contain two entities from a negative seed in $NF(R_i)$. These sentences are collected into the sets $PS(R_b)$ and $NS(R_b)$, respectively. Analogously, for each temporal relation R_t, the positive sentences are those that contain both entities and the temporal expression (date) of a positive seed, and the negative sentences are those that contain a negative seed. The sets of positive and negative sentences are denoted as $PS(R_t)$ and $NS(R_t)$, respectively.

Given a relation $R_i \in \mathcal{R}$, a candidate sentence s with lifted surface string $L(s)$ containing a basic pattern p, support and confidence are defined as follows:

$$supp(p, R_i) = |\{s \in PS(R_i) | p \subseteq L(s)\}|$$

$$conf(p, R_i) = \frac{|\{s \in PS(R_i) | p \subseteq L(s)\}|}{|\{s \in PS(R_i) \cup NS(R_i) | p \subseteq L(s)\}|}$$

The support value captures the frequency of a pattern in conjunction with positive seed facts, whereas the confidence value denotes the ratio of a pattern co-occurring with positive seed facts versus negative seed facts. If $supp(p, R_i)$ is above predefined thresholds, the basic pattern p is considered as a *seed pattern* of R_i. $SP(R_i)$ denotes the set of all seed patterns for R_i. In our experiment, we vary the threshold of support in different settings.

Fact Patterns. Given a candidate sentence $s = x\ e_i\ y\ e_j\ z$, the *fact pattern* of the sentence w.r.t. the entity pair (e_i, e_j) is defined as:

$$(TYPE_1, TYPE_2, BP(s, e_i, e_j)) \tag{4.1}$$

$TYPE_1$ and $TYPE_2$ are the types of e_i and e_j, respectively. $BP(s, e_i, e_j)$ is a basic pattern of sentence s w.r.t. the entity pair (e_i, e_j). In addition, a fact pattern is associated with a weight vector that contains the weights of the fact patterns with regard to the target relations of interest: $(w(BP(s, e_i, e_j), R_1), w(BP(s, e_i, e_j), R_2), \ldots, w(BP(s, e_i, e_j), R_m))$.

Given a relation R_k, if its type signature is the same as the fact pattern's type signature, the weight is defined by Equation (4.2):

$$\begin{aligned} weight(BP(s, e_i, e_j), R_k) = & \max_{p \in SP(R_k)} (sim(BP(s, e_i, e_j), p) \\ & \times \quad conf(p, R_k)) \end{aligned} \tag{4.2}$$

where the similarity between the basic pattern $BP(s)$ and a pattern p in the seed patterns of R_i is defined based on the Jaccard coefficient with regard to n-grams q:

$$sim(BP(s, e_i, e_j), p) = \frac{|\{q \in BP(s, e_i, e_j) \cap q \in p\}|}{|\{q \in BP(s, e_i, e_j) \cup q \in p\}|}$$

Graph Model

A weighted undirected graph $G(V, E, \mathbf{W})$ is employed to facilitate the knowledge harvesting, where V is the set of vertices, E is the set of edges, and $\mathbf{W}$ is the weight matrix (i.e., an entry $\mathbf{W}_{ij}$ indicating the similarity of the corresponding vertices v_i and v_j).

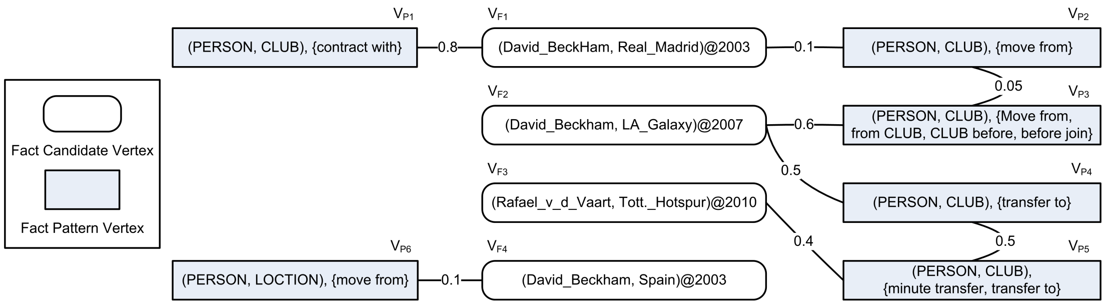
We divide the set of vertices into two subsets: *fact candidate vertices* (V_F) and *fact pattern vertices* (V_P), where $V_F \cap V_P = \emptyset$ and $V = V_F \cup V_P$. A vertex in V_F corresponds to a fact candidate, i.e., either a base fact (e_i, e_j) or a temporal fact (e_i, e_j)@t_k. A vertex in V_P corresponds to a fact pattern in the form of $(TYPE_1, TYPE_2, p)$, as defined in formula (4.1), where p indicates a basic pattern.

Edges between a fact candidate vertex and a fact pattern vertex: An edge between a fact candidate vertex v_f and a fact pattern vertex v_p is generated, if there is a candidate sentence containing both. Intuitively, more candidate sentences provide higher support for the fact candidate v_f held in the relation v_p. Let $S(v_f, v_p)$ be the set of candidate sentences, then the weight of the edge (v_f, v_p) is defined and normalized as $1 - e^{(-1)*\beta*|S(v_f, v_p)|}$, where $\beta \in (0, 1]$. The parameter β smoothens similarity values by reducing the impact of outliers with undesirable high cardinality such as spam.

Edges between two fact pattern vertices: If two fact patterns are similar, their corresponding vertices are connected by an edge with a similarity-based weight. The intuition is that similar patterns often indicate the same relation. Two fact pattern vertices v_{p_i} and v_{p_j} are considered dissimilar (similarity is zero) if their corresponding type signatures are inconsistent (i.e., $v_{p_i}.TYPE_1 \neq v_{p_j}.TYPE_1$ or v.v.). If their type signatures are consistent, the edgeweight is defined by the similarity of the basic patterns $v_{p_i}.\mathrm{p}$ and $v_{p_j}.\mathrm{p}$ as follows:

1. If the two patterns share the same verb and preposition, the similarity between them is the distance-weighted Jaccard similarity. If the two patterns differ in their verbs or prepositions, then the similarity is set to zero.
2. If one pattern contains a verb and the other one does not, the similarity between them is zero.
3. If neither of them contains a verb, the similarity is the distance-weighted Jaccard similarity.

The distance-weight of an n-gram in a basic pattern is defined based on its position in the sentence. The closer the n-gram appears to the target entity, the higher the weight. This way, we can effectively deal with relatively long patterns. For instance in the example graph in Figure 4.4, although the two patterns V_{P2} and V_{P3} share the n-gram "move from", its weight in V_{P3} is very low, thus resulting in a very low similarity between the two patterns.



Fact Candidate Vertex
Fact Pattern Vertex
V_{P1}
(PERSON, CLUB), {contract with}
0.8
V_{F1}
(David_BeckHam, Real_Madrid)@2003
0.1
V_{P2}
(PERSON, CLUB), {move from}
0.05
V_{F2}
(David_Beckham, LA_Galaxy)@2007
0.6
V_{P3}
(PERSON, CLUB), {Move from, from CLUB, CLUB before, before join}
0.5
V_{F3}
(Rafael_v_d_Vaart, Tott._Hotspur)@2010
V_{P4}
(PERSON, CLUB), {transfer to}
0.4
0.5
V_{P6}
(PERSON, LOCTION), {move from}
0.1
V_{F4}
(David_Beckham, Spain)@2003
V_{P5}
(PERSON, CLUB), {minute transfer, transfer to}

Figure 4.4.: An example graph.

4.3.3. Label Propagation Algorithm

After constructing the graph of fact candidates and fact patterns, a semi-supervised label propagation algorithm is applied to extract the relations that hold between fact candidates. The idea of label propagation is that the labeling of vertices (here the possible relations) is propagated to nearby vertices via weighted edges until convergence. In this process, the vertices linked by highly weighted edges are more likely to have the same labels, because the weights of edges represent the vertex similarities. All vertices are treated uniformly, regardless of whether they represent fact candidates or fact patterns.

Suppose the graph contains n vertices, consisting of fc fact candidate and fp fact pattern vertices, where $|V| = n$, $|V_F| = fc$, $|V_P| = fp$, and $n = fc + fp$. In addition, there are $m + 1$ labels. The first m labels correspond to the m relations in the set $\mathcal{R}$ of target relations, plus a "none" label (denoting no or an unknown relation). In the following, we will use labels and relations interchangeably.

The matrix $\mathbf{Y} \in \mathbb{R}_+^{n\times(m+1)}$ denotes the graph's initial label assignment. The vector $\mathbf{Y}_{i*}$ denotes the i^{th} row of matrix $\mathbf{Y}$, with each element $\mathbf{Y}_{ik} \in [0, 1]$ indicating vertex v_i's confidence score for label k, where $k \in \{1, 2, ..., m + 1\}$. The ℓ^{th} column of matrix $\mathbf{Y}$ is denoted by vector $\mathbf{Y}_{*\ell}$.

When a fact candidate vertex v_i corresponds to a positive seed fact of relation R_k, the vertex is labelled as R_k with confidence 1, i.e., $\mathbf{Y}_{ik} = 1$. A fact pattern vertex v_j is labelled as R_k with confidence 1 (namely $\mathbf{Y}_{jk} = 1$), if its weight for a seed pattern of relation R_k (evaluated by Equation 4.2) is greater than a pre-specified threshold value γ. Note that if v_j has a strong weight (greater than γ) with the seed patterns of more than one relation, all the corresponding relations are labelled. All other entries in $\mathbf{Y}$ are initially set to 0.

When the label propagation process finishes, each vertex is associated with a vector indicating the estimated labels. We use matrix $\widehat{\mathbf{Y}} \in \mathbb{R}_+^{n\times(m+1)}$ to indicate the estimated labels for all vertices. Note that the matrix $\widehat{\mathbf{Y}}$ has exactly the same ordering of vertices and labels as $\mathbf{Y}$. Finally, for each vertex, the labels with sufficiently high confidence scores are selected.

Basic Objective Function

As shown in [66], the process of label propagation is realized by minimizing the following objective function:

$$\mathcal{L}(\widehat{\mathbf{Y}}) = \sum_{\ell} \left[(\mathbf{Y}_{*\ell} - \widehat{\mathbf{Y}}_{*\ell})^T \mathbf{S}^{\ell} (\mathbf{Y}_{*\ell} - \widehat{\mathbf{Y}}_{*\ell}) + \mu_1 \widehat{\mathbf{Y}}_{*\ell}^T \mathbf{L} \widehat{\mathbf{Y}}_{*\ell} + \mu_2 \| \widehat{\mathbf{Y}}_{*\ell} - \mathbf{R}_{*\ell} \|^2 \right] \tag{4.3}$$

where $\mathbf{Y}_{*\ell}$ and $\widehat{\mathbf{Y}}_{*\ell}$ denote the ℓ^{th} column vector of the initial assignment matrix $\mathbf{Y}$ and estimated label matrix $\widehat{\mathbf{Y}}$, respectively.

The intuition of keeping estimated labels centred around initial labels is realized by the first term of the objective function (4.3), which is the quadratic loss that measures the discrepancies between the initial labels and the estimated labels. In the original MAD algorithm of [66], the diagonal matrix $\mathbf{S}^{\ell}$ is identical for every label, with p_i^{inj} on the main diagonal for labelled vertices and 0 for unlabelled ones, where p_i^{inj} is set to be proportional to vertex v_i's degree. Since the initially labelled fact pattern vertices may be noisy, the corresponding elements on the main diagonal of $\mathbf{S}^{\ell}$ should also be able to control the confidences on the label. Assuming that we are considering label ℓ and v_i is a fact pattern vertex, we represent the confidences on the label ℓ by multiplying p_i^{inj} with the vertex v_i's fact-pattern weight on label ℓ (evaluated by Equation 4.2).

The label spreading effect is achieved by the unnormalized graph Laplacian matrix $\mathbf{L}$, which smoothes label similarities between linked vertices based on edge weights. This property becomes clearer when $\mu_1 \widehat{\mathbf{Y}}_{*\ell}^T \mathbf{L} \widehat{\mathbf{Y}}_{*\ell}$ is rewritten as

$$\mu_1 \widehat{\mathbf{Y}}_{*\ell}^T \mathbf{L} \widehat{\mathbf{Y}}_{*\ell} = \mu_1 \frac{1}{2} \sum_{i,j} \mathbf{Z}_{ij} (\widehat{\mathbf{Y}}_{i\ell} - \widehat{\mathbf{Y}}_{j\ell})^2 \tag{4.4}$$

where $\mathbf{Z}_{ij} = \mathbf{W}_{ij} \times p_i^{cont} + \mathbf{W}_{ji} \times p_j^{cont}$ and $\mathbf{W}_{ij}$ is the edge weight between vertices v_i and v_j. Following [66], we set p_i^{cont} to be inversely proportional to vertex v_i's degree. The hyper-parameter μ_1 controls the influence of the graph Laplacian. In order to minimize the function, similarities of labels between vertex pairs linked with high edge-weight are enforced, while different labels are tolerated for those vertex pairs with low edge-weights.

The last term in the objective function (4.3) can be regarded as a combination of an L2 regularizer for the m relations of interest and a loss function for the "none" label. $\mathbf{R}_{*\ell}$ is the ℓth column of the abandon matrix $\mathbf{R} \in \mathbb{R}_+^{n \times (m+1)}$. Specifically, $\mathbf{R}_{*\ell}$ is a zero-valued column vector for every $\ell \in \{1, 2, ..., m\}$, which works exactly as an L2 regularizer for the estimated labels of the m relations of interest to avoid over-fitting to their seed labels. However, the vector of the last label $(m+1)$ is $\mathbf{R}_{*(m+1)} = (p_1^{abnd}, p_2^{abnd}, ... \ p_n^{abnd})^T$, where p_i^{abnd} is set to 1-p_i^{inj}-p_i^{cont} like in the original MAD algorithm [66]. The hyper-parameter μ_2 adjusts the degree of regularization and the trust in the "none" relation.

Since we only change the diagonal matrices $\mathbf{S}_\ell$, the objective function can still be solved by the optimization algorithm of MAD [66].

Incorporating Inclusion Constraints

Sometimes, a relation R_i implies another relation R_j, meaning that a valid fact of R_i should also be a valid fact of R_j. In such a case, there is an inclusion constraint (subsumption) that R_i implies R_j. For example, the relation *isCapitalOf* between cities and countries implies the *isCityOf* relation. Inclusion dependencies arise in specific forms for temporal relations, that is, between events and their implied lasting relations, and also between base relations and their temporal counterparts. For example, the event relation *joinsClubTemp* implies *worksForClubTemp* and the base relation *worksForClub*. The temporal fact `joinsClubTemp(David_Beckham, Real_Madrid)@2003` also implies the temporal fact `worksForClubTemp(David_Beckham, Real_Madrid)@2003`, which in turn implies the base fact `worksForClub(David_Beckham, Real_Madrid)`.

Inclusion constraints provide an additional asset: they can generate (or reinforce) the evidence for a fact by observing another fact about a different relation. Thus, if relation ℓ implies relation ℓ', it suggests a fact candidate v_m having label ℓ, that the estimated confidence value $\widehat{\mathbf{Y}}_{m\ell'}$ should be greater or equal to the estimated confidence value $\widehat{\mathbf{Y}}_{m\ell}$. For instance, if a fact candidate v_m is estimated as *joinsClubTemp*, its estimated labels should satisfy the following constraint $\widehat{\mathbf{Y}}_{m(\mathrm{worksForClubTemp})} \geq \widehat{\mathbf{Y}}_{m(\mathrm{joinsClubTemp})}$. In the following, we will show how to incorporate inclusion constraints into an extended label propagation framework.

Objective Function with Inclusion Constraints. We first introduce some notation. The matrix $\mathbf{C} \in \mathbb{R}_+^{m\times m}$ records the inclusion dependencies between different relations of interest. Specifically, if relation ℓ implies relation ℓ', we set $\mathbf{C}_{\ell'\ell} = 1$. Given a label ℓ, $\mathrm{imply}(\ell)$ denotes the labels that imply ℓ and

$\mathtt{implied}(\ell)$ those labels that are implied by ℓ.

$$\begin{aligned}\mathcal{L}(\widehat{\mathbf{Y}}) = \sum_{\ell} \Big[&(\mathbf{Y}_{*\ell} - \widehat{\mathbf{Y}}_{*\ell})^{\mathsf{T}} \mathbf{S}_{\ell} (\mathbf{Y}_{*\ell} - \widehat{\mathbf{Y}}_{*\ell}) + \mu_1 \widehat{\mathbf{Y}}_{*\ell}^{\mathsf{T}} \mathbf{L} \widehat{\mathbf{Y}}_{*\ell} \\ &+ \mu_2 \|\widehat{\mathbf{Y}}_{*\ell} - \mathbf{R}_{*\ell}\|^2 + \mu_3 \sum_{\nu} \sum_{k \neq \ell} \mathbf{C}_{\ell k} \mathbf{Y}_{\nu k} (\widehat{\mathbf{Y}}_{\nu \ell} - \widehat{\mathbf{Y}}_{\nu k})^2 \Big] \\ &\text{s.t. } \widehat{\mathbf{Y}}_{\nu \ell'} \geq \widehat{\mathbf{Y}}_{\nu \ell}, \;\; \nu \in V, \; \ell' \in \mathtt{implied}(\ell) \end{aligned} \tag{4.5}$$

We introduce a new objective function, described in Equation (4.5), to incorporate inclusion constraints between different relations. The last term in Equation (4.5) serves to smoothen the estimated labels to satisfy the inclusion constraints. Note that only vertices containing initial labels are smoothed. The loss is only added when ℓ is labelled, and then propagated from ℓ to the labels in higher levels. Some examples of inclusion constraints are: *joinsClubTemp* implies *playsForClubTemp* implies *worksForClubTemp* implies *worksForTemp*. If *playsForClubTemp* is labelled, the loss will be added to (*playsForClubTemp* and *worksForClubTemp*), and (*worksForClubTemp* and *worksForTemp*). The loss should not be added between the two labels *joinsClubTemp* and *playsForClubTemp*. In contrast to the MADDL algorithm [66], which assumes all correlations among classes to be symmetric, our approach does not assume symmetry of correlations. For example, suppose that *joinsClubTemp* implies *worksForClubTemp* and the fact candidate `(David_Beckham, Real_Madrid)@2003` holds for *joinsClubTemp* with high probability (e.g. 0.9), but has low probability for *worksForClubTemp* relation (e.g. 0.1). This is undesirable and should be smoothed, as *joinsClubTemp* implies *worksForClubTemp*. Therefore, the objective function (4.5) will pull the estimated value for *worksForClubTemp* closer to the value for *joinsClubTemp* by reducing the labels' differences. On the other hand, if the same fact candidate holds for *worksForClubTemp* with high probability (e.g., 0.9), but has low probability for the *joinsClubTemp* (e.g., 0.1), the objective function (4.5) will not pull the estimated value for *joinsClubTemp*.

Optimization Problem Solution. In the Appendix, we prove that adding inclusion constraints does not change the *convexity* of the objective function in Equation (4.3). The convex optimization problem is solved by *cyclic coordinate descent* [5]. The algorithm iterates through each coordinate of $\widehat{\mathbf{Y}}$, and solves sequentially the following sub-problems until it converges:

$$\min_{\widehat{\mathbf{Y}}_{v\ell}} \mathcal{L}(\widehat{\mathbf{Y}}) \text{ s.t. } \mathtt{lb} \leq \widehat{\mathbf{Y}}_{v\ell} \leq \mathtt{ub} \tag{4.6}$$

where $\mathtt{lb}$ and $\mathtt{ub}$ are the lower and upper bounds of $\widehat{\mathbf{Y}}_{v\ell}$, respectively.

In each sub-problem, all variables of $\widehat{\mathbf{Y}}_{v\ell}$ except the one at the current coordinate are fixed. So this technique optimizes the objective function by solving a sequence of single-variable optimization problems. Because the problem is a weighted sum of quadratic functions, there is a closed-form solution for each sub-problem. For coordinate $\widehat{\mathbf{Y}}_{v\ell}$, we set the first partial derivative with respect to $\widehat{\mathbf{Y}}_{v\ell}$ to zero. Thus, the optimum in the absence of constraints is:

$$\widehat{\mathbf{Y}}_{v\ell} = \frac{\mathbf{S}_{\ell vv}\mathbf{Y}_{v\ell} + \mu_1 \sum_i \mathbf{Z}_{vi}\widehat{\mathbf{Y}}_{i\ell} + \mu_2 \mathbf{R}_{v\ell} + \mu_3 \sum_{k\neq\ell}(\mathbf{C}_{\ell k} + \mathbf{C}_{k\ell})\mathbf{Y}_{vk}\widehat{\mathbf{Y}}_{vk}}{\mathbf{S}_{\ell vv} + \mu_1 \sum_i \mathbf{Z}_{vi} + \mu_2 + \mu_3 \sum_{k\neq\ell}(\mathbf{C}_{\ell k} + \mathbf{C}_{k\ell})\mathbf{Y}_{vk}} \tag{4.7}$$

From the inclusion constraints we can infer a lower ($\mathtt{lb}$) and an upper bound ($\mathtt{ub}$) for the solution. Specifically, if $\mathtt{imply}(\ell)$ is not empty, $\mathtt{lb} = \max_{\ell' \in \mathtt{imply}(\ell)} \widehat{\mathbf{Y}}_{v\ell'}$, otherwise $\mathtt{lb} = 0$; if $\mathtt{implied}(\ell)$ is not empty, $\mathtt{ub} = \min_{\ell' \in \mathtt{implied}(\ell)} \widehat{\mathbf{Y}}_{v\ell'}$, otherwise $\mathtt{ub} = 1$. For example, the lower bound of *worksForClubTemp* is the maximum of *joinsClubTemp* and *leavesClubTemp*, with the upper bound 1. Similarly, the upper bound of *joinsClubTemp*/*leavesClubTemp* is the value on *worksForClubTemp*, with the lower bound 0. The final optimum is computed by $\max(\min(\widehat{\mathbf{Y}}_{v\ell}, \mathtt{ub}), \mathtt{lb})$ to ensure that the optimal solution satisfies the inclusion constraints.

We refer to this algorithm as *ICMAD* (Inclusion-Constraints-aware MAD). The complete method is shown in Algorithm 1. Compared to the original MAD method, the lower and upper bounds are checked after each label update.

Combining Base and Temporal Graphs

The above *ICMAD* algorithm can be applied to either base or temporal facts and their patterns. So far, however, these would be two separate instantiations of the same algorithm. In this subsection, we discuss how to combine the two settings for enhanced output.

Algorithm 1 ICMAD Algorithm.

```
Input: Graph: $G = (V, E)$,
       Matrices $\mathbf{S}^{\ell}$ for each $\ell \in [1...m+1]$,
       Matrix $\mathbf{Z}$ derived from the graph Laplacian,
       Abandon matrix $\mathbf{R}$,
       Initial labels matrix $\mathbf{Y}$,
       Inclusion constraint weight matrix $\mathbf{C}$
Output: Estimated labels matrix $\widehat{\mathbf{Y}}$
 1: $\widehat{\mathbf{Y}} \leftarrow \mathbf{Y}$;
 2: repeat
 3:   for each $v \in V$ do
 4:     for each $\ell \in [1, ...m+1]$ do
 5:       / / Evaluation based on Equation (4.7);
 6:       $numerator \leftarrow {\mathbf{S}^{\ell}}_{vv}\mathbf{Y}_{v\ell} + \mu_1 \sum_i \mathbf{Z}_{vi}\widehat{\mathbf{Y}}_{i\ell} + \mu_2\mathbf{R}_{v\ell} + \mu_3 \sum_{k\neq\ell}(\mathbf{C}_{\ell k} + \mathbf{C}_{k\ell})\mathbf{Y}_{vk}\widehat{\mathbf{Y}}_{vk}$;
 7:       $denominator \leftarrow {\mathbf{S}^{\ell}}_{vv} + \mu_1 \sum_i \mathbf{Z}_{vi} + \mu_2 + \mu_3 \sum_{k\neq\ell}(\mathbf{C}_{\ell k} + \mathbf{C}_{k\ell})\mathbf{Y}_{vk}$;
 8:       $\widehat{\mathbf{Y}}_{v\ell} \leftarrow \frac{numerator}{denominator}$;
 9:       for each $k \in [1, ...m] \wedge k \neq \ell$ do
10:         / / Lower and upper bounds checking;
11:         if $\mathbf{C}_{\ell k} > 0 \wedge \widehat{\mathbf{Y}}_{v\ell} < \mathbf{Y}_{vk}$ then $\widehat{\mathbf{Y}}_{v\ell} \leftarrow \mathbf{Y}_{vk}$;
12:         if $\mathbf{C}_{k\ell} > 0 \wedge \widehat{\mathbf{Y}}_{v\ell} > \mathbf{Y}_{vk}$ then $\widehat{\mathbf{Y}}_{v\ell} \leftarrow \mathbf{Y}_{vk}$;
13:       end for
14:     end for
15:   end for
16: until converged
```

For base relations, seed patterns may be noisy, especially when positive seeds (entity pairs) hold for multiple relations. For example, a person may be born and may die in the same place, so both relations *isBornIn* and *hasDiedIn* have similar seed patterns, which is undesired. However, the corresponding temporal relations are better distinguishable, as - usually - people do not die on the same date (or same year) they are born. Thus, by combining base and temporal graph, the seed patterns from the temporal graph can help to counter ("correct") noisy patterns in the base graph.

The pattern nodes in the temporal graph are actually a subset of those in the base graph. Thus, we can unify both graphs by sharing all pattern nodes and conceptually connecting the fact candidate nodes from base and temporal graph with their respective patterns. A new objective function is described in Equation (4.8), where matrix $\mathbf{G} \in \mathbb{R}_{+}^{p_b \times p_t}$ records the relationship of the same pattern nodes in base and temporal graph. p_b and p_t are the number of pattern nodes in base and temporal graph. If a pattern node v_i in the base graph is exactly the same as the pattern node v_j in the temporal graph, then we assign $\mathbf{G}_{ij}$ to 1. Otherwise, $\mathbf{G}_{ij}$ is set to 0.

Suppose $\mathbf{Y} = \begin{pmatrix} \mathbf{Y_b} & 0 \\ 0 & \mathbf{Y_t} \end{pmatrix}$, where $\mathbf{Y_b}$ and $\mathbf{Y_t}$ indicate the initial label matrices of base and temporal graph, respectively. Likewise, $\widehat{\mathbf{Y}} = \begin{pmatrix} \widehat{\mathbf{Y_b}} & 0 \\ 0 & \widehat{\mathbf{Y_t}} \end{pmatrix}$, where $\widehat{\mathbf{Y_b}}$ and $\widehat{\mathbf{Y_t}}$ indicate the estimated label matrices of base and temporal graph, respectively. $\mathbf{S}^{\ell}$ and $\mathbf{L}$ can also be re-written analogously. On this basis, we define Equation (4.8) as the new objective function as follows:

$$\begin{aligned} \mathcal{L}(\widehat{\mathbf{Y}}) = \sum_{\ell} \Big[& (\mathbf{Y}_{*\ell} - \widehat{\mathbf{Y}}_{*\ell})^{T} \mathbf{S}^{\ell} (\mathbf{Y}_{*\ell} - \widehat{\mathbf{Y}}_{*\ell}) + \mu_1 \widehat{\mathbf{Y}}_{*\ell}^{T} \mathbf{L} \widehat{\mathbf{Y}}_{*\ell} + \mu_2 \|\widehat{\mathbf{Y}}_{*\ell} - \mathbf{R}_{*\ell}\|^2 \\ & + \mu_3 \sum_{v} \sum_{k \neq \ell} \mathbf{C}_{\ell k} \mathbf{Y}_{vk} (\widehat{\mathbf{Y}}_{v\ell} - \widehat{\mathbf{Y}}_{vk})^2 + \mu_4 \sum_{v,u} \mathbf{G}_{vu} (\widehat{\mathbf{Y}_{\mathbf{b}}}_{v\ell} - \widehat{\mathbf{Y}}_{\mathbf{t} u \ell''})^2 \Big] \\ & \text{s.t. } \widehat{\mathbf{Y}}_{v\ell'} \geq \widehat{\mathbf{Y}}_{v\ell}, \;\; v \in V, \; \ell' \in \mathtt{implied}(\ell) \end{aligned} \tag{4.8}$$

where label ℓ'' is the corresponding temporal relation of base label ℓ (e.g., ℓ for *worksForClub* with ℓ'' for *worksForClubTemp*).

The new objective function shown in Equation (4.8) keeps the same form as Equation (4.5). Therefore, it is still solvable by cyclic coordinate descent.

4.3.4. Experiments

System Implementation

The whole system is implemented in Java. For pattern analysis, PostgreSQL [5] database serves as the back-end database server. OpenNLP [6] is exploited to get the part of speech tag, convert verbs to present tense and stem nouns from textual articles. For MAD algorithm, we use Junto [7]–The Label Propagation Toolkit.

[5] http://www.postgresql.org/

[6] http://opennlp.apache.org/

[7] http://code.google.com/p/junto/

Base Relations	**Temporal Relations**	**Type Signatures**
isBornIn	*isBornInTemp*	(*PERSON*, *CITY*)
hasDiedIn	*hasDiedInTemp*	(*PERSON*, *CITY*)
worksForClub	*worksForClubTemp*	(*PERSON*, *CLUB*)
	joinsClubTemp	(*PERSON*, *CLUB*)
	leavesClubTemp	(*PERSON*, *CLUB*)
isMarriedTo	*isMarriedToTemp*	(*PERSON*, *PERSON*)
	getsMarriedWithTemp	(*PERSON*, *PERSON*)
	getsDivorcedFromTemp	(*PERSON*, *PERSON*)
	winsAwardTemp	(*PERSON*, *AWARD*)

Table 4.2.: Relations of interest.

Experimental Setup

Data Sets. Our methods have been evaluated against two corpora from the soccer and celebrity domain. In the soccer domain, we selected more than 23,000 soccer players' Wikipedia articles. In addition, we retrieved around 110,000 on-line news articles (e.g., BBC, Yahoo! news, ESPN, etc.) by searching for players contained in the "FIFA 100 list" [8]. Likewise, the celebrity corpus has been populated with more than 88,000 news articles of persons mentioned in the "Forbes 100 list" [9] in addition to their Wikipedia articles.

Relations of Interest. In the current experiments, we are interested in four base relations and nine temporal relations. The relations and their type signatures are summarized in Table 4.2. It is worth mentioning that temporal relations (highlighted in grey) such as *isBornInTemp* or *hasDiedInTemp* are "extensions" of their corresponding base relations *isBornIn* resp. *hasDiedIn* by temporal information for the identified event (e.g. birth, death, etc.).

[8] http://en.wikipedia.org/wiki/FIFA_100

[9] http://www.forbes.com/lists/2010/53/celeb-100-10_The-Celebrity-100.html

Seed Selection. For each relation positive and negative seed facts have been manually selected. Facts with prominent entities are chosen as seed facts due to their frequency. Thus, the k most frequently co-occurring entity pairs are chosen as seed facts for each relation type signature.

For example, when selecting seed facts for the *worksForClub* relation, we compute the co-occurrence statistics (eventually supported by keyword filtering) of *PERSON* and *CLUB* entities based on the same sentence. Starting with the highest ranked entity pair, we manually check in Wikipedia if the relation of interest is satisfied. If approved, it becomes a positive seed fact, otherwise a negative.

Performance Metrics. For base facts, the baseline for our experiments is the state-of-the-art PROSPERA system [49]. We evaluate both approaches in terms of precision correctness of the extracted facts. For temporal facts, due to the size of the corpus, evaluation of recall is impossible. Precision estimates are based on sampling (50 facts per relation) with a human evaluation against ground truth information in Wikipedia.

Sample Output. For the relations of interest we generate patterns and (temporal) facts. Table 4.3 depicts randomly chosen results from both corpora. Results for temporal facts are marked in grey.

Results on Base Fact Observation Extraction

We first compare b-fact observation extraction with PROSPERA and PRAVDA. The support threshold in pattern analysis is $supp(p, R_i) = 10$ with 100 positive and 10 negative seed facts for each relation.

Table 4.4 summarizes the results of this experiment. As one can see both approaches achieve comparable precision. However, PRAVDA outperforms PROSPERA with respect to the number of extracted facts. The reason for this gain can be explained with LP's propagation of the labels' confidence scores from one node to another. This results in confidence propagation among very similar, but less frequently occurring patterns ("be born in" and "born in"). Of course, this comes with the risk of inducing more noise, which becomes apparent for more complex relations such as *isMarriedTo*. The explanation for

Relation	**Pattern**	**Fact**
joinsClubTemp	join *LEAGUE* side	(David_Beckham, Los_Angeles_Galaxy)@2007
leavesClubTemp	transfer from	(Thierry_Henry, Juventus_F.C.)@1999
getsMarriedWithTemp	marry	(Demi_Moore, Ashton_Kutcher)@2005
getsDivorcedFromTemp	divorce	(Jennifer_Aniston, Brad_Pitt)@2005
worksForClub	to join at	(Michael_Owen, Real_Madrid_C.F.)
isMarriedTo	husband	(Hillary_Rodham_Clinton, Bill_Clinton)

Table 4.3.: Patterns and facts extracted for relations of interest.

	PRAVDA		**PROSPERA**	
Relation	**# Observations**	**Precision**	**# Observations**	**Precision**
worksForClub	12,724	88%	4,536	82%
isBornIn	3,629	88%	2,426	90%
hasDiedIn	183	90%	55	93%
isMarriedTo	428	74%	147	90%

Table 4.4.: Base fact observation extraction (100 positive, 10 negative seeds).

this "misbehavior" is simple: a prominent couple will be frequently mentioned in the media in completely different contexts (e.g. when dating, talking, visiting, etc.), thus, propagating high confidence to less useful labels, too.

In order to quantify the impact of the number of seed facts, we now evaluate the performance of two base relations (*worksForClub* and *isMarriedTo*) for varying numbers of positive seed facts while keeping the number of negative seeds fixed with 10. We vary them from 10, 20, 50 to 100 by simultaneously adopting the support $supp(p, R_i)$ used in pattern analysis to 4, 6, 8 and 10 respectively.

		PRAVDA		**PROSPERA**	
	# Positive Seeds	**# Observations**	**Precision**	**# Observations**	**Precision**
worksForClub	10	4,882	84%	2,202	86%
	20	8,381	82%	3,358	86%
	50	10,826	82%	4,321	84%
	100	12,724	88%	4,536	82%
isMarriedTo	10	388	74%	16	0%
	20	391	64%	123	88%
	50	391	78%	159	82%
	100	428	74%	147	90%

Table 4.5.: Impact of varying the number of positive seed facts.

Table 4.5 summarizes the results on the impact of varying the number of positive seed facts. PRAVDA extracts far more facts than PROSPERA, particularly if the number of positive seeds is low. Both approaches achieve similar precision quality throughout the experiments (except for PROSPERA for the *isMarriedTo* relation with 10 positive seeds given). It can again be recognized that PRAVDA is slightly prone to generate more noise, particularly when the relation of interest is more complex. As an example of the *isMarriedTo* relation, consider the following text from our corpus:

> “The Scottish News of the World has published photographs which it claims show how Sir Paul McCartney is secretly dating Hollywood actress Rosanna Arquette - the virtual double of his estranged wife Heather Mills.”

Since we do not use a dependency parser for efficiency reasons, there is no evidence that the two entities `Rosanna Arquette` and `Heather Mills` are not directly connected. Consequently, the surface string pattern “wife” links `Rosanna Arquette` with `Heather Mills` and identifies them (incorrectly) as a couple.

Results on Temporal Fact Observation Extraction with Inclusion Constraints

In the following we present the results of our methods on t-fact observation extraction with inclusion constraints (yearly granularity). Since PROSPERA is not geared for harvesting temporal facts, we only report the performance of our method. We evaluate inclusion constraints for two relations:

- *joinsClubTemp* and *leavesClubTemp* imply *worksForClubTemp*
- *getsMarriedWithTemp* and *getsDivorcedFromTemp* imply *isMarriedToTemp*

The results are depicted in Table 4.6 (10 positive and negative seed facts for the first seven relations) with a special highlighting in dark grey of the relations *worksForClubTemp* and *isMarriedToTemp*. Without giving any seed facts to the *worksForClubTemp* and *isMarriedToTemp* relations, inclusion constraint are successfully applied to extract facts for the two relations.

We see that results with respect to the number of extracted facts and precision are not so good as for b-fact observation extraction. This is not surprising for at least three reasons. First, our extraction currently operates on the sentence level, meaning that entity pair and temporal expression have to be in the very same sentence. Second, we are currently able to detect only explicit temporal expressions. Third, associating temporal expressions to the appropriate relation and/or entity/ies is sometimes complex for human, too.

Extraction of facts for the *leavesClubTemp* and *joinsClubTemp* relations are particularly difficult. Usually media coverage starts a long time before the actual transfer actually occurs. Consequently, this induces a lot of noise to the *leavesClubTemp* relation. In addition, both before mentioned relations are - of course - affected by unsupported rumors, which are common to sport news coverage.

Relation	**# P/N Seeds**	**# Observations**	**Precision**
isBornInTemp	10/10	3,471	82%
hasDiedInTemp	10/10	65	88%
winsAwardTemp	10/10	243	84%
joinsClubTemp	10/10	1,871	80%
leavesClubTemp	10/10	235	71%
getsMarriedWithTemp	10/10	76	78%
getsDivorcedFromTemp	10/10	13	71%
worksForClubTemp	0/0	2,086	86%
isMarriedToTemp	0/0	88	80%

Table 4.6.: Temporal fact observation extraction with inclusion constraints.

Results on Joint Base and Temporal Fact Observation Extraction

In the following we now investigate the results of joint base and temporal fact observation extraction. In contrast to the experiments before, we now combine the base and temporal graphs and extract the facts simultaneously. Table 4.7 compares the joint approach with the separate (for base facts resp. temporal facts) ones. For the sake of clarity, results of t-fact observation extraction have been highlighted in grey. Experiments have been conducted with 10 positive/negative seed facts per base relation and 5 positive/negative seed facts per temporal relation (except *worksForClubTemp* and *isMarriedToTemp* in order to evaluate the impact of inclusion constraints).

As we can see from Table 4.7, particularly b-fact observation extraction benefits from the joint approach, while t-fact observation extraction remains more or less unchanged. The change in precision is not statistically significant. It is worth mentioning that the *worksForClub* relation increases a lot in terms of

	Joint		Separate	
Relation	**# Observations**	**Precision**	**# Observations**	**Precision**
worksForClub	10,467	84%	4,882	84%
isBornIn	3,139	92%	368	96%
worksForClubTemp	2,761	84%	2,032	88%
isBornInTemp	3,184	84%	3,297	85%

Table 4.7.: Comparison of joint and separate fact observation extraction.

number of extracted fact observations. Even more, the number of extracted facts as well as the precision for the *worksForClub* relation is higher than its separate "counterpart", which is the 20-seed baseline compared with 10 seeds for *worksForClub* + 5 seeds for *joinsClubTemp* + 5 seeds for *leavesClubTemp* in the joint approach.

	W/ Constraints		W/O Constraints	
Relation	**# Observations**	**Precision**	**# Observations**	**Precision**
worksForClub	10,467	84%	5,657	90%
isMarriedTo	413	64%	480	64%
worksForClubTemp	2,761	84%	0	NA
isMarriedToTemp	98	82%	0	NA
joinsClubTemp	1,788	82%	1,840	82%
leavesClubTemp	249	64%	273	66%
getsMarriedWithTemp	73	77%	73	77%
getsDivorcedFromTemp	13	78%	13	73%

Table 4.8.: Joint extraction with and without constraints.

Finally, we study the impact of inclusion constraints on the joint approach. From the results in Table 4.8 we can see that inclusion constraints play an important role in the joint extraction approach. For relation *worksForClub* benefits from the inclusion, and the change in precision is not statistically significant. Accuracy for temporal relations (colored in grey) remains almost unchanged. Please note that no fact observations have been discovered for *worksForClubTemp* and *isMarriedToTemp* (indicated in dark grey in Table 4.8) as they have been extracted via inclusion constraints, only.

4.3.5. Conclusions

This section introduced a unified framework for harvesting base facts and temporal facts from textual Web sources. Our experimental results with news articles and Wikipedia pages demonstrate the viability and high accuracy of our approach. Moreover, for the base-facts case, we have shown much higher recall over one of the best state-of-the-art baselines, while achieving similar precision. We believe that extended forms of constrained label propagation, as developed in this thesis, are a very promising alternative to prior methods like Max-Sat reasoning or sampling over Markov-Logic models or factor graphs.

Our extended LP method is nicely geared for scale-out on distributed platforms. So far, however, our experiments were at a scale where we did not need to utilize this property. Our future work aims at experiments with Web-scale datasets (e.g., the TREC ClueWeb corpus), and will explore the scalability behavior of our method.

4.4. Scalable Spatio-temporal Knowledge Harvesting

4.4.1. Motivation

Who attended the 2008 G-20 Washington Summit? Which countries did George W. Bush visit during 2006? How many times did President Bush meet with Prime Minister Brown in 2007?

Such questions can be easily answered if the traveling trajectories of people are known. We propose and develop a scalable and effective framework for harvesting the spatio-temporal knowledge from news archives. A spatio-temporal fact holds the "*person* `isIn` *location* `at` *time*" relationship, where the *person* entity indicates a particular person, the *location* entity indicates a spatial location, and the *time* entity indicates a temporal period. For example, given the sentence "When President George H. W. Bush visited Mainz in 1989, he made a landmark appeal for a special relationship between the United States and Germany.", a spatio-temporal fact "George H. W. Bush `isIn` Mainz `at` 1989" can be extracted. The pertinent spatio-temporal facts of individual *persons* constitute their trajectories. Based on these trajectories, more complicated analytic jobs can be conducted. For example, it is interesting to explore the relationship of *who* meets *whom*, *where*, and *when*. Furthermore, these trajectories can be used for identifying important events by capturing several politicians' trajectories intersected in the same place at a particular time.

This work distinguishes itself from existing works by considering both spatial and temporal annotations associated with the harvested facts. In the NewsStand [70] project, the spatial focus is identified and associated with a cluster of news articles. Our work associates both spatial and temporal annotations to specific entities. The evaluation on the 20 years' New York Times news article corpus showed that our methods are effective and scalable.

4.4.2. Methodology

Our spatio-temporal knowledge harvesting framework is composed of an entity extraction and disambiguation module as well as a fact generation module. In this section, we describe the methodologies that are applied in each module and how the needed trajectories are produced.

Entity extraction and disambiguation. Considering efficiency, we did the entity extraction and disambiguation in a simple but effective way. We aim to identify *person*, *location* and *time* entities from text. *Person* and *location* entities are identified by a named entity recognizer, and *time* is recognized by regular expressions. An identified *person* or *location* may be ambiguous and may correspond to more than one entity. For example, "President Bush" may refer to "George H. W. Bush" or "George W. Bush"; and "Paris" may refer to the capital of France or to Paris in Texas, U.S., etc. We propose a multi-stage disambiguation process.
Person disambiguation: (1) Entities having only first or last name are disambiguated with the use of the full name entities. Suppose that a full name,

of the form "*firstname lastname*", appears in an article or in the metadata of an article. For the same article, an entity annotated either with "*firstname*" or "*lastname*" should be disambiguated as the full name entity. If there are more than one proper full names, we take the closest one. (2) If the aforementioned procedure does not provide us with the needed entities for disambiguation, we simply use the most popular entity at the article's publication time. For example, for an article published in 1991, "President Bush" is disambiguated as "George H. W. Bush", since 1991 is in George H. W. Bush's president term, as it can be found in T-YAGO. Therefore, it is more probable that "President Bush" refers to "George H. W. Bush" than it refers to "George W. Bush".

Location disambiguation: (1) Group the identified *location* entities by text locality. (a) If these *location* entities satisfy a containment relationship, this relationship can be used for *location* disambiguation. For example, if "Paris in Texas, U.S. " appears in an article, "Paris" can be disambiguated as the one in Texas, U.S.. (b) If these *location* entities do not imply possible containment relationships with each other, we use the most frequent candidate containing country of these *location*s as their containing county. For example, suppose that "Paris" and "Rouen" appear in the same paragraph. A city with the name "Paris" exists in France and in the U.S. Moreover, "Rouen" is a city of France and Barbados. Since the most frequent candidate containing country of the two cities is France, "Paris" and "Rouen" can be disambiguated as "Paris, France" and "Rouen, France" respectively. (2) The location name may change because of changes in country boundaries and regimes. For example, the city Saint Petersburg in Russia was called Leningrad in USSR. The obsolete location names and their corresponding current location names can be obtained by a specific dictionary from GeoNames [10]. Based on the publication time of an article, such location names can be disambiguated to the same *location* entity.

Fact generation. Spatio-temporal facts, in the form of (*person*, *location*, *time*), can be generated using all possible combinations of the disambiguated entities. To increase precision, we only consider the entities which appear in a pre-defined window, e.g. within a sentence. Additionally, for further fact cleaning, we have defined two pruning rules. Sometimes *location* may appear

[10]GeoNames, `http://www.geonames.org/`

in noun phrases, meaning that it does not indicate a real spatial location. (1) If *location* appears before a *person*, we expect that some prepositions, e.g. “in” or “at”, exist before the *location*. (2) If *location* appears after a *person*, we expect to find at least a verb between them. For instance, consider the sentence “Two New York advisors sent a signal to Senator Barack Obama, Democrat of Illinois in 2004.” Since there is no “in” or “at” before New York, and no verbs between Barack Obama and Illinois, all generated facts will be pruned.

4.4.3. Experiments

System implementation. In our implementation, we used MapReduce, a framework for large-scale data-intensive computations, to execute our algorithm and to generate the trajectories from the extracted facts. The news archives are distributed and stored in different nodes. During the map phase, each article is processed by the entity extraction and disambiguation module and the fact generation module. The output of the map phase is a set of spatio-temporal facts. Articles located at different nodes, can be processed in parallel. In the reduce phase, the facts are grouped by *person* first, and sorted according to the *time. This way, the trajectory records of each individual person are generated.*

Experimental setup. We evaluate our methods on the New York Times Annotated Corpus [11], which contains more than 1.8 million articles published between 1987 and 2007. The raw data size of the textual content is about 8 GB. All methods in this section were implemented in Java using Sun JDK 1.6. LingPipe [12] is the named entity recognizer that we chose and Hadoop 0.21.0 is the distributed computing infrastructure that we used. All the experiments are run on a local cluster. Each node of the cluster has two Intel Xeon 2.40GHz, 4-core CPUs.

Visualization. Since two strict pruning rules are employed in fact generation module, only 79321 facts in total are extracted from the whole corpus, which indicates that the overall recall is not high. However, at the same time, the

[11]New York Times Annotated Corpus, `http://corpus.nytimes.com/`

[12]LingPipe, `http://alias-i.com/lingpipe/index.html`

pruning rules make the survived facts have relatively high precision. We visualize part of the trajectory of George W. Bush on a map, as shown in Figure 4.5. The number of each marker represents the location visit order.

Scalability. We configure different data and cluster size to test the scalability of our algorithm. We run our algorithm on each configuration twice, and report the average execution time. Note that we focus on harvesting the spatio-temporal knowledge from the whole corpus rather than querying the individual trajectories. The results, as shown in Figure 4.6, indicate that the larger the data size is, the more performance benefit the bigger cluster can gain, which indicate that our method is scalable.

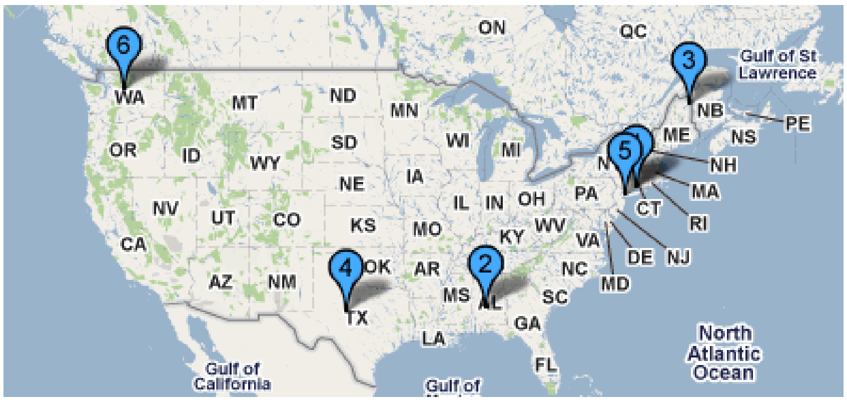


Figure 4.5.: Visualization.

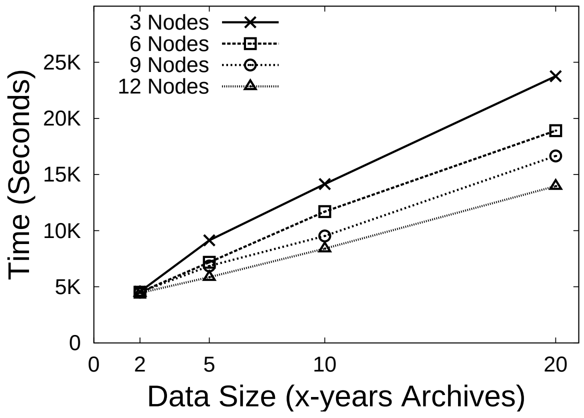


Figure 4.6.: Scalability.

Chapter 5.

Temporal Knowledge Cleaning

Time information is ubiquitous on the Web, and considering temporal constraints among facts extracted from the Web is a key for high-precision query answering over time-variant factual data. This chapter introduces how to clean noisy and inconsistent temporal fact observations, and reconcile these diverse observations into clean temporal facts.

5.1. Problem Statement and Contributions

A major shortcoming that modern knowledge bases [3, 86, 62, 89, 13] still face is the lack of time information in both the representation model and the types of queries they support. Thus we perform temporal knowledge harvesting to populate temporal knowledge bases, as introduced in the previous chapter.

Achieving 100% precision in temporal fact extraction is an elusive goal. In addition to errors from the imperfect extraction methodology, noise also comes from data sources. Mistakes made by the writers of social media are the key factors for the inconsistency among different sources. Furthermore, temporal expressions could appear in different granularities: yearly, monthly, or daily.

We propose an Integer Linear Programming (ILP) based approach to clean the previously extracted noisy temporal fact observations; this is presented in Section 5.2. A histogram-based model is designed to reconcile the diverse (potentially inconsistent) temporal observations of a specific temporal fact into a concise histogram across time. New temporal facts are then derived via the histogram model by Datalog-like reasoning. The histogram-based method is presented in Section 5.3.

5.2. Cleaning Temporal Fact Observations

5.2.1. Problem Statement and Contributions

Motivation. Extracting temporal facts is a complex and time-consuming endeavor. Apart from the efforts required for entity recognition and disambiguation, the identification of the relationships between entities is delicate. There are "conservative" strategies that aim at high precision, but they tend to suffer from low recall. On the other hand, there are "aggressive" approaches that target at high recall, but frequently suffer from low precision. To this end, we introduce a method that allows us to gain maximum benefit from both "worlds" by "aggressively" gathering fact candidates and subsequently "cleaning out" the incorrect ones.

Contributions. The salient properties of our approach and the novel contributions of this section are the following:

- an ILP solver incorporating constraints on temporal relations among events (e.g., marriage of a person must be non-overlapping in time) (see Section 5.2.3),
- experiments on real world news and Wikipedia articles showing that we gain recall while keeping up with precision (see Section 5.2.4).

5.2.2. Framework

Facts and Observations. We aim to extract factual knowledge transient over time from free-text. More specifically, we assume *time* $\mathcal{T} = [0, \mathrm{T}_{max}]$ to be a finite sequence of time points. Furthermore, a *fact* consists of a relation with two typed arguments and a time-interval defining its validity. For instance, we write `worksForClub(David_Beckham, Real_Madrid)@[2003,2008)` to express that Beckham played for Real Madrid from 2003 to 2007. Since sentences containing a fact and its full time-interval are sparse, we consider three kinds of textual observations for each relation, namely *begin*, *during*, and *end*. "Beckham signed for Real Madrid from Manchester United in 2003." includes both the *begin* observation of Beckham being with Real Madrid as well as the *end* observation of working for Manchester. A *positive seed fact* is a valid fact of a relation, while a *negative seed fact* is incorrect. For example, for relation `worksForClub`, a *positive seed fact* is `worksForClub(David_Beckham, Real_Madrid)`, while `worksForClub(David_Beckham, Bayern_Munich)` is a *negative seed fact*.

Framework. As depicted in Figure 5.1, our framework is composed of five stages, where the first collects fact candidates with their corresponding sentences, the second mines patterns from the candidate sentences, the third builds graph for label propagation of the fourth stage to extract fact observations and the last removes noisy fact observations by enforcing constraints. Except *noise cleaning*, all the other stages are presented in Section 4.3. Thus in this chapter, we focus on presenting how to clean noisy temporal fact observations.

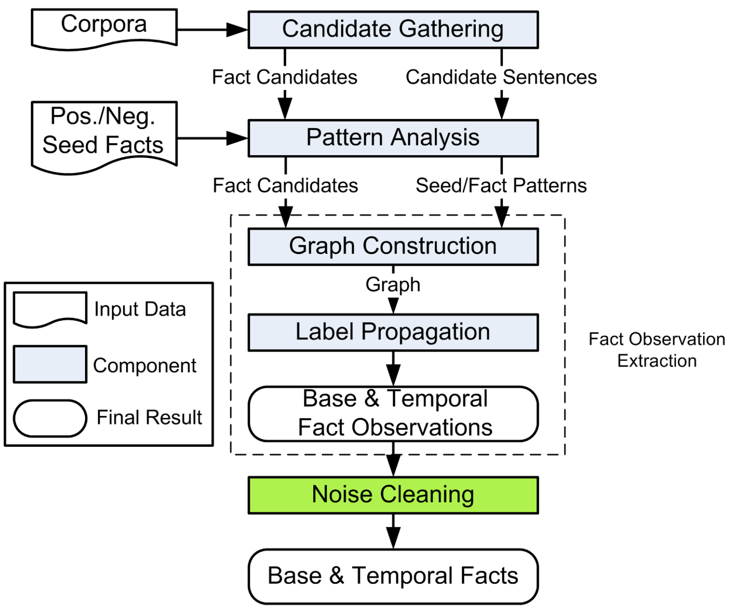


Figure 5.1.: System overview.

5.2.3. Applying Temporal Constraints

To prune noisy t-observations, we compute a consistent subset of t-observations with respect to temporal constraints (e.g. joining a sports club takes place before leaving a sports club) by an Integer Linear Program (ILP).

Variables. We introduce a variable $x_{f,r} \in \{0,1\}$ for each t-observation $r \in \{\mathcal{R}_{begin}, \mathcal{R}_{during}, \mathcal{R}_{end}\}$ of a t-fact $f \in \mathcal{F}$, where 1 means the observation is valid. Two variables $f^B, f^E \in [0, K]$ denote begin (B) and end (E) of time-interval of a t-fact $f \in \mathcal{F}$. Note, that many t-observations refer to the same t-fact f, since they share their entity pairs. $t^B_{x_{f,r}}$ and $t^E_{x_{f,r}}$ denote the possible earliest begin and latest end time point of t-observation $x_{f,r}$.

Objective Function. The objective function intends to maximize the number of t-observations, where $w_{x_{f,r}}$ is a weight obtained from the previous stage:

$$\max \sum_{f\in\mathcal{F}, r\in\{\mathcal{R}_{begin},\mathcal{R}_{during},\mathcal{R}_{end}\}} w_{x_{f,r}} \cdot x_{f,r}$$

Intra-Fact Constraints. f^B and f^E encode a proper time-interval by adding the constraint:

$$\forall f \in \mathcal{F} \quad f^B < f^E$$

Considering only a single relation $\mathcal{R}$, we assume the sets $\mathcal{R}_{begin}$, $\mathcal{R}_{during}$, and $\mathcal{R}_{end}$ to comprise its t-observations with respect to the *begin*, *during*, and *end* observations. Then, we introduce the constraints

$$\forall r \in \mathcal{R}_{begin} \quad t^B_{x_{f,r}} \cdot x_{f,r} \le f^B \le t^B_{x_{f,r}} \cdot x_{f,r} + (1 - x_{f,r})K \tag{5.1}$$

$$\forall r \in \mathcal{R}_{end} \quad t^E_{x_{f,r}} \cdot x_{f,r} \le f^E \le t^E_{x_{f,r}} \cdot x_{f,r} + (1 - x_{f,r})K \tag{5.2}$$

$$\forall r \in \mathcal{R}_{during} \quad f^B \le t^B_{x_{f,r}} \cdot x_{f,r} + (1 - x_{f,r})K \tag{5.3}$$

$$\forall r \in \mathcal{R}_{during} \quad t^E_{x_{f,r}} \cdot x_{f,r} \le f^E \tag{5.4}$$

Here we want to infer the *begin* and *end* time point of f: f^B and f^E, by given the t-observations $x_{f,r}$ with their weights $w_{x_{f,r}}$ and time intervals $[t^B_{x_{f,r}}, t^E_{x_{f,r}}]$. f has the same entity pair as $x_{f,r}$ and $t^B_{x_{f,r}}$, $t^E_{x_{f,r}}$ are begin and end of $x_{f,r}$'s time-interval.

Whenever $x_{f,r}$ is set to 1 for *begin* or *end* t-observations, Eq. (5.1) and Eq. (5.2) set the value of f^B or f^E of t-fact f to $t^B_{x_{f,begin}}$ or $t^E_{x_{f,end}}$, respectively. For each *during* t-observation with $x_{f,r} = 1$, Eq. (5.3) and Eq. (5.4) enforce $f^B \le t^B_{x_{f,begin}}$ and $t^E_{x_{f,end}} \le f^E$.

Inter-Fact Constraints. Since we can refer to a fact f's time interval by f^B and f^E and the connectives of Boolean Logic can be encoded in ILPs [36], we can use all temporal constraints expressible by Allen's Interval Algebra [1] to specify inter-fact constraints. For example, we leverage this by prohibiting marriages of a single person from overlapping in time.

Previous Work. In comparison to [68], our ILP encoding is time-scale invariant. That is, for the same data, if the granularity of $\mathcal{T}$ is changed from months to seconds, for example, the size of the ILP is not affected. Furthermore, because we allow all relations of Allen's Interval Algebra, we support a richer class of temporal constraints.

5.2.4. Experiments

System Implementation. The whole system is implemented in Java. For temporal knowledge harvesting, the system is implemented in the same way with the system presented in Section 4.3.4. The integer linear program tackling the constraints utilizes Gurobi[1].

Corpus. Experiments are conducted in the soccer and the celebrity domain by considering the *worksForClub* and *isMarriedTo* relation, respectively. In the soccer domain, we selected more than 23,000 soccer players' Wikipedia articles. Moveover for each person in the "FIFA 100 list" and "Forbes 100 list" we retrieve their Wikipedia article. In addition, we obtained about 450,000 documents for the soccer and celebrity domain from BBC, The Telegraph, Times Online and ESPN by querying Google's News Archive Search[2] in the time window from 1990-2011. All hyperparameters are tuned on a separate data set.

Seeds. For each relation we manually select the 10 positive and negative fact candidates with highest occurrence frequencies in the corpus as seeds.

Evaluation. We evaluate *precision* by randomly sampling 50 (*isMarriedTo*) and 100 (*worksForClub*) facts for each observation type and manually evaluating

[1]`http://www.gurobi.com/`

[2]`news.google.com/archivesearch`

them against the text documents. All experimental data is available for download from our website[3].

Pipeline vs. Joint Model

Setting. In this experiment we compare the performance of the pipeline being stages 3, 4 and 5 in Figure 5.1 and a joint model in form of an ILP solving the t-observation extraction and noise cleaning at the same time. Hence, the joint model resembles [55] extended by Section 5.2.3's temporal constraints.

Relation	Observation	Label Propagation		ILP for T-Fact Extraction		
		Precision	# Observations	Precision	# Observations	
worksForClub	*begin*	80%	2537	81%	2426	*Without Noise Cleaning*
	during	78%	2826	86%	1153	
	end	65%	440	50%	550	
isMarriedTo	*begin*	52%	195	28%	232	
	during	76%	92	6%	466	
	end	62%	50	2%	551	
worksForClub	*begin*	85%	2469	87%	2076	*With Noise Cleaning*
	during	85%	2761	79%	1434	
	end	74%	403	72%	275	
isMarriedTo	*begin*	64%	177	74%	67	
	during	79%	89	88%	61	
	end	70%	47	71%	28	

Table 5.1.: Pipeline vs. joint model.

Results. Table 5.1 shows the results on the pipeline model (lower-left), joint model (lower-right), label-propagation w/o noise cleaning (upper-left), and ILP for t-fact extraction w/o noise cleaning (upper-right).

[3]`www.mpi-inf.mpg.de/yago-naga/pravda/`

Analysis. Regarding the upper part of Table 5.1 the pattern-based extraction works very well for *worksForClub*, however it fails on *isMarriedTo*. The reason is, that the types of *worksForClub* distinguish the patterns well from other relations. In contrast, *isMarriedTo*'s patterns interfere with other person-person relations making constraints a decisive asset. That is the reason why the *ILP for t-fact extraction* wins on recall but fails on precision than *Label Propagation* for *isMarriedTo*. The ILP accepts most of the patterns co-occur with the seed facts, so all the fact candidates co-occurred with such patterns are accepted. However, those patterns are very noisy. For example, `David Beckham` can *kiss, date, visit* and *marry* `Victoria Beckham` in 1999. If `isMarriedTo(David_Beckham, Victoria_Beckham)@1999` is a seed fact, then patterns "kiss, date, visit and marry" are considered as good patterns for *isMarriedTo*. The noisy patterns cause a lot of false positives.

When comparing the joint model and the pipeline model, the former sacrifices recall in order to keep up with the latter's precision level. That is because the joint model's ILP decides with binary variables on which patterns to accept. For example "wedding" is a good pattern for *isMarriedTo*, but it may co-occur with many noisy fact candidates due to the extraction mistakes. However the ILP removes all the fact candidates with pattern "wedding". In contrast, label propagation addresses the inherent uncertainty by providing label assignments with confidence numbers. The ILP constraint solver in the pipeline model just removes the conflicted noisy fact observations from label propagation.

Increasing Recall

Setting. In a second experiment, we move the t-fact observation extraction stage away from high precision towards higher recall, where the successive noise cleaning stage attempts to restore the precision level.

Results. The columns of Table 5.2 show results for different values of μ_1 of Eq. (4.3) in Section 4.3.3. From left to right, we used $\mu_1 = e^{-1}, 0.6, 0.8$ for *worksForClub* and $\mu_1 = e^{-2}, e^{-1}, 0.6$ for *isMarriedTo*. The table's upper part reports on the output of *fact observation extraction*, whereas the lower part covers the facts returned by *noise cleaning*.

		Conservative		Standard		Relaxed		
		Precision	**# Observations**	**Precision**	**# Observations**	**Precision**	**# Observations**	
worksForClub	*begin*	84%	2443	80%	2537	80%	2608	*Without Noise Cleaning*
	during	81%	2523	78%	2826	76%	2928	
	end	76%	377	65%	440	62%	501	
isMarriedTo	*begin*	72%	112	52%	195	44%	269	
	during	90%	63	76%	92	52%	187	
	end	67%	37	62%	50	36%	116	
worksForClub	*begin*	83%	2389	85%	2469	84%	2536	*With Noise Cleaning*
	during	88%	2474	85%	2761	75%	2861	
	end	78%	349	74%	403	70%	463	
isMarriedTo	*begin*	72%	111	64%	177	46%	239	
	during	90%	62	79%	89	54%	177	
	end	69%	36	70%	47	38%	110	

Table 5.2.: Increasing recall.

Analysis. For the conservative setting label propagation produces high precision facts with only few inconsistencies, so the noise cleaning stage has no effect, i.e. no pruning takes place. This is the setting usual pattern-based approaches without cleaning stage are working in. In contrast, for the standard setting (coinciding with Table 5.1's left column) t-observation extraction yields less precision, but higher recall. Since there are more inconsistencies in this setup, the noise cleaning stage accomplishes precision gains compensating for the losses in the previous stage. In the relaxed setting precision drops too low, so the noise cleaning stage is unable to figure out the truly correct facts. In general, the effects on *worksForClub* are weaker, since in this relation the constraints are less influential.

5.2.5. Conclusions

In this section we have developed a method that combines label propagation with constraint reasoning for temporal fact extraction. Our experiments have shown that best results can be achieved by applying "aggressive" label propagation with a subsequent ILP for "clean-up". By coupling both approaches we achieve both high(er) precision and high(er) recall. Thus, our method efficiently extracts high quality temporal facts at large scale.

5.3. Temporal Fact Reasoning

5.3.1. Problem Statement and Contributions

Motivation. For reasoning and query answering temporal knowledge bases, new temporal facts need to be derived from existing temporal facts. Knowing, for example, the facts that a player *joined* and *left* a club, we could derive a time interval for when this player actually *played* for the club. Furthermore, teammates of the player and their corresponding time intervals could be derived as well, which calls for a principled approach to reasoning in temporal knowledge bases with uncertainty. Similar to work done on temporal databases [33], validity intervals provide simple, yet effective, support for query semantics built on interval intersections and unions in T-YAGO. Simple interval operations are however only of limited use for query processing (or reasoning) with *uncertainty*, i.e., with probabilistic models or otherwise statistically quantified degrees of uncertainty. In this section, we adopt the common possible-worlds semantics known from probabilistic databases and extend it towards histogram-like confidence distributions that capture the validity of facts across time. Query processing is done via a Datalog-like, rule-based inference engine, which employs the *lineage* of derived facts for confidence computations to remain consistent with the possible-worlds model.

Contributions. We propose an approach for representing and reconciling facts with temporal annotations for time-aware reasoning over uncertain and potentially inconsistent temporal knowledge bases. We briefly summarize the main contributions of this section as follows:

- an algorithm to reconcile multiple (potentially inconsistent) observations of facts with temporal annotations into a concise histogram at extraction time (see Section 5.3.2),

- the data lineage in the form of Boolean formulas that capture the logical dependencies between base and derived facts, in a recursive, Datalog-like reasoning environment (see Section 5.3.3),
- the evaluation on a real-world temporal knowledge (Timely YAGO) with more than 270,000 (aggregated) temporal facts, using handcrafted rules for query processing and reasoning in the football domain (see Section 5.3.4).

5.3.2. Histogram Aggregation

In our temporal model for extraction, each fact is associated with its possible earliest and latest time information. For example, from the sentence "Beckham signed up for Real Madrid in 2003.", we infer that Beckham joined Real in the *year* 2003. Using *days* as our primary granularity for reasoning, we determine the possible earliest (begin) time point of starting his contract to be *2003-1-1* and the latest (end) time point as *2003-12-31* (using date-formatted time points for better readability). The begin and end time points then constitute an initial time interval *[2003-1-1, 2003-12-31]* for this observation (evidence) of the fact `joinsClub(Beckham, Real)` in the document. But then the question arises, how we should reconcile multiple of these (potentially inconsistent) observations, which we are likely to observe in different documents during the extraction phase, and how to represent these in a concise histogram for query processing.

Fact	Time Expression	Begin Time	End Time	Freq.	Event Type
joinsClub	"July, 2003"	2003-7-1	2003-7-31	2	*begin*
(Beckham, Real)	"Summer, 2003"	2003-6-1	2003-9-30	3	
leavesClub	"June, 2007"	2007-6-1	2007-6-30	1	*end*
(Beckham, Real)	"Early June, 2007"	2007-6-1	2007-6-10	2	
hasContract	"Season 2003	2003-7-1	2007-6-30	2	*during*
(Beckham, Real)	to 2007"				

Table 5.3.: Examples of time expressions and their corresponding intervals.

The extraction stage produces fact observations which may be valid at both a single time point (e.g., a day or a year) and entire intervals (e.g., multiple days or years). Staying in our football example, we aim to *aggregate* multiple observations of such events into a single *state* fact `playsForClub(Beckham, Real)[2003-1-1, 2007-12-31]`. This state fact for `playsForClub` can be inferred, for example, from two event facts `joinsClub(Beckham, Real)[2003-1-1, 2003-12-31]` and `leavesClub(Beckham, Real)[2007-1-1, 2007-12-31)`. Besides events that indicate the `begin` and `end` of an interval, we can also directly extract events that happened `during` the period when Beckham played for Real, such as `hasContract(Beckham, Real)[2003-7-1, 2007-6-30]`. Table 5.3 depicts a few examples of time expressions along with their corresponding intervals and possible observation frequencies as they occur at extraction time.

From these facts, we aim to derive the histogram for `playsForClub(Beckham, Real)`. Notice, that even in case a player might have played for a team multiple times (which occurs frequently), our approach allows for aggregating multiple overlapping observations of `begin`, `end` and `during` events into a single histogram.

Slicing and Coalescing. In analogy to temporal databases [33], different operations for reorganizing time intervals (and thus histograms) apply. For an *event* relation, we can slice an interval into any set of disjoint and contiguous subintervals by applying our uniformity assumption of confidences. Further, we can coalesce any two contiguous intervals into a single interval, only if the individual time points in both intervals have the same probability. In this case, the confidence of the coalesced interval is the sum of the confidences of the two input intervals. For a *state* relation, however, slicing intervals into subintervals is generally not allowed. Further, we can coalesce any two contiguous subintervals into a single interval, only if they have the same confidence. In this case, the confidence of the coalesced interval in a state relation is the same as the confidence of the two input intervals.

Merging Observations. Before presenting the forward and backward aggregation of event frequencies into a histogram, we first introduce the basic algorithm for reorganizing the bins of an output histogram, given two or more input histograms, as depicted in Algorithm 2. That is, at each time point where the confidence of an input histogram changes (i.e., at every interval boundary of an input interval), the confidence in the output histogram may change as well, and a new bin in the output histogram is created. Initially, the input histograms correspond to the basic intervals we extracted for the `begin`, `end` and `during` events (see Table 5.3).

This (binary) reorganization operation of bins is associative and commutative, hence multiple input histograms can be reorganized into a single output histogram in arbitrary order. Runtime and the number of bins in the output histogram are *linear* in the number of bins in the input histograms. Notice that the smallest-possible width of a histogram bin is a single time point.

Algorithm 2 Reorganizing Histograms.

```
Require: Two input histograms $H_1, H_2$
 1: Let $T$ be the disjoint union of begin and end time points from intervals in $H_1$ and $H_2$,
    respectively (in ascending order)
 2: Let $H_3$ be an empty output histogram
 3: Set $t_b$ := -1
 4: for each $t_e \in T$ do
 5:     if $t_b > -1$ then
 6:         Insert a new interval $[t_b, t_e)$ into $H_3$
 7:     end if
 8:     Set $t_b := t_e$
 9: end for
10:
11: return $H_3$
```

Forward and Backward Aggregation of Frequencies. As we have finished the histogram reorganization from the basic `begin`, `end` and `during` events, we continue to aggregate and normalize the frequencies for our fact in the target relation *playsForClub*. Intuitively, the confidence of the *playsForClub* should increase while we aggregate frequencies of intervals that indicate a *begin* event; it should increase at the begin of a *during* interval but decrease at the end of a *during* interval; and it should decrease for intervals relating to *end* events. The amount of observations for *begin* and *end* events may however be imbalanced, such that we also need to normalize the frequencies of each of these two types individually, before combining them into a single histogram. To obtain an increasing confidence from *begin* events, we cumulate frequencies of each bin from the first bin to the last bin (forward aggregation). In contrast, to obtain a decreasing confidence from *end* events, we cumulate frequencies of each bin from the last bin to the first one (backward aggregation).

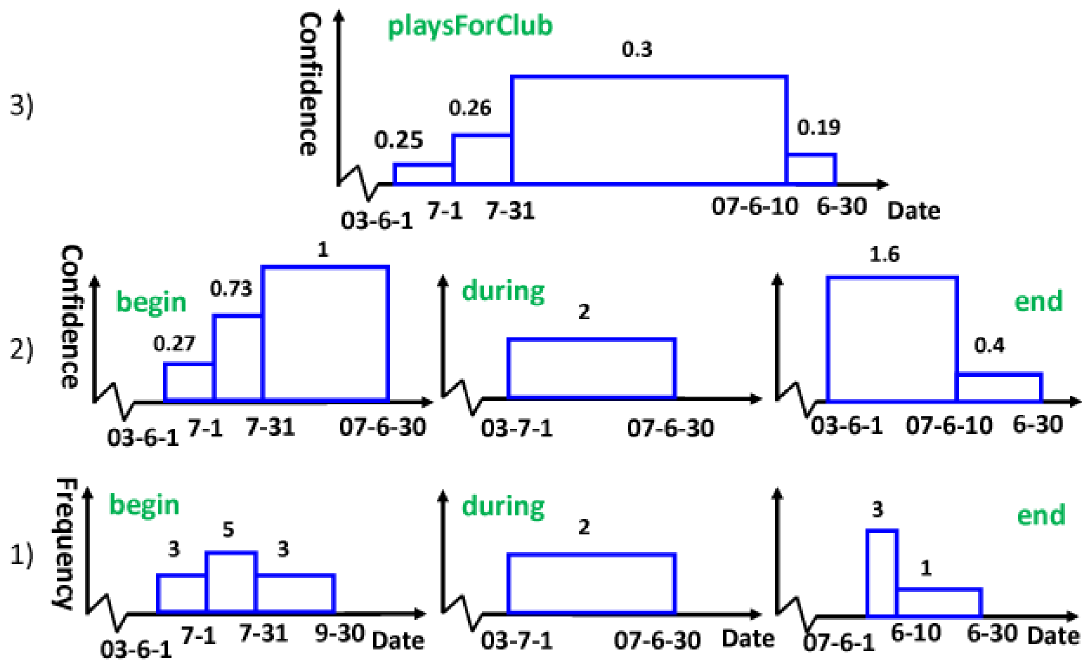


Figure 5.2.: Example for reorganizing and merging histograms based on the input facts from Table 5.3.

As shown in Figure 5.2, we first define the reorganized histograms H_1 and H_2 by aggregating the frequencies of all *begin* and *end* events of Table 5.3 according to their type. Forward aggregation then iterates over all bins of H_1 by cumulating the bins' weights as $H_1[i] = \sum_{0 \leq j \leq i} H_1[j]$, starting with the first bin $H_1[0]$. On the contrary, the backward aggregation iterates over all bins of H_2 by cumulating the bins' weights as $H_2[i] = \sum_{e \geq j \geq i} H_2[j]$, starting from the last bin $H_2[e]$. In the next step, both H_1 and H_2 are normalized to the weight of H_3, i.e., the aggregated histogram of all *during* events, before all three histograms are again aggregated and normalized to form the final confidence distribution of the *playsForClub* fact. In case no *during* event could be extracted from the sources, an artificial *during* interval with the earliest and latest time points of *begin* and *end* events with weight 1 can be created as H_3, in order to normalize H_1 and H_2. These various levels of aggregation are summarized in Algorithm 3.

Figure 5.2 provides an illustration of these three iterative reorganization and aggregation steps based on the facts in Table 5.3. We remark that this aggregation of frequencies is just one possible way of deriving an initial histogram at extraction time. In the following, we call facts like `playsForClub(Beckham, Real)`, which are obtained from such a forward/backward aggregation step, the *base facts*. Confidences in a probabilistic sense are traced back to only those base facts at reasoning time. Further, we assume these base facts to be *independent*.

Algorithm 3 Merging Histograms.

```
Require: Aggregated begin histogram $H_1$, end histogram $H_2$, and during histogram $H_3$
 1: Let $H_4$ be an empty output histogram
 2: Reorganize $H_1$, $H_2$, $H_3$, and $H_4$ using Algorithm 2
 3: Forward-cumulate begin histogram $H_1$, backward-cumulate end histogram $H_2$
 4: Normalize $H_1$ and $H_2$ such that $\sum_i H_1[i] = \sum_i H_3[i]$ and $\sum_i H_2[i] = \sum_i H_3[i]$
 5: for each $i \in H_4$ do
 6:     Set $H_4[i] := H_1[i] + H_2[i] + H_3[i]$
 7: end for
 8: Normalize $H_4$ such that $\sum_i H_4[i] = 1$
 9:
10: return $H_4$
```

5.3.3. Rule-based Reasoning, Lineage, and Possible Worlds

Our approach for reasoning in semantic knowledge bases is based on Datalog-like inference rules (Horn clauses), which can be employed to either enforce integrity constraints (Horn clauses with only negated literals) or provide means for actual inference and query answering (Horn clauses with exactly one positive literal). Recall that Horn clauses with exactly one positive literal can equivalently be rewritten as implications, where the positive literal becomes the head of the rule and the body is a conjunction of the remaining literals. Our key observation is that the logical dependencies of query answers (i.e., the *possible worlds* the entire knowledge base can take) are determined only by the way rules were processed in order to ground the query atoms (potentially recursively) down to the base facts. In this section, we focus on the case of Horn clauses with exactly one positive head literal, because it results in Boolean formulas with positive (i.e., conjunctive or disjunctive) lineage only.

Temporal Predicates. For reasoning about time intervals, we employ additional *temporal predicates* such as `overlaps`, `before`, `after`, etc. (see, e.g., Allen et al. [23] for an overview of temporal relations among intervals). These temporal predicates allow us to constrain the temporal relationships of time-annotated facts in the rules. Within the formulation of a rule, we also extend the given (binary) predicates by a third time variable t which is used as reference when reasoning with the temporal predicates (see Rules 5.5 and 5.6). While this extension clearly remains in first-order logic, it—strictly speaking—no longer conforms with the core RDF data model.

Queries. Queries in Datalog can be expressed as Boolean combinations of literals (again, we do not allow negation). Hence, $teammates(Beckham, x)$ would retrieve all teammates of Beckham, while $teammates(x, y)$ would denote all pairs of teammates that could be inferred from the knowledge base. Literals in queries are grounded against the knowledge base. Semantically, a disjunction of two literals relates to a disjoint union of two sets of facts (obtained from grounding each literal), while a conjunction relates to a set intersection. Set operations in these reasoning settings are always duplicate eliminating.

Conjunctive vs. Disjunctive Lineage. When processing a query, predicates in the body of an inference rule are combined conjunctively, while multiple rules with the same head predicate create a disjunctive derivation of the query answer. In analogy to probabilistic databases, processing the body of a rule thus conforms to a join operation with conjunctive lineage, whereas grounding the same derived fact from multiple rules conforms to a duplicate-elimination step with disjunctive lineage [6, 57]. We thus adopt a similar notion of data lineage as in [6] to compute the individual confidences of bins in the time histogram of a derived fact. In a Datalog-like setting, however, rules are potentially recursive, such that the derivation of answers typically is less uniform than for a regular SQL query or materialized view. Lineage however remains acyclic also in our setting, because all rules are grounded against base facts to find valid answers.

ID	Fact	Histogram	Relation Type
F_1	`playsForClub(Beckham, Real)`	[2003,2008):0.8	state
F_2	`playsForClub(Ronaldo, Real)`	[2002,2008):0.7	state
F_3	`winsCupForClub(Ronaldo,Real)`	[2003,2004):0.6	event

Table 5.4.: Base facts with time histograms (intervals).

As an example, consider we want to retrieve the probability of Beckham and Ronaldo being teammates for Rules 5.5 and 5.6 and the base facts depicted in Table 5.4. We will next discuss how confidence computation works in this setting.

$$\begin{aligned} \mathsf{teammates}(x,y) \;\leftarrow\;& \mathsf{playsForClub}(x,z,t_1) \wedge \mathsf{playsForClub}(y,z,t_2) \\ & \wedge \mathsf{notEquals}(x,y) \wedge \mathsf{overlaps}(t_1,t_2) \qquad (5.5) \\ \mathsf{teammates}(x,y) \;\leftarrow\;& \mathsf{playsForClub}(x,z,t_1) \wedge \mathsf{winsCupForClub}(y,z,t_2) \\ & \wedge \mathsf{notEquals}(x,y) \wedge \mathsf{overlaps}(t_1,t_2) \qquad (5.6) \end{aligned}$$

Confidence Computation. While grounding queries via rules yields exactly one Boolean lineage formula for a derived fact, the input confidences of base facts may vary across time. Hence our algorithm needs to ensure that the correct confidences are chosen as input when calculating the confidence of a result histogram. This is achieved via reorganizing the bins of the output histogram using Algorithm 2 and slicing and coalescing the input intervals of base facts belonging to an event relation accordingly. Notice that intervals from base facts belonging to a state relation do not have to be sliced, since a fact is defined to be valid at each time point of an interval with the same probability (see Section 3.3).

Thus, grounding the query `teammates(Beckham,x)` over the above rules (5.5) and (5.6) and base facts depicted in Table 5.4 results in the (single) grounded query answer `teammates(Beckham, Ronaldo)` with lineage $(F_1 \wedge F_2) \vee (F_1 \wedge F_3)$. However, by simply multiplying the probability of each literal in the lineage of `teammates(Beckham, Ronaldo)`, we would get $0.8 \times 0.7 \times 0.8 \times 0.6 = 0.2688$. This is not correct, since the probability of `playsForClub(Beckham, Real)` is considered twice. Assuming independence among base facts, we can calculate the correct probability of `teammates(Beckham, Ronaldo)` for the interval $[2003, 2004)$ as $0.8 \times 0.7 \times 0.6 + 0.8 \times 0.7 \times (1 - 0.6) + 0.8 \times (1 - 0.7) \times 0.6 = 0.704$ (as can be verified by a truth table). For simplicity, we show the confidence computation only for a single interval. In general, one such computation can be triggered for each bin of a time histogram (again using Algorithm 2 for reorganizing the histogram, but with a possible-worlds-based confidence computation instead of the simple aggregation of Algorithm 3).

Our approach for confidence computations with time histograms can thus be summarized into the following two steps:

1) reorganizing bins of the output histogram using Algorithm 2, and

2) computing the confidence for a fact's validity at each bin of its histogram.

While step 1) is linear in the number of input bins, each confidence computation per output bin is #P-complete for general Boolean formulas [54]. We thus employ the Luby-Karp family of sampling algorithms for approximating the confidence computation. Different versions for Luby-Karp sampling [37] are available, depending on whether the formula is in CNF, DNF, or of generic Boolean shape, each with different approximation guarantees. Thus, as a simple optimization, our implementation is able to check for the structure of the formulas at query time, and it can select the most appropriate variant of Luby-Karp, or even an exact confidence computation if this is still feasible.

In our current implementation, lineage is transient, i.e., we keep lineage information only in memory at query processing time. For future work, we aim to investigate also making lineage persistent, thus being able to "learn" new facts from existing facts in the knowledge base and storing these derived facts along with their derivation in the knowledge base for further processing and faster subsequent inference.

5.3.4. Experiments

System Setup and Experiments

Our system is implemented as an extension of URDF [71, 48], which is a framework for efficient reasoning over uncertain RDF knowledge bases developed at the Max Planck Institute for Informatics. URDF employs SLD resolution for grounding first-order formulas (Horn clauses) against an underlying knowledge base. Unlike most Datalog engines, URDF follows a top-down grounding approach, i.e., for an incoming query, it aims to resolve answers by processing rules recursively from the head predicate down to the body predicates, which are conjunctions of predicates found either in the knowledge base or which can in turn be processed via the head predicate of a rule. URDF is implemented in Java 1.6 with about 4,000 lines of code. All experiments were run on an Intel Xeon 2.40GHz server (in single-threaded mode) with 48GB RAM. We use Oracle 10g as backend for storing the T-YAGO knowledge base, which was installed on a second AMD Opteron 2.6 GHz server with 32GB RAM.

As competitors we employ the original URDF framework (without the temporal extension) and the IRIS [8] reasoner, a default reasoning engine used in many Semantic Web applications. In terms of reasoning, IRIS [8] is an open-source Datalog engine supporting built-in predicates. It is designed to be highly configurable, allowing for different Datalog evaluation strategies and the definition of custom data types and predicates.

Timely YAGO Knowledge Base Our experiments are based on the semantic graph of T-YAGO (see Chapter 4). For T-YAGO, we extracted more than 270K temporal facts from Wikipedia and free-text, with 16 distinct relationship types. Currently it covers the football domain, including relationships such as *playsForSeniorClub*, *participatedIn* and *winsCup*, but also fact observations for the *begin*, *end*, and *during* events of these relations, such as *joinsSeniorClub* or *leavesSeniorClub*. These raw facts can be integrated with the existing facts of the corresponding relations (e.g., *playsForSeniorClub*), in order to reconcile time histograms using the aggregation rules depicted in Table A.1.

The facts and time histograms are stored in two separate tables. The facts table contains three columns for RDF triplets (i.e., first argument, second argument, and relation name) and a column for the fact id. The time table is composed of two columns (i.e., start time point and end time point) corresponding to the begin and end time point of an interval, a foreign key connecting to the fact's id, and a column for the fact's confidence at the interval.

Rules and Queries

Table A.1 depicts 4 aggregation rules for reasoning about the time interval of a player's or coach's career period, as well as 9 partly recursive, hand-crafted inference rules for reasoning about people's activities and relationships in the football domain. As URDF (without time) and IRIS do not support time-aware reasoning, we remove all temporal predicates in the inference rules, such as `overlaps` or `after`, when comparing their runtimes and results. Table A.1 illustrates 8 queries including single-fact queries, chains, stars and cliques of interconnected facts used as our baseline for experiments.

Experimental Results

Our experiments focus on investigating the overhead of time-aware query processing, compared to a time-oblivious setting. We compare the running times and result precision of URDF (with time) to IRIS and URDF (without time). The running time of URDF (with time) includes grounding time (using SLD resolution) and histogram creation (i.e., possible-worlds-based histogram calculation time).

Baseline Runs Without Time. Since IRIS and URDF (without time) do not support time-aware reasoning, we compare grounding time and result precision of IRIS to URDF (without time) in the first experiment. The grounding time in URDF (without time) denotes the time to ground the query atoms, using the inference rules and queries depicted in Table A.1. The measured time in IRIS is the time required to ground the query using magic sets rewriting, which includes both the rule rewriting step followed by a bottom-up query evaluation over the rewritten rules. We can see that URDF already outperforms IRIS for the grounding time (both without using time-specific predicates).

	Without time information				With time histograms			T-URDF/URDF
	IRIS	URDF	PWs-conf	#	T-URDF	PWs-conf	#	
	ms	ms	ms	results	ms	ms	results	precision
Q_1	6893	35	2	8	45	<1	8	8/8
Q_2	821	11	<1	5	12	<1	5	8/8
Q_3	7127	1905	1191	766	2113	1	184	184/766
Q_4	6686	699	188	239	308	5	58	58/239
Q_5	7628	3099	314	190	1423	51	114	114/190
Q_6	4317	693	20345	14	1054	87600	14	8/8
Q_7	6909	6712	574	183	3277	5	17	17/183
Q_8	7125	6396	190	133	4441	1	25	25/133
$\sum$	47506	19550	<22805	1538	12673	<87665	425	avg=0.545

Table 5.5.: Experimental results.

Overhead of Confidence Computations with Histograms. In the second experiment, we compare the grounding time and result precision of URDF (without time) to URDF (with time). Besides the grounding time consumed by URDF (without time), URDF (with time) also includes the possible-worlds-based histogram computation time. A comparable confidence computation for facts with just a single confidence value but without a time histogram is also shown on the left-hand side for URDF (without time).

Interestingly, Table 5.5 shows that URDF (with time) even partly achieves better runtimes than URDF (without time) for complex queries, because URDF (with time) does not ground any answers that do not satisfy the temporal predicates. This is also the main reason for the lower precision of URDF (without time) compared to URDF (with time). The grounding time of URDF (with time) is better than URDF (without time) for Queries 4, 5, 7 and 8, even when taking also the time for building the final histogram into account. However, the time for building the histogram for Query 6 is much worse than the others, yielding 14 results with 6,552 literals in 504 disjunctions in their lineage. Also, only for Query 6 we needed to employ Luby-Karp-based sampling (using $\varepsilon = 0.05$ and $\delta = 0.05$), while all the other confidences could be computed exactly.

5.3.5. Conclusions

Time-aware information extraction increases the demand for coping with imprecise or otherwise uncertain data and is an excellent showcase for uncertain data management. In our approach, we show that adding time histograms involves only a light overhead over a comparable probabilistic setting that does not consider time. Time-aware reasoning may even spare unnecessary computations for false-positive answers at an early stage and thus reduce the overall runtime for query answering.

Chapter 6.

Tools

6.1. Interactive Knowledge Harvesting

6.1.1. Motivation

Acquiring high-quality (temporal) facts for knowledge bases is a labor-intensive process. Although there has been recent progress in the area of semi-supervised fact extraction to which this dissertation contributes, these approaches still have limitations, including a restricted corpus, a fixed set of relations to be extracted or a lack of assessment capabilities.

To lower the bar of customized fact extraction from free-text, we introduce an *interactive* knowledge harvesting system called PRAVDA-live[1]. Our system is an out-of-the-box solution to address this issue by covering all subproblems within a web-based interface. This allows users of all provenance (from greenhorns to experts) to perform fact extraction on their own.

Problem Setting. Suppose the user provides the whole Wikipedia corpus as the input, and he/she wants to know the birth place and date of all the people in Wikipedia. PRAVDA-live can guide the user to query the system by using the relation *bornIn*, which connects a subject of type person with an object of type location. And then the user is requested to label limited number of seed facts. Afterwards, our system can be employed to automatically distill the fact, for example `bornIn(Einstein, Ulm)@14-03-1879` from the sentence: *"Albert Einstein was born in Ulm, in the Kingdom of Württemberg in the German Empire on*

[1] `http://www.mpi-inf.mpg.de/yago-naga/pravda/`

14 March 1879."

Contributions. We present a system called PRAVDA-live, which supports fact extraction of *user-defined* relations from ad-hoc selected text documents and ready-to-use RDF exports. Further features include the support for temporally annotated relations, customized or mined extraction-patterns, and a constraint solver being able to clean extracted facts by inter-fact constraints.

6.1.2. Framework

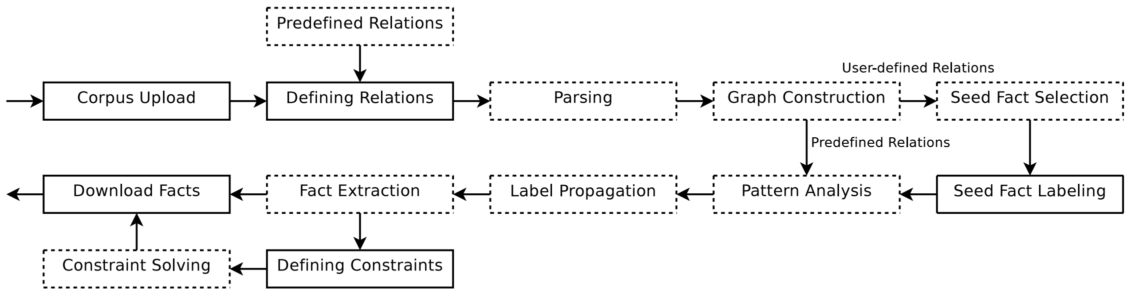


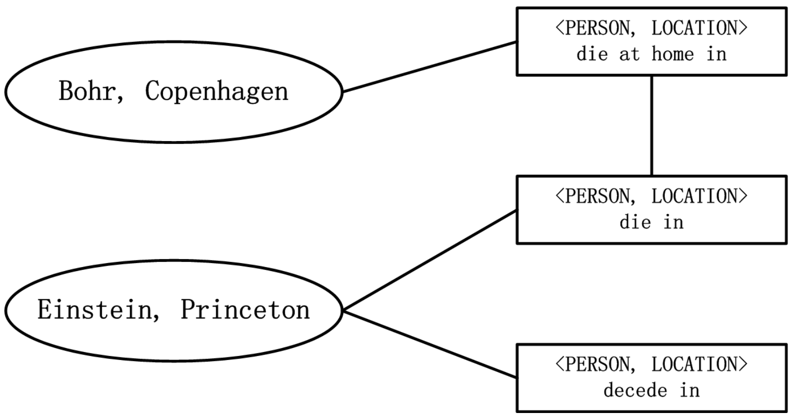
Figure 6.1.: System workflow.

PRAVDA-live is constructed based on the framework of PRAVDA (see Figure 5.1) presented in Section 4.3 and Section 5.2 by adding user interactive features. The new framework is depicted in Figure 6.1, where boxes with continuous lines require user-interaction. As a first step, the user uploads a corpus for the extraction process. Then, the user either selects predefined relations or defines customized relations or both. What follows are the parsing and graph construction stages performed in the backend. If customized relations are present, the next step is the seed fact selection followed by manual labeling of these. Afterwards we continue with pattern analysis and label propagation, eventually yielding extracted facts. Finally, there is the option to define and apply constraints to the extracted facts, or to download the facts immediately.

6.1.3. Algorithms

Parsing. Once the corpus is uploaded and the relations of interest are known, our system proceeds with the parsing stage. We provide several approaches for named entity recognition and disambiguation. One approach follows Section 4.3, entities are recognized and disambiguated by leveraging YAGO [62]. The other approaches utilize AIDA framework [30]. The surface string between two entities are lifted to patterns by considering n-grams of nouns, verbs converted to present tense, and prepositions. A detailed description of patterns is beyond the scope of this section; we refer the reader to Section 4.3. Finally, we detect temporal expressions by regular expressions.

Graph Construction. As in Section 4.3 we construct an undirected graph $G = (V, E)$ comprising two types of vertices $V = V_e \dot{\cup} V_p$. The former set V_e contains one vertex per entity pair discovered in the corpus and the latter set V_p has one vertex per pattern. Edges between V_p and V_e are added, if an entity pair occurs in a pattern. Additional edges between vertices in V_p are derived from similarities among patterns. An example graph is shown in Figure 6.2, where oval vertices belong to V_e and box-shaped vertices are members of V_p.



<PERSON, LOCATION> die at home in
Bohr, Copenhagen
<PERSON, LOCATION> die in
Einstein, Princeton
<PERSON, LOCATION> decede in

Figure 6.2.: Example graph.

Seed Fact Selection. In previous works seed facts were chosen entirely manually. To ease usability of our system, we develop a novel ranking algorithm returning the entity pairs to be labelled by the user. The algorithm is presented in Section 6.1.4. This stage is skipped for problem instances with only predefined relations, where labelled seed facts are available in our system.

Pattern Analysis. In this stage, the labelled seed facts are employed to compute an initial weighting of the patterns, being derived from frequency counts of both the patterns as well as the entity pairs. A more detailed description is available in Section 4.3.

Label Propagation. Building on the work presented in Section 4.3 we utilize Label Propagation [66] to determine the relation expressed by each pattern. Here, the labelled seed facts and patterns serve as input. Label Propagation is a semi-supervised learning algorithm, where the supervision results from the labelled seed facts. It passes labels on the previously described graph (see Figure 6.2), where each label corresponds to a relation.

Fact Observation Extraction. After Label Propagation has terminated, the entity pair vertices which hold a relation's label weighted above a threshold for a fact (see Section 4.3).

Constraint Solving. Given user-defined constraints and a set of extracted facts, we intend to select a maximal consistent subset of facts. The resulting optimization problem is encoded into an integer linear program as the work presented in Section 5.2.

6.1.4. Seed Fact Selection

Our seed fact selection procedure acts on the graph introduced in Section 6.1.2. By construction the graph consists of disconnected components corresponding to a different pair of entity types. Considering a single connected component, we cluster its vertices reflecting patterns in the following manner: If two patterns have an identical main verb and last preposition, they belong to the same cluster. For example, in the disconnected component of Figure 6.2, we cluster *'die in'* and *'died at home in'* with pair of entity type – "PERSON, LOCATION". More formally, a disconnected component is defined as $C = (V_e \dot{\cup} (\dot{\bigcup}_i V_{p,i}), E)$, where V_e are the vertices standing for entity pairs (fact

Algorithm 4 Seed Fact Selection

```
Require: Component $C = (V_e \dot{\cup} (\dot{\bigcup}_i V_{p,i}), E)$, number of seeds $k$
1: $S = \emptyset$   //Seed facts to return
2: for $V_{p,i} \in C$ by decreasing $|V_{p,i}|$ do
3:     //select the vertice $v_e$ (not in S) with the maximum degree
4:     //from $V_e$ connecting with the pattern cluster $V_{p,i}$
5:     $v_e := \mathrm{argmax}_{v_e \in V_e \setminus S, \exists v_p \in V_{p,i}:(v_e,v_p) \in E} degree(v_e)$
6:     $S := S \cup \{v_e\}$
7:     if $|S| \geq k$ then
8:         break
9:     end if
10: end for
11:
12: return S
```

candidates), and each $V_{p,i}$ represents a cluster of vertices embodying patterns with a certain pair of entity type. With respect to Figure 6.2, we have $V_e = \{(Bohr, Copenhagen), (Einstein, Princeton)\}$ and $V_{p,0} = \{$*die at home in*, *die in*$\}$, $V_{p,1} = \{$*decede in*$\}$, for instance. To each disconnected component we apply Algorithm 4 implementing a greedy strategy. The goal of this algorithm is to select prominent fact candidates in each cluster.

The loop in Line 2 repeatedly cycles through all clusters of pattern vertices $V_{p,i}$ beginning with the largest cluster. It stops when k seed facts have been determined. For each cluster we add an entity-pair-vertex v_e (not in S) as seed, which has maximum degree and is connected to $V_{p,i}$ (Line 5). In Figure 6.2 the largest cluster is $V_{p,0} = \{$*die at home in*, *die in*$\}$ where the algorithm selects $(Einstein, Princeton)$ since its degree is maximal.

6.1.5. System Implementation

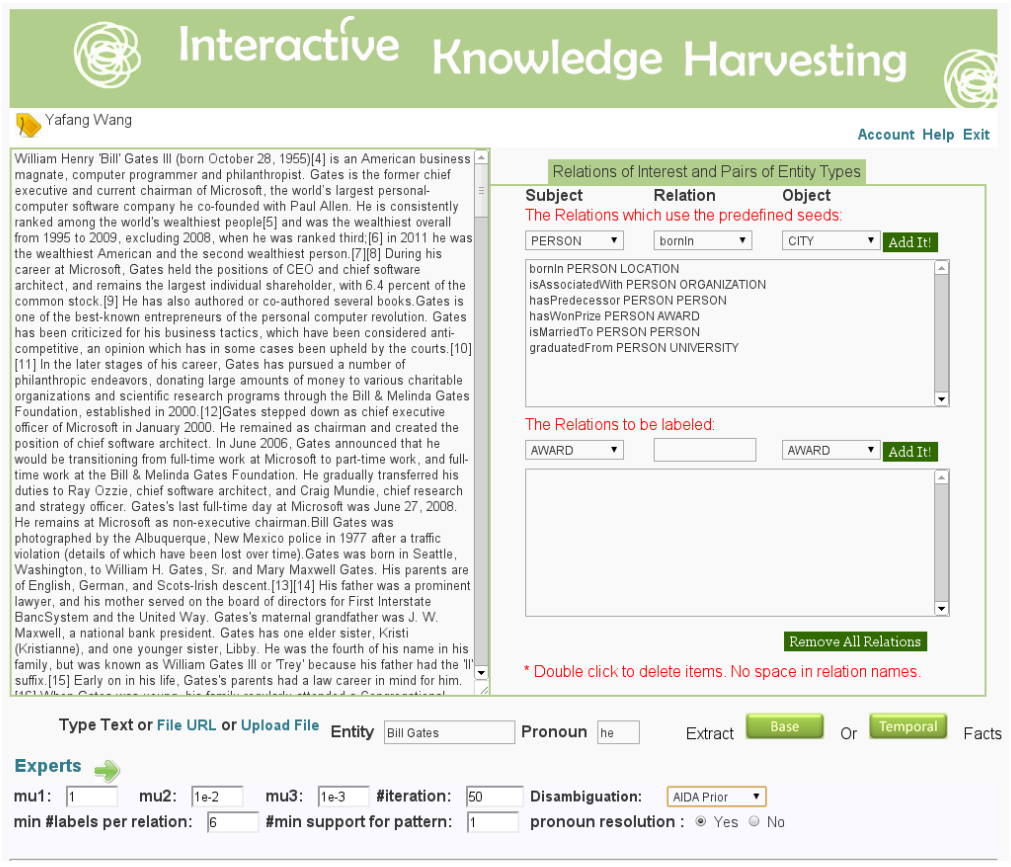
PRAVDA-live is implemented in Java, where Apache Tomcat[2] acts as Webserver. While parsing text documents we employ OpenNLP [3] for part of speech tagging, converting verbs to present tense and stemming nouns.

[2]`http://tomcat.apache.org/`

[3]`http://opennlp.apache.org/`

Furthermore, the integer linear program tackling the constraints utilizes Gurobi[4]. All data is managed by a PostgreSQL[5] database.

User Interface



Yafang Wang

William Henry 'Bill' Gates III (born October 28, 1955)[4] is an American business magnate, computer programmer and philanthropist. Gates is the former chief executive and current chairman of Microsoft, the world's largest personal-computer software company he co-founded with Paul Allen. He is consistently ranked among the world's wealthiest people[5] and was the wealthiest overall from 1995 to 2009, excluding 2008, when he was ranked third;[6] in 2011 he was the wealthiest American and the second wealthiest person.[7][8] During his career at Microsoft, Gates held the positions of CEO and chief software architect, and remains the largest individual shareholder, with 6.4 percent of the common stock.[9] He has also authored or co-authored several books.Gates is one of the best-known entrepreneurs of the personal computer revolution. Gates has been criticized for his business tactics, which have been considered anti-competitive, an opinion which has in some cases been upheld by the courts.[10][11] In the later stages of his career, Gates has pursued a number of philanthropic endeavors, donating large amounts of money to various charitable organizations and scientific research programs through the Bill & Melinda Gates Foundation, established in 2000.[12]Gates stepped down as chief executive officer of Microsoft in January 2000. He remained as chairman and created the position of chief software architect. In June 2006, Gates announced that he would be transitioning from full-time work at Microsoft to part-time work, and full-time work at the Bill & Melinda Gates Foundation. He gradually transferred his duties to Ray Ozzie, chief software architect, and Craig Mundie, chief research and strategy officer. Gates's last full-time day at Microsoft was June 27, 2008. He remains at Microsoft as non-executive chairman.Bill Gates was photographed by the Albuquerque, New Mexico police in 1977 after a traffic violation (details of which have been lost over time).Gates was born in Seattle, Washington, to William H. Gates, Sr. and Mary Maxwell Gates. His parents are of English, German, and Scots-Irish descent.[13][14] His father was a prominent lawyer, and his mother served on the board of directors for First Interstate BancSystem and the United Way. Gates's maternal grandfather was J. W. Maxwell, a national bank president. Gates has one elder sister, Kristi (Kristianne), and one younger sister, Libby. He was the fourth of his name in his family, but was known as William Gates III or 'Trey' because his father had the 'II' suffix.[15] Early on in his life, Gates's parents had a law career in mind for him.

Relations of Interest and Pairs of Entity Types

Subject Relation Object

The Relations which use the predefined seeds:

PERSON bornIn CITY Add It!

bornIn PERSON LOCATION
isAssociatedWith PERSON ORGANIZATION
hasPredecessor PERSON PERSON
hasWonPrize PERSON AWARD
isMarriedTo PERSON PERSON
graduatedFrom PERSON UNIVERSITY

The Relations to be labeled:

AWARD AWARD Add It!

Remove All Relations

* Double click to delete items. No space in relation names.

Type Text or File URL or Upload File Entity: Bill Gates Pronoun: he Extract Base Or Temporal Facts

Experts

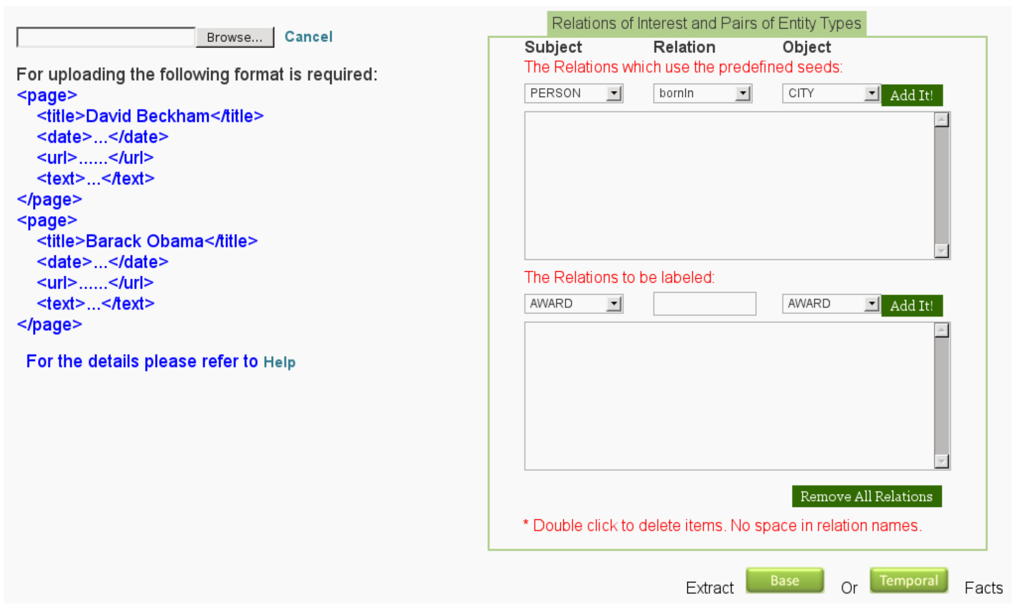
mu1: 1 mu2: 1e-2 mu3: 1e-3 #iteration: 50 Disambiguation: AIDA Prior

min #labels per relation: 6 #min support for pattern: 1 pronoun resolution : [✓ Yes] No

Figure 6.3.: Paste text.

1. **Corpus Upload.** The user interface offers the opportunity for both pasting text into a text field and uploading larger text files. In Figure 6.3 the text field holds excerpts from Bill Gate's Wikipedia article. The user can also choose to upload a file (or a URL showing the address of the file) containing many documents in an XML-like format (see Figure 6.4 and Figure 6.5). We allow experts to modify the parameters used in seed facts

[4]http://www.gurobi.com/
[5]http://www.postgresql.org/


Browse... Cancel

For uploading the following format is required:

```
<page>
    <title>David Beckham</title>
    <date>...</date>
    <url>......</url>
    <text>...</text>
</page>
<page>
    <title>Barack Obama</title>
    <date>...</date>
    <url>......</url>
    <text>...</text>
</page>
```

For the details please refer to Help

Relations of Interest and Pairs of Entity Types

Subject	Relation	Object

The Relations which use the predefined seeds: PERSON bornIn CITY Add It!

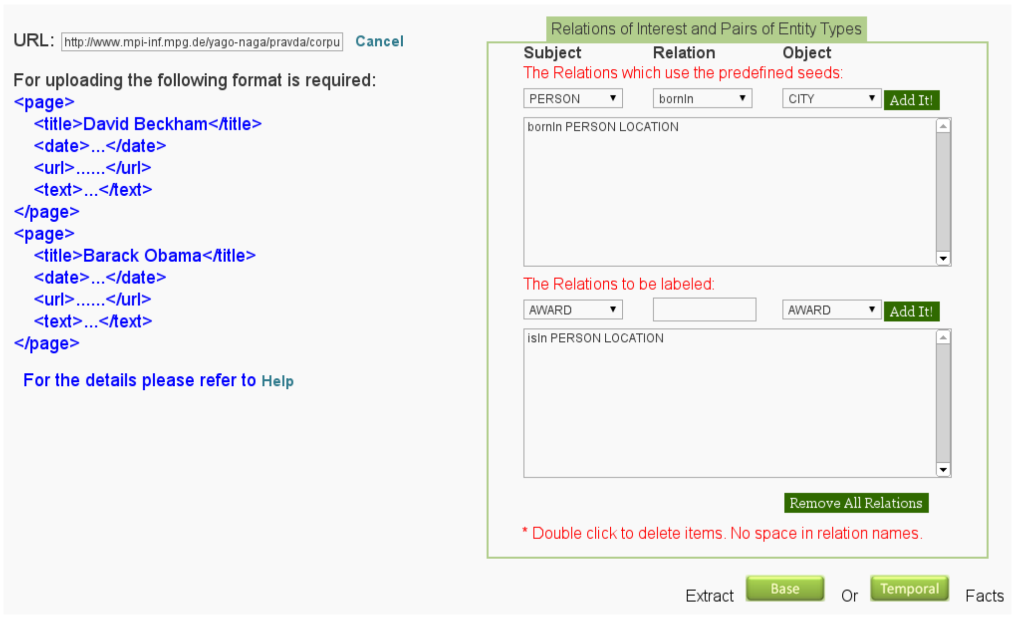
The Relations to be labeled: AWARD AWARD Add It!

Remove All Relations

* Double click to delete items. No space in relation names.

Extract Base Or Temporal Facts

Figure 6.4.: Upload a file.


URL: http://www.mpi-inf.mpg.de/yago-naga/pravda/corpu Cancel

For uploading the following format is required:

```
<page>
    <title>David Beckham</title>
    <date>...</date>
    <url>......</url>
    <text>...</text>
</page>
<page>
    <title>Barack Obama</title>
    <date>...</date>
    <url>......</url>
    <text>...</text>
</page>
```

For the details please refer to Help

Relations of Interest and Pairs of Entity Types

Subject	Relation	Object

The Relations which use the predefined seeds: PERSON bornIn CITY Add It!

bornIn PERSON LOCATION

The Relations to be labeled: AWARD AWARD Add It!

isIn PERSON LOCATION

Remove All Relations

* Double click to delete items. No space in relation names.

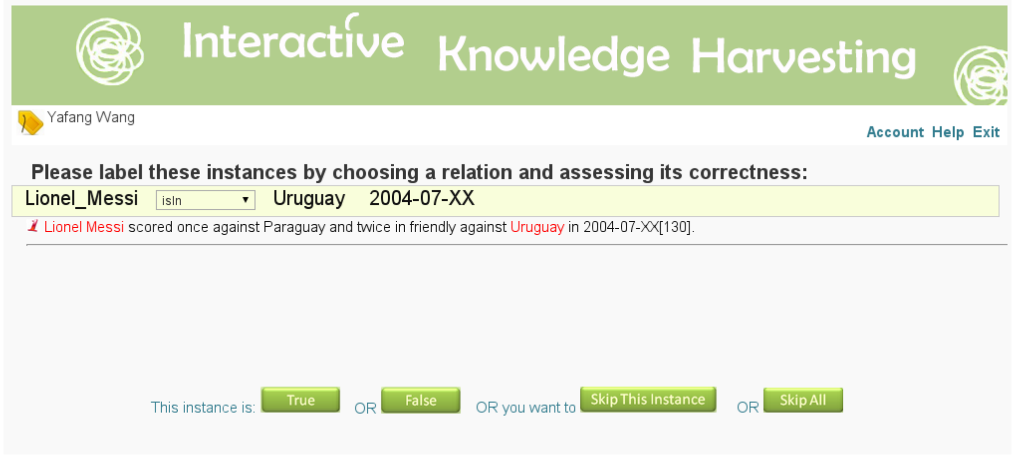
Extract Base Or Temporal Facts

Figure 6.5.: Upload a URL showing the address of a file.

labeling, pattern analysis and label propagation. Users are also free to choose the disambiguation approach (see the bottom part of Figure 6.3).

2. **Defining Relations.** By investigating the text on the left in Figure 6.3, Figure 6.4 and Figure 6.5, the user can define relations of interest. There

are two lists of relations. The upper list holds exclusively predefined relations, such as *bornIn* typed by *person* and *location* in the screenshot. On the other hand, the lower list may contain both predefined and user-defined relations, since on these the seed fact selection process will be invoked. In Figure 6.5, *isIn* typed by *person* and *location* is user-defined relation, which invokes the seed fact selection.

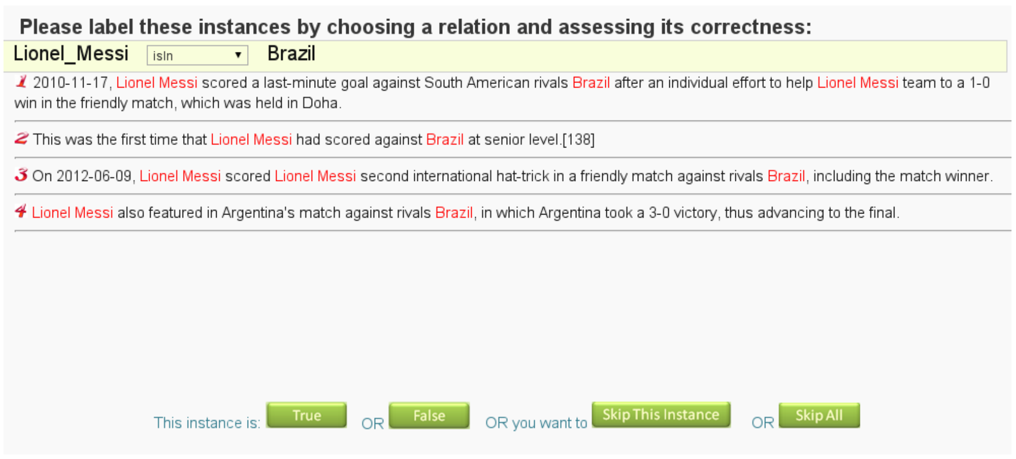

Interactive Knowledge Harvesting
Yafang Wang
Account Help Exit
This instance is: True OR False OR you want to Skip This Instance OR Skip All

Please label these instances by choosing a relation and assessing its correctness:

Lionel_Messi isIn Uruguay 2004-07-XX

1 Lionel Messi scored once against Paraguay and twice in friendly against Uruguay in 2004-07-XX[130].

Figure 6.6.: Label seed temporal facts.


This instance is: True OR False OR you want to Skip This Instance OR Skip All

Please label these instances by choosing a relation and assessing its correctness:

Lionel_Messi isIn Brazil

1 2010-11-17, Lionel Messi scored a last-minute goal against South American rivals Brazil after an individual effort to help Lionel Messi team to a 1-0 win in the friendly match, which was held in Doha.

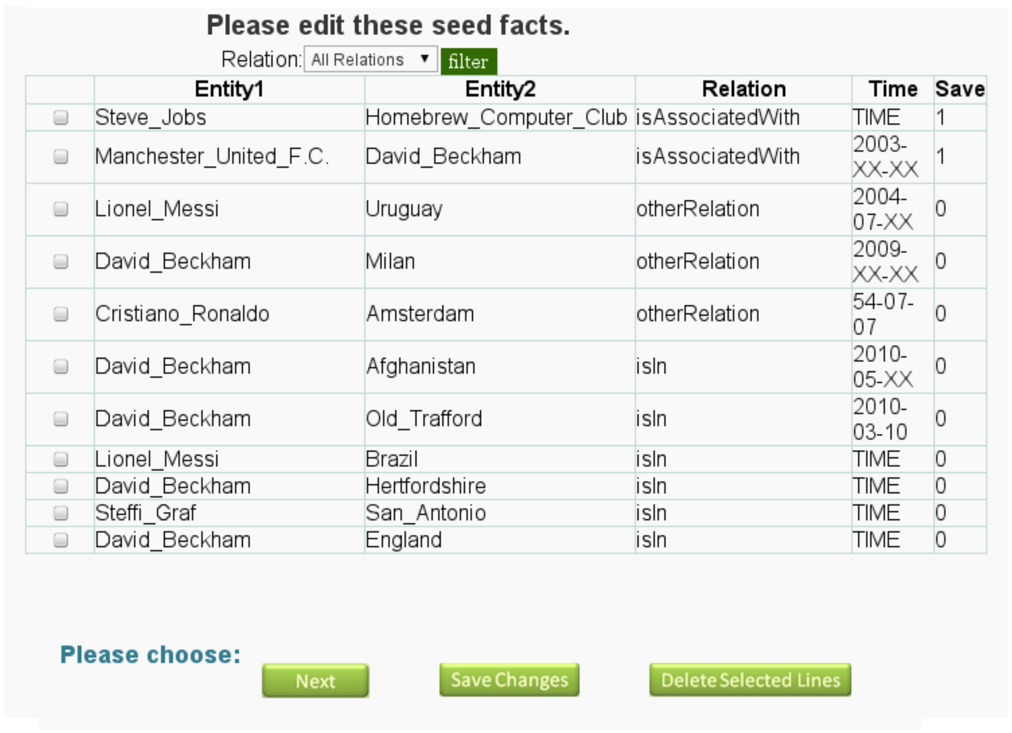
2 This was the first time that Lionel Messi had scored against Brazil at senior level.[138]

3 On 2012-06-09, Lionel Messi scored Lionel Messi second international hat-trick in a friendly match against rivals Brazil, including the match winner.

4 Lionel Messi also featured in Argentina's match against rivals Brazil, in which Argentina took a 3-0 victory, thus advancing to the final.

Figure 6.7.: Label seed base facts.

3. **Seed Labeling.** During the labeling process text snippets are displayed, where recognized entities are marked in red. It is the user's task to select the correct relation (here "*isIn*") connecting both entities (see Figure 6.6



Please edit these seed facts.

Relation: All Relations filter

	Entity1	Entity2	Relation	Time	Save
▢	Steve_Jobs	Homebrew_Computer_Club	isAssociatedWith	TIME	1
▢	Manchester_United_F.C.	David_Beckham	isAssociatedWith	2003-XX-XX	1
▢	Lionel_Messi	Uruguay	otherRelation	2004-07-XX	0
▢	David_Beckham	Milan	otherRelation	2009-XX-XX	0
▢	Cristiano_Ronaldo	Amsterdam	otherRelation	54-07-07	0
▢	David_Beckham	Afghanistan	isIn	2010-05-XX	0
▢	David_Beckham	Old_Trafford	isIn	2010-03-10	0
▢	Lionel_Messi	Brazil	isIn	TIME	0
▢	David_Beckham	Hertfordshire	isIn	TIME	0
▢	Steffi_Graf	San_Antonio	isIn	TIME	0
▢	David_Beckham	England	isIn	TIME	0

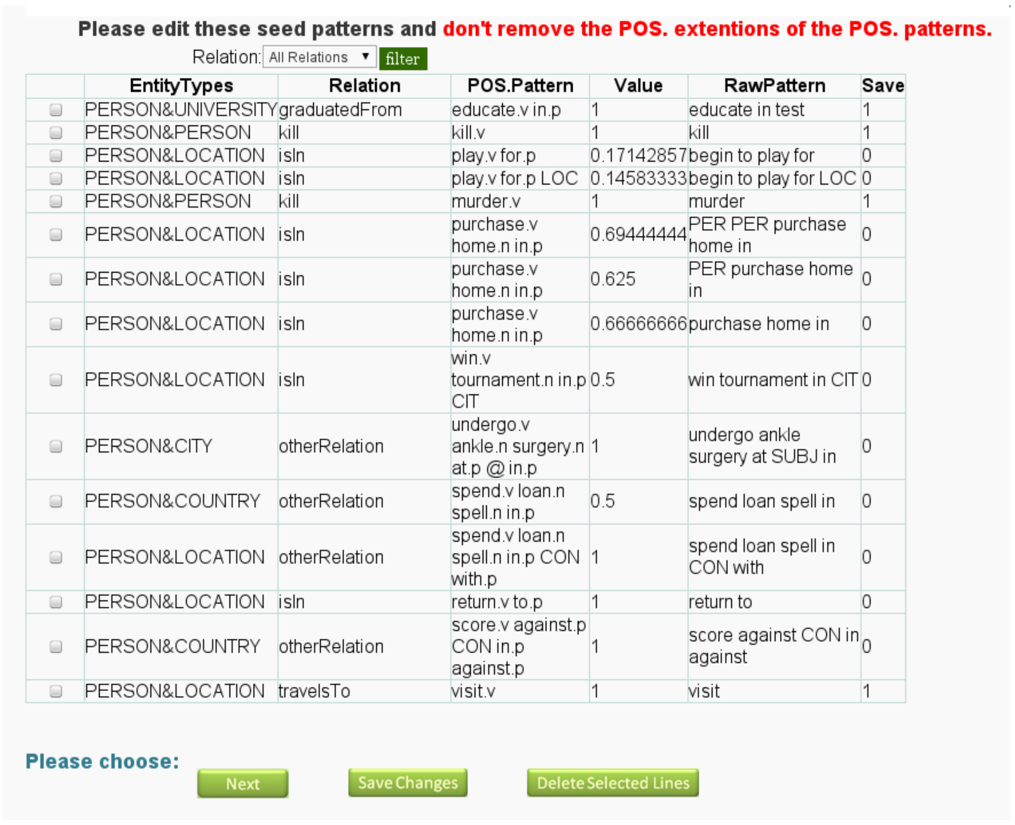
Please choose:

Figure 6.8.: All the labelled seed facts.

and Figure 6.7). Seed facts can be labelled as true or false, as indicated by the respective buttons. All the labelled seed facts (including both base and/or temporal seed facts) can be checked or modified from the interface in Figure 6.8. If the user clicks the button “Next”, the *pattern analysis* stage will be invoked. All the patterns which are labelled with initial values are shown in Figure 6.9. They are used for the next stage – *fact observation extraction*. The patterns are set as *to-be-deleted* by default (*Save* is 0 in Figure 6.9). Every *new-log-in* or *new-start* deletes all the patterns with the save status in “0”. Users are free to change the status of good patterns to “1” to keep them for future use.

4. **Fact Observations Evaluation.** The extracted fact observations can be evaluated via the assessment interface. Figure 6.10 and Figure 6.11 illustrate the extracted base and temporal fact observations respectively.

5. **Defining Constraints.** PRAVDA-live allows the specification of constraints to be applied to the extracted fact observations. In the example screenshot of Figure 6.12, we require birthdays to precede dates



Please edit these seed patterns and don't remove the POS. extentions of the POS. patterns.

Relation: All Relations filter

	EntityTypes	Relation	POS.Pattern	Value	RawPattern	Save
▢	PERSON&UNIVERSITY	graduatedFrom	educate.v in.p	1	educate in test	1
▢	PERSON&PERSON	kill	kill.v	1	kill	1
▢	PERSON&LOCATION	isIn	play.v for.p	0.17142857	begin to play for	0
▢	PERSON&LOCATION	isIn	play.v for.p LOC	0.14583333	begin to play for LOC	0
▢	PERSON&PERSON	kill	murder.v	1	murder	1
▢	PERSON&LOCATION	isIn	purchase.v home.n in.p	0.69444444	PER PER purchase home in	0
▢	PERSON&LOCATION	isIn	purchase.v home.n in.p	0.625	PER purchase home in	0
▢	PERSON&LOCATION	isIn	purchase.v home.n in.p	0.66666666	purchase home in	0
▢	PERSON&LOCATION	isIn	win.v tournament.n in.p CIT	0.5	win tournament in CIT	0
▢	PERSON&CITY	otherRelation	undergo.v ankle.n surgery.n at.p @ in.p	1	undergo ankle surgery at SUBJ in	0
▢	PERSON&COUNTRY	otherRelation	spend.v loan.n spell.n in.p	0.5	spend loan spell in	0
▢	PERSON&LOCATION	otherRelation	spend.v loan.n spell.n in.p CON with.p	1	spend loan spell in CON with	0
▢	PERSON&LOCATION	isIn	return.v to.p	1	return to	0
▢	PERSON&COUNTRY	otherRelation	score.v against.p CON in.p against.p	1	score against CON in against	0
▢	PERSON&LOCATION	travelsTo	visit.v	1	visit	1

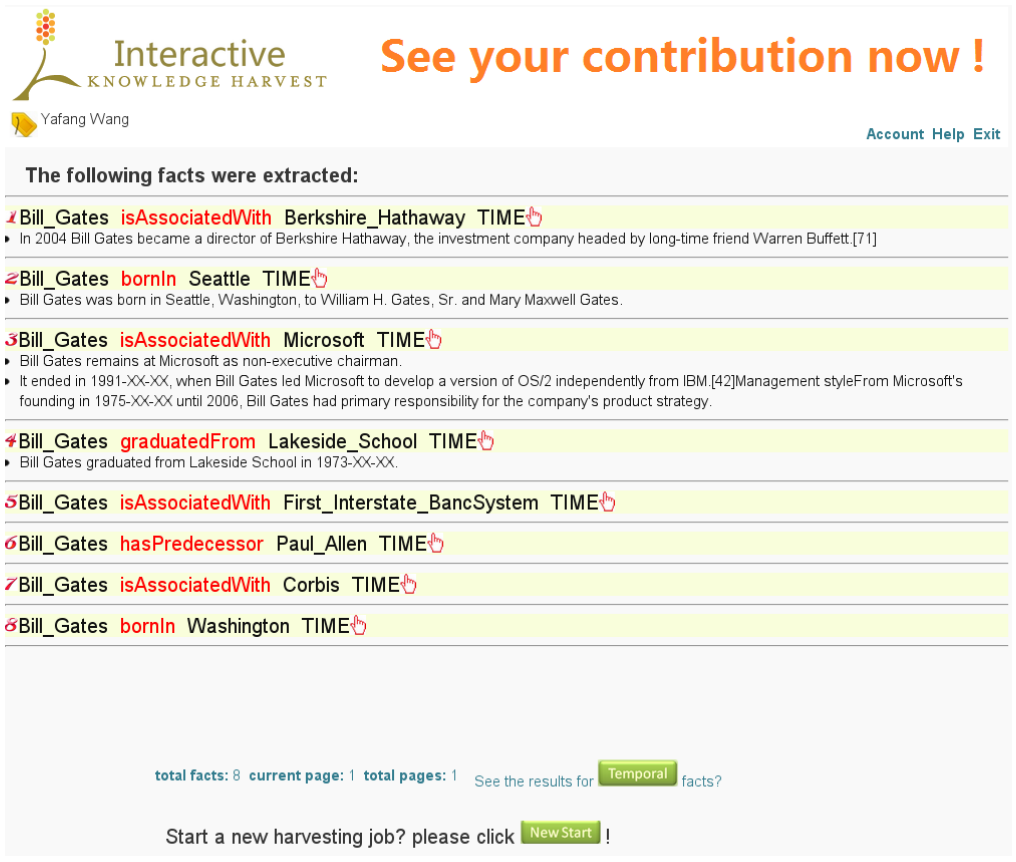
Please choose:

Figure 6.9.: All the seed patterns from pattern analysis.

of death and we disallow the time annotation of *livesIn* to exceed 120 years in length. As for non-temporal constraints, we enforce *bornIn* to be functional, such that persons can be born in at most one location.

6. **Downloads.** Finally, the parsed fact candidates, the extracted fact observations and the clean facts are exported as RDF documents, which can be downloaded from the interface in Figure 6.13.

7. **Adding and Editing Patterns.** The user is free to manually add any new patterns of the interesting relations by the interface shown in Figure 6.14. In the example, the user creates a pattern "visited" for relation "travelsTo". Afterwards, the user can edit/delete the added patterns via the interface illustrated in Figure 6.15. Once the "Save" value is set as "1", the pattern will be used for the future extraction. In Figure 6.9, the patterns with the save value "1" are the old ones from either manually added patterns or previous pattern analysis stage. These patterns, together with the new



The following facts were extracted:

1 Bill_Gates isAssociatedWith Berkshire_Hathaway TIME
- In 2004 Bill Gates became a director of Berkshire Hathaway, the investment company headed by long-time friend Warren Buffett.[71]

2 Bill_Gates bornIn Seattle TIME
- Bill Gates was born in Seattle, Washington, to William H. Gates, Sr. and Mary Maxwell Gates.

3 Bill_Gates isAssociatedWith Microsoft TIME
- Bill Gates remains at Microsoft as non-executive chairman.
- It ended in 1991-XX-XX, when Bill Gates led Microsoft to develop a version of OS/2 independently from IBM.[42]Management styleFrom Microsoft's founding in 1975-XX-XX until 2006, Bill Gates had primary responsibility for the company's product strategy.

4 Bill_Gates graduatedFrom Lakeside_School TIME
- Bill Gates graduated from Lakeside School in 1973-XX-XX.

5 Bill_Gates isAssociatedWith First_Interstate_BancSystem TIME

6 Bill_Gates hasPredecessor Paul_Allen TIME

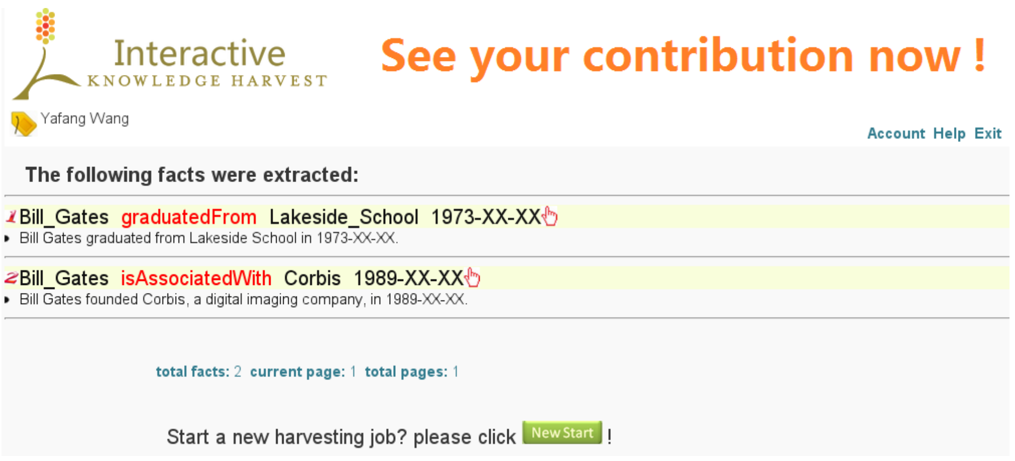
7 Bill_Gates isAssociatedWith Corbis TIME

8 Bill_Gates bornIn Washington TIME

total facts: 8 current page: 1 total pages: 1 See the results for Temporal facts?

Start a new harvesting job? please click New Start !

Figure 6.10.: Extracted base fact observations.



The following facts were extracted:

1 Bill_Gates graduatedFrom Lakeside_School 1973-XX-XX
- Bill Gates graduated from Lakeside School in 1973-XX-XX.

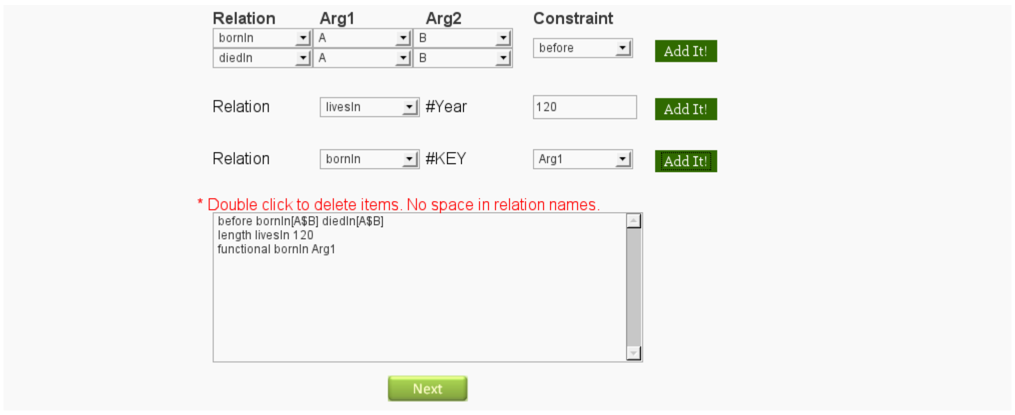
2 Bill_Gates isAssociatedWith Corbis 1989-XX-XX
- Bill Gates founded Corbis, a digital imaging company, in 1989-XX-XX.

total facts: 2 current page: 1 total pages: 1

Start a new harvesting job? please click New Start !

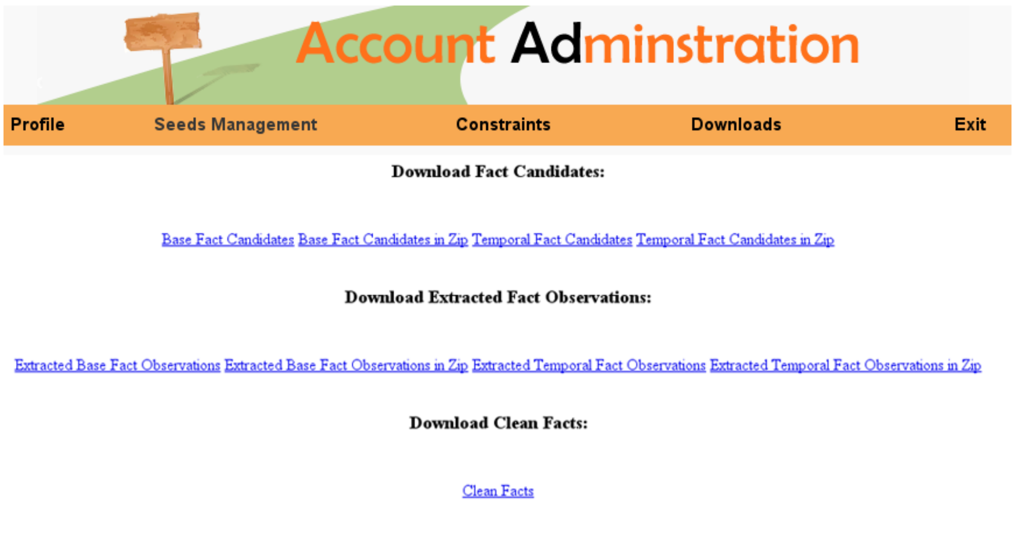
Figure 6.11.: Extracted temporal fact observations.

labelled patterns from pattern analysis (the "0" ones), are used in the *fact observation extraction* stage.



before bornIn[A$B] diedIn[A$B]
length livesIn 120
functional bornIn Arg1

Figure 6.12.: Add constraints.



Download Fact Candidates:

Base Fact Candidates Base Fact Candidates in Zip Temporal Fact Candidates Temporal Fact Candidates in Zip

Download Extracted Fact Observations:

Extracted Base Fact Observations Extracted Base Fact Observations in Zip Extracted Temporal Fact Observations Extracted Temporal Fact Observations in Zip

Download Clean Facts:

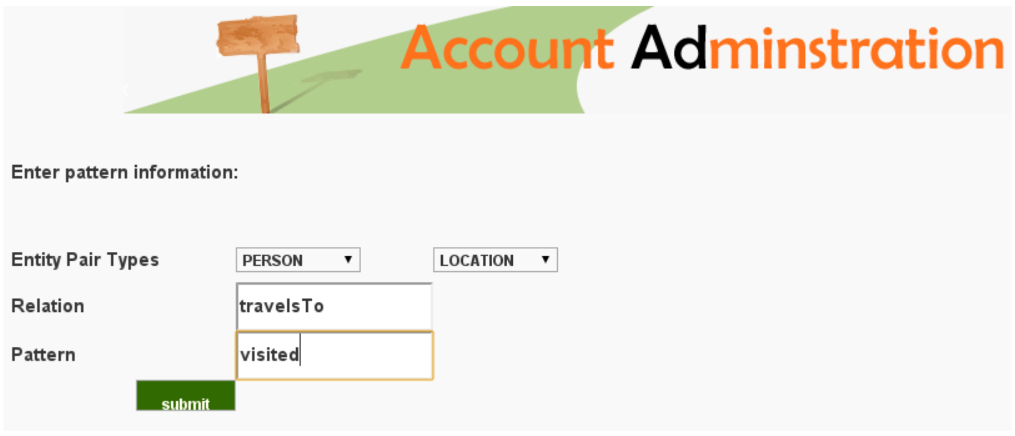
Clean Facts

Figure 6.13.: Downloads.

6.1.6. Demonstration Scenarios

In order to showcase the entire pipeline of our interactive knowledge harvesting system PRAVDA-live, we have prepared two dedicated demo scenarios: **Ad-hoc Fact Extraction for YAGO** and **Fact Extraction on Customized Relations**. Users may freely interact with our system.

Ad-hoc Fact Extraction for YAGO. In the first scenario we enable users to harvest (temporal) facts based on the relations supported by YAGO. To this end, users are able to either upload a document collection or paste a text

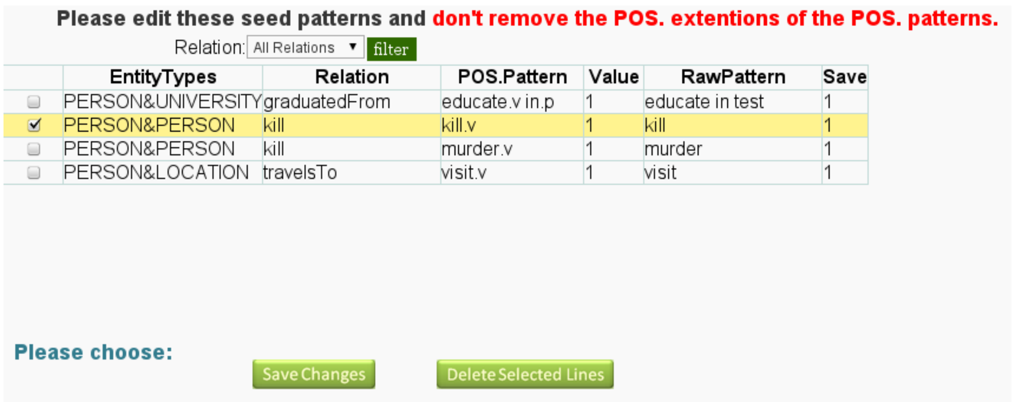


Account Adminstration

Enter pattern information:

Entity Pair Types	PERSON	LOCATION
Relation	travelsTo	
Pattern	visited	

Figure 6.14.: Add patterns.



Please edit these seed patterns and don't remove the POS. extentions of the POS. patterns.

Relation: All Relations filter

	EntityTypes	Relation	POS.Pattern	Value	RawPattern	Save
	PERSON&UNIVERSITY	graduatedFrom	educate.v in.p	1	educate in test	1
✓	PERSON&PERSON	kill	kill.v	1	kill	1
	PERSON&PERSON	kill	murder.v	1	murder	1
	PERSON&LOCATION	travelsTo	visit.v	1	visit	1

Please choose:

Figure 6.15.: Edit patterns.

document in the user interface of PRAVDA-live for a subsequent fact extraction (see Figure 6.3). After that, the facts can be evaluated via the assessment interface (see Figure 6.10 and Figure 6.11). Finally, the so created fact set can be exported as an RDF document (see Figure 6.13).

Fact Extraction on Customized Relations. The second scenario allows users to harvest facts from customized relations. This use case is of particular interest for those, who want inject RDF exports from PRAVDA-live into a proprietary knowledge base. To this end, we demonstrate the specification of additional relations (see Figure 6.5) and the corresponding labeling of a small number seed facts (see Figure 6.6 and Figure 6.7). Also users can manually add patterns for any relation of interest (see Figure 6.14). Further, we showcase the bulk processing feature of PRAVDA-live with a subsequent assessment. The demo concludes with a RDF export of the extracted facts (see Figure 6.13).

Chapter 7.

Applications

7.1. Time Aware Querying of Temporal Knowledge

7.1.1. Query Processing

From the work presented in Chapter 4 and Chapter 5, we build a temporal knowledge base called Timely YAGO (T-YAGO). In this section, we provide a time-aware query language for T-YAGO on its knowledge base of temporal facts. Recall that facts are represented as subject-property-object triples, SPO triples for short, of the RDF data model. Conditions on S, P, or O are expressed by SPARQL triple patterns, and can be combined in a conjunctive manner. For temporal conditions we have extended the query language by a suite of time predicates that refer to the *on*, *since*, and *until* relations of temporal facts: *before*, *after*, *equal*, *during*, *overlaps*, *sameYear*, and a few more. Each of these predicates takes as input two time points or two time periods or a time point and a time period, and returns a Boolean value. As T-YAGO's notion of validity times refers to fact identifiers, we need to be able to have variables for fact identifiers and also variables that denote time points for which we need to compute appropriate bindings. As an example, consider a query about teammates of David Beckham - soccer players who played for the same club as Beckham during overlapping periods. We can express this in the T-YAGO query language as follows:

?id1 : "David Beckham" *playsForClub* ?x .
?id2 : ?a *playsForClub* ?x .
?id1 *since* ?t1 . ?id1 *until* ?t2 .
?id2 *since* ?t3 . ?id2 *until* ?t4 .

[?t1 − ?t2] *overlaps* [?t3 − ?t4] .

?a *notEqual* "David Beckham"

where [?t1 − ?t2] denotes a time interval. The query returns important players such as Paul Scholes, Gary Neville, and Ruud van Nistelrooy at Manchester United, and Zinedine Zidane, Luis Figo, and Ronaldo at Real Madrid. In the query, predicates like *overlaps* are used as if they were relations (or properties in RDF jargon). However, they are not necessarily materialized, but instead computed dynamically as needed. We call these virtual relations. As they are actually evaluated as run-time functions, it is easy to switch to relaxed-matching semantics as an alternative to exact-matching evaluation of time predicates. The rationale for this option is that we may have different time resolution or uncertainty in the validity times of different facts. For example, consider a query about politicians who visited the same city on the same day. We may know that one politician visited Rome on May 21, and that another politician visited Rome in May of the same year. These time points do not match exactly because of the different resolutions. But we should still consider them as equal in a relaxed-matching mode. For such cases, T-YAGO supports relaxed-matching variants of all temporal predicates. These are based on the [earliest, latest] representation of uncertain time points. Two time points with uncertainty are considered equal if there is a non-zero probability that they are truly equal if they were exactly known. Figure 7.1 illustrates the matching of time points with [earliest, latest] uncertainty in a graphical manner: while the time points t1 and t2 are not equal in the strict sense they should be regarded as potentially equal applying our relaxed matching scheme. The other temporal predicates like overlaps, during, etc., are handled analogously.

t1[1999 − 01 − 01, 1999 − 06 − 30]   t2[1999 − 03 − 01, 1999 − 12 − 31]

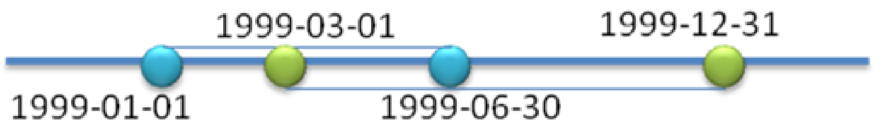
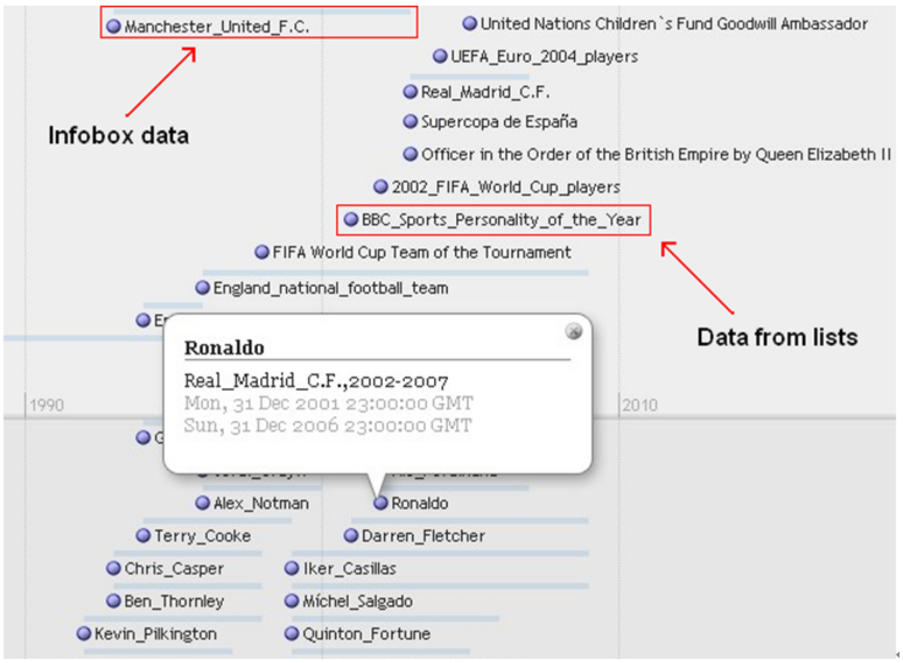


Figure 7.1.: Relaxed matching of time points.

7.1.2. Use Case Scenario

At this point, T-YAGO contains around 200.000 temporal facts from the sports domain. Among them, about 70.000 facts have been extracted from Wikipedia categories and lists embedded in articles. All these temporal facts have been integrated into the existing YAGO knowledge base. This way, we can demonstrate our querying capabilities for temporal facts in our prototype system.


Manchester_United_F.C.
United Nations Children`s Fund Goodwill Ambassador
UEFA_Euro_2004_players
Infobox data
Real_Madrid_C.F.
Supercopa de España
Officer in the Order of the British Empire by Queen Elizabeth II
2002_FIFA_World_Cup_players
BBC_Sports_Personality_of_the_Year
FIFA World Cup Team of the Tournament
England_national_football_team
Data from lists
1990
2010
Alex_Notman
Ronaldo
Terry_Cooke
Darren_Fletcher
Chris_Casper
Iker_Casillas
Ben_Thornley
Míchel_Salgado
Kevin_Pilkington
Quinton_Fortune

Ronaldo

Real_Madrid_C.F.,2002-2007
Mon, 31 Dec 2001 23:00:00 GMT
Sun, 31 Dec 2006 23:00:00 GMT

Figure 7.2.: The timeline of David Beckham.

To present the temporal facts, we implemented our demonstration based on SIMILE Timeline [1] which is a DHTML-based AJAX widget for visualizing time-based events. In our demo, we support several types of temporal queries. For example, users can query temporal facts for a person. Figure 7.2 is a snapshot segment of the querying result for "David Beckham". A point means a fact valid only on a time point like "`David_Beckham` *`isBornOn`* `1975`". A timespan denotes a fact valid in a time interval, e.g. "David Beckham *playsForSeniorClub* from 1993 to 2003". The users can navigate all facts along the whole timeline and click on anyone of them for the details. Above the timeline

[1]SIMILE Timeline, `http://www.simile-widgets.org/timeline/`

are the temporal facts concerning "David Beckham" himself, such as the awards he gained and his duration of time playing for different clubs. Below the timeline, there are the results for the query in Subsection 7.1.1 "who is the teammate of David Beckham?" We can click on each player's name and check his career information when David Beckham was at the same team with him. With the visualization, we can easily identify the related facts if they are in the same or the overlapping timespans.

7.2. Summarization

The recent dramatic growth of the Internet has attracted much attention on document summarization methods. However, most traditional techniques do not consider the semantic representation of a document, thus missing out the opportunities to address issues such as redundancy and incoherence of the resulting summaries. Prior approaches typically do not address specific users' interests, when a user seeks fine grained knowledge about a particular target entity. This section develops a novel method for abstractive summarization. Our method distills temporal knowledge from documents and generates a concise summary according to a particular user's interest, for example, focusing on a soccer player's career. Experiments are conducted on biography-style Wikipedia pages, and the results demonstrate the performance of our system compared to existing, both abstractive and extractive, summarization methods.

7.2.1. Problem Statement and Contributions

Document summarization denotes the process of generating a compressed version of one or more input documents that reflects the most important content of these documents in a condensed form. Automatic document summarization techniques have attracted a lot of attention recently [14, 26, 78, 22, 42]. The task of document summarization contains two main approaches: *extractive* and *abstractive* summarizations [27]. Usually extractive summarization methods generate the summary by extracting and reproducing entire sentences from the source documents, whereas abstractive document summarization techniques produce a grammatical summary by advanced language generation techniques [18]. Since abstractive summarization is generally found to be difficult, most existing summarization strategies focus on extractive summarization. However, one serious problem for most extractive methods is the presence of redundant information that is often formulated in slight variations in the natural-language input. To ensure a reasonable trade-off between summary size and information content, these systems rank sentences by various informativeness and redundancy measures and then select the top-ranked sentences for the summary. If sentences are very similar at the word level, but rather dissimilar in the content they reflect, this may lead to many false removals of candidate sentences from the summary. For example, the following three, syntactically very similar sentences

- “Beckham transferred to Real Madrid”,
- “Beckham moved to Real Madrid”, and
- “Ronaldo moved to Real Madrid”

might be arranged in the result summary in descending order of weight. An overly eager removal of syntactically similar sentences might thus lead to an erroneous removal of the last two sentences. In fact, only the second sentence is redundant in this case, while the third one is not. Other issues, such as bad readability and incoherence of sentences in the resulting summaries, are also longstanding difficulties of extractive summarization approaches.

Another challenging problem in document summarization is the relevance of the summary to a particular user’s intent. Often, the summary does not reflect the specific information that is requested by a user (along with some relevant context information), especially when a user seeks a rather specific piece of information. Topic-based summarization strategies have been proposed to address this issue; however, in many cases users do not just search for a predefined topic, but ask for the summary related to more fine grained knowledge, e.g., a soccer players’ family life. This would call for a combination of information extraction, information retrieval, and knowledge-based representation techniques for a specific and concise form of document summarization.

To solve these challenges, our goal is to provide an abstractive summarization method that is capable of identifying the key factual knowledge provided by a natural-language input document with similar precision and recall as a human, and to aggregate and represent this information in a natural, non-redundant way. What, for example, should a short summary (of, say, 100 words) for a soccer player's career include? For a famous player, it will be quite impossible to list all the clubs and games the player was associated with during his or her entire career. After reading, for example, the Wikipedia article about the player, a human could easily provide the total number of clubs in which the player served, as well as the number of honors the player won. Reporting these numbers in a summary could help a reader to know the popularity of the player. The next step would be to identify the most important clubs and honors—in chronological order—and to add these to the summary. While identifying and aggregating the key facts of a document is not difficult for a human, this certainly remains a challenge for any automated summarization approach.

To this end, we argue that the recent advent in information extraction and knowledge harvesting, and in particular the extraction of temporal knowledge, may come to our rescue.

Contributions. In this section, we propose a method that, similarly to a human, distills factual temporal knowledge from input documents and generates concise summaries from these facts directly via templates for natural-language output sentences. We summarize our contributions:

- a new abstractive approach for multi-document summarization from both semi-structured and natural-language input contents,
- an algorithm for reordering the extracted temporal knowledge and for presenting this knowledge as natural language text (see Section 7.2.4),
- a method for aggregating potentially noisy pieces of evidence into coherent, high-confidence time intervals for facts (see Section 7.2.3),
- an experimental study with Wikipedia biographies demonstrating the effectiveness of our knowledge-based summarization approach in comparison to existing baseline systems for extractive and abstractive summarization (see Section 7.2.5).

7.2.2. System Architecture

Given a set of input documents $\mathcal{D} = \{D_1, D_2, \ldots, D_n\}$ and a set of typed target relations $\mathcal{R} = \{R_1, R_2, \ldots, R_m\}$, our system aims at distilling key factual knowledge (i.e., instances of the target relations in $\mathcal{R}$) from different kinds of data formats that are captured in $\mathcal{D}$. Our approach is designed to follow the way a human would summarize a document, by first digesting the document and then by representing and rearranging the interesting knowledge and associating the different facts occurring in the documents with each other.

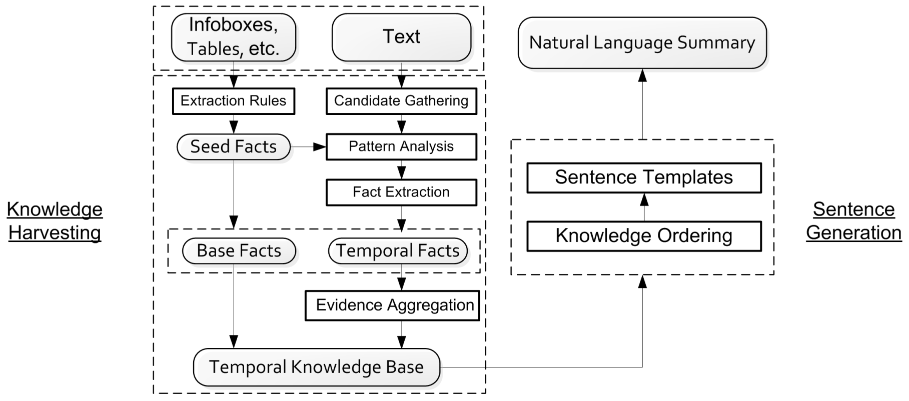


Figure 7.3.: System architecture.

Figure 7.3 depicts an overview of our approach. Knowledge harvesting and evidence aggregation serve as the digesting and representation step, while knowledge ordering and sentence generation serve to arrange and present the key facts in the form of a coherent, natural-language summary to the user. To harvest interesting knowledge from both semi-structured and textual data sources, we employ the general architecture of PRAVDA (including candidate gathering, pattern analysis and fact extraction) to extract new facts (both base and temporal facts) from free-text. Multiple occurrences of temporal facts are reconciled by a form of evidence aggregation, which serves to condense the extracted knowledge, and to extract high-confidence time intervals at which these new facts are found to be valid. Finally, for better readability and coherence of the final summary, these facts along with their time intervals are ordered chronologically and presented as natural language sentences by mapping the facts onto a set of handcrafted sentence templates. For details of knowledge harvesting, we refer the reader to Chapter 4. All the other details for each of these steps are provided in the following sections.

7.2.3. Evidence Aggregation Model

In this section, we present two basic processing steps for aggregating time points attached to the event facts provided by knowledge harvesting into a concise time histogram of a corresponding state fact. These aggregation steps include (1) the aggregation of event facts with individual time points into state facts with a single time histogram, and (2) the distillation of high-confidence time intervals out of these histograms. Please notice that the aggregation algorithm presented in Section 5.3.2 could only cope with unimodal time histograms, whereas the aggregation approach in this section also supports multimodal histograms.

Aggregating Events into State Histograms

Among all observations of event facts found in input sentences by knowledge harvesting step, we first determine the time range $[t_b, t_e]$ of the largest possible validity interval of a corresponding state fact by selecting the earliest time point t_b and the latest time point t_e out of these event facts, respectively. According to the relation an event fact has been labelled with, we refine the individual event facts into *begin*, *end*, and *during* observations that mark either the possible begin or end time point, or a time point during which the corresponding state fact may be valid.

Next, all observations of *begin*, *end*, and *during* events are aggregated into three initial histograms, each ranging over $[t_b, t_e]$. It yields one frequency value *freq*$[t_i]$ per time point t_i. Initially, the i^{th} bin's frequency value *freq*$[t_i]$ refers to the plain number of observations corresponding to this time point, for each of the three types of event facts. Subsequent time points with equal frequencies are coalesced into a single histogram bin. In each of the histograms, the bins' frequencies are then normalized to 1. For combining the three event-oriented histograms into a single histogram of the corresponding state fact, we apply the following assumptions:

- A *during* observation at time point t_j should increase the confidence in the state fact being correct at t_j (for all time points captured by the interval of the *during* observation).

- A *begin* observation at time point t_j should increase the confidence in the state fact for all time points ranging from t_j to t_e.
- An *end* observation at time point t_j should decrease the confidence in the state fact for all time points t_j to t_e.

Algorithm 5 shows pseudo-code for combining the *begin*, *end* and *during* histograms. We first merge the two *begin* and *end* histograms, before we merge the resulting *begin-end* histogram with the *during* histogram using Equation 7.1 (based on De Morgan's law).

$$\begin{aligned} P = P_{\text{during}} \cup P_{\text{begin,end}} &= \overline{\overline{P_{\text{during}}} \bigcap \overline{P_{\text{begin,end}}}} \\ &= 1 - (1 - P_{\text{during}}) \cdot (1 - P_{\text{begin,end}}) \end{aligned} \tag{7.1}$$

For this, let P denote the final frequency obtained after all aggregation steps, let P_{during} denote the frequency of the *during* event, and let $P_{\text{begin,end}}$ be the output (i.e., $\texttt{freq}[t_i]$ after the inner *for* loop in Algorithm 5) of aggregating the *begin* and *end* histograms. For all the non-empty bins in the *during* histogram, we use Equation 7.1 to compute the new frequency value P. Finally, all consecutive bins with the same frequency values are merged, and the bins are once more normalized to 1 (cf. Algorithm 5 and Figure 7.4).

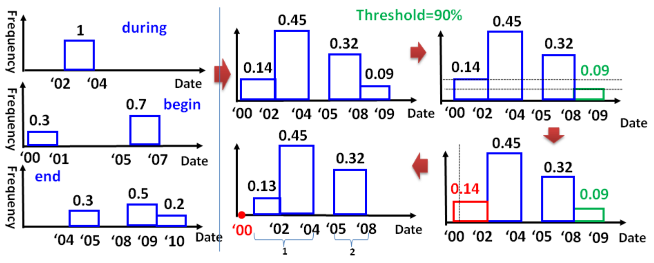


Figure 7.4.: Aggregating events into state histograms.

Extracting High-Confidence Intervals

For the final temporal knowledge base, we aim at further simplifying the potentially very fine-grained histogram we have obtained from the previous

Algorithm 5 Aggregating Events into State Histograms.

```
Require: Event histograms with frequencies $freq_{begin}$, $freq_{end}$, $freq_{during}$ over the time range $[t_b, t_e]$
1: for each $t_i \in [t_b, t_e]$ do
2:     Set $freq[t_i] = 0$
3:     // Aggregate begin histogram with end histogram
4:     for each $t_j \in [t_i, t_e]$ do
5:         Set $freq[t_j] = freq[t_j] + freq_{begin}[t_i]$   //aggregate begin
6:         Set $freq[t_j] = max(0, freq[t_j] - freq_{end}[t_i])$   //reduce end
7:     end for
8:     // Combine merged begin,end histogram with during histogram
9:     Set $freq[t_i] = (1 - (1 - freq_{during}[t_i]) \cdot (1 - freq[t_i]))$
10: end for
11: Reorganize the bins and normalize their frequencies to 1
12:
13: return State histogram with frequencies freq
```

aggregation step by discarding bins with a low confidence. Assuming, for example, we are interested in a final histogram that captures at least 90 percent of the confidence mass of the original histogram, we discard all low-confidence bins whose cumulative frequencies sum up to at most 10 percent.

Since the original histogram's bins form a discrete confidence distribution, we pursue an iterative algorithm. Starting from the lowest-frequency bin, we first sort all bins by their frequency values and then check for the remaining confidence mass when cutting off these bins horizontally. Let τ be the expected threshold of the confidence interval (e.g., 90 percent). Our algorithm stops as soon as we have cut off more than a threshold of $1 - \tau$ (e.g., 10 percent) of the overall confidence mass. We then pick the previous solution, which must still be above τ. This procedure is further refined by a final vertical trimming step of the remaining bins. To this end, we assume a uniform confidence distribution within each bin, and we adjust the frequency value *freq*[i] of the trimmed bin proportionally to its cut-off width (cf. Figure 7.4) until we reach τ.

7.2.4. Sentence Generation and Reordering

Most extractive summarization methods return summary sentences in the order in which these sentences are presented in the original articles. We argue that a *chronological* order of facts is a more natural way to present more concise summaries (e.g., biographies), that allows us to further abstract the summary contents from the contents of the original input documents. The extraction of temporal facts of course strongly facilitates this task.

Knowledge Ordering

Before sorting the individual facts about an entity of interest, we first roughly sort the more abstract relations associated with t-facts. Some relations can be very naturally ordered. Considering a person's life, for example, the time point of a t-fact for the *isBornIn* relation must lie before the start point of a *isMarriedTo* t-fact for the same person, which in turn must lie before the time point of a *diedIn* t-fact of that person.

This order of relations can be learned statistically from t-facts. Given a set of relations $\mathcal{R}$ and their temporal instances (t-facts), we aim at building a time-ordered directed graph $G = (V, E)$, where each vertex refers to a relation and each edge represents a chronological dependency. We start by creating an initial graph $G' = (\mathcal{R}, E)$ by adding an edge (R_i *before* R_j) if the support s_{ij} of (R_i *before* R_j) is much greater than s_{ji}. s_{ij} is calculated by counting the instances of R_i and R_j having the same subject, that is $r_i(a, b)$ and $r_j(a, c)$, satisfying ($r_i(a, b)$ *before* $r_j(a, c)$). The final graph G is then obtained from G' by adding two virtual vertices representing two *start* and *end* states to G' and by removing all transitive dependencies from G'. For example, *isBornIn* may have an edge with many relations, such as *graduatedFromHighSchool*, *graduatedFromUniversity* and *diedIn*. These edges are removed according to the transitive dependencies among these relations, and only a path from *isBornIn* through *graduatedFromHighSchool*, *graduatedFromUniversity* to *diedIn* is kept. If the graph G contains a cycle, we remove the cycle by dropping the edge with the lowest support within the cycle. Figure 7.5 illustrates an example for transforming a set of relations into G. Algorithm 6 shows details about how to determine the chronological order of both t-facts and b-facts according to G. The algorithm is based on topological sorting. Regarding a state fact, which is valid during an

Algorithm 6 Knowledge Ordering

```
Require: Graph $G$; the base and temporal facts $F_b$ and $F_t$.
 1: $S = \emptyset$  //Empty list that will contain the sorted facts.
 2: $L \Longleftarrow$ Set of all vertices with no incoming edges.
 3: while $L$ is non-empty do
 4:    remove a vertex $n$ from $L$
 5:    if $n$ has not been visited yet then
 6:       insert all the t-facts(in time-order) and then b-facts of relation $n$ into $S$
 7:    end if
 8:    for each node $m$ with an edge $e$ from $n$ to $m$ do
 9:       remove edge $e$ from the graph $G$
10:       if $m$ has no other incoming edges then
11:          insert $m$ into $L$ and mark $m$ as visited
12:          sort all the t-facts of relation $m$ by time and insert them into $S$
13:          insert b-facts of relation $m$ into $S$
14:       end if
15:    end for
16: end while
17:
18: return $S$  //Facts sorted by topological order of relations in $G$.
```

entire time interval, only the start time point is taken into consideration. For example, suppose we captured that David_Beckham played for Manchester_United from 1991 to 2003, Real_Madrid from 2003 to 2007 and got married on 4-July-1999. The three temporal facts are ordered as {David_Beckham *playsForClub* Manchester_United, David_Beckham *getsMarriedWith* Victoria_Beckham, David_Beckham *playsForClub* Real_Madrid}, according to the time points {1-January-1991, 4-July-1999, and 1-January-2003}. Base facts (b-facts) which generally cannot be ordered explicitly by time are inserted into G after the temporal facts (t-facts) in the same relation according to the topological order (Line 13).

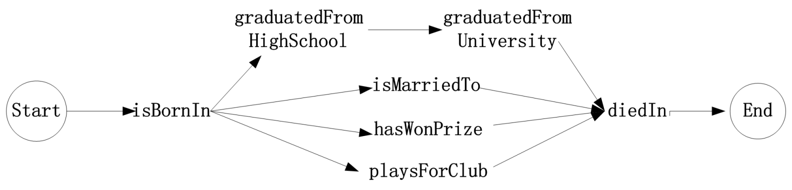


Figure 7.5.: Relation graph.

Sentence Generation and Redundancy Elimination

For each relation, we manually define sentence templates to construct the summary sentences. These templates aim at providing easy-to-read summaries also for non-native speakers. We depict some example templates used in our system in Table 7.1: These are the templates for base facts.

Relation	**Templates**
isBornIn	ARG1 was born in ARG2
worksForClub	ARG1 served for ARG2; ARG1 worked for ARG2
actedIn	ARG1 acted in ARG2; ARG1 appeared in ARG2
hasWonHonor	ARG1 has won ARG2; ARG1 received ARG2

Table 7.1.: Example sentence templates for relations.

For temporal facts these templates are additionally associated with a place holder for a time point or interval. For example, the template for temporal facts of *isBornIn* is "ARG1 was born in ARG2 on TIME". The template for temporal facts of *worksForClub* containing single time interval is "ARG1 served for ARG2 from begin_TIME to end_TIME";and for multiple time intervals is "ARG1 served for ARG2 (begin_TIME*1*–end_TIME*1*, begin_TIME*2*–end_TIME*2*, ..., begin_TIME*n*–end_TIME*n*)".

Specifically, we create a number of different templates for each relation and randomly choose from them for each subject to improve the diversity of the output. After the knowledge ordering, t-facts of the same relation are ordered next to each other due to the topological order in the relation graph. In terms of templates, sentences representing the same relation are likely to contain a lot of redundancy. Thus we propose two steps to merge redundant sentences into one. For example, "David Beckham played for Manchester United from 1993 to 2003” and “David Beckham played for Real Madrid from 2003 to 2007” are merged into “David Beckham played for Manchester United (1993-2003) and Real Madrid (2003-2007)”. For his honors, we could similarly get the following merged sentence “David Beckham won the Premier League (1996), the FA Cup (1999), the UEFA Champions League (1999), the Intercontinental Cup (1999), and the La Liga (2007), etc.” Since there are many honors, we resort to only show the first ones. Such summaries correspond to the *long summary* in the experiments section (see Section 7.2.5, *baseline systems*).

In case there are too many facts holding the same relation, we could omit some unimportant facts, report the total number, and choose only some examples for the summary sentences. Considering once more Beckham’s career, if the importance of a career depends on the number of honors a player has won during this time period, we could keep Manchester United and omit all the others. The number of honors in each club is not difficult to get by comparing the time point when at which the player won the honor with the time period during which he played for the club. Similarly, the repeated occurrence of the full name “David Beckham” can be replaced by the corresponding pronoun “he”. Hence the final summary is compressed into ”David Beckham has played for about eight clubs. He joined Manchester United in 1993. During his career in Manchester United, he won about fifteen honors including the Premier League (1996), the FA Cup (1999), etc.”. The initially redundant sentences were thus compressed into just three sentences with the key facts about David Beckham. Such summary is called *short summary* in Section 7.2.5 (see *baseline systems*).

7.2.5. Experiments

Experimental Setup

Data Sets: We evaluate our method on Wikipedia articles from two domains: soccer players and movie stars. The corpora include Wikipedia articles for soccer players from the “FIFA 100 list”[2], and movie stars from the “Top 100 movies stars”[3]. The gender of the person was heuristically deterimined by the most

[2]http://en.wikipedia.org/wiki/FIFA_100/

[3]http://articles.cnn.com/2003-05-06/entertainment/movie.poll.100_1_star-movies-godfather?_s=PM:SHOWBIZ/

frequent pronoun in the Wikipedia article. Thus we preprocessed the corpora by replacing the most frequent pronoun by the title of the Wikipedia article and all the entity mentions were disambiguated against the YAGO [62] knowledge base using the AIDA [30] framework for named entity disambiguation.

Queries: For both domains, we query the system for summaries about facts associated with the birth and death dates of the respective persons, their family life (including marriage and children), honors they won, and their career (including the relations *worksForClub* for soccer players or *actedIn* for movie stars, as well as playing positions for soccer players).

Baseline Systems: According to Section 7.2.4, we conduct two experiments for our system. 1) We generate the summary with most of the facts about a person (about 200 words), and 2) we generate a summary with only the most important facts and aggregated statistics (about 100 words). We call the results from each experiment a *long summary* and a *short summary*, respectively.

We compare our system to three different approaches introduced in the literature. Since the top paragraphs in a Wikipedia article are usually a small biography of the subject of the article, we extracts the first n sentences from a Wikipedia article. Such method is called **NIST-Wiki**.

LDA is a generative probabilistic model used for discovering "latent" topics based on a hierarchical Baysian model [9]. **LDA**-based document summarization has been applied to document summarization in 2008 [2] and uses probabilistic topic distributions to calculate the salience for each input sentence.

Additionally, as a representative model for recent abstractive summarization methods, we use Opinosis as another baseline. **Opinosis** [26] is a graph-based abstractive summarization framework. It constructs a graph from a set of input sentences set by considering redundancy and generates an optimal path from the graph.

Since these baseline systems can only consider textual input data, the semi-structured input data (such as infoboxes) are translated to natural-language sentences by sentence templates, in order to make the input equal for both our system and the baseline systems. For example, all the honors of Beckham are translated into "David Beckham has won Premier League in

1996. David Beckham has won Premier League in 1997. David Beckham has won FIFA 100. ...”. For the short summary, we limit the number of words to about 100; and to 200 for the long summaries. Since Opinosis is limited by the number of sentences, a short summary is limited to 10 sentences, while a long summary contains 20 sentences.

Evaluation Procedures: We evaluate the summary for *informativeness*, *diversity*, *coherence* and *precision* [44, 42] by performing a user study. There are four participants. Each participant randomly samples thirty summaries for each domain. For each of the above metrics, we rate the summary from one to five. With respect to the four metrics, a rating of *one* means {“least informative”, “least diverse”, “very incoherent”, “very imprecise”}; while a rating of *five* means {“very informative”, “very diverse”, “very coherent”, “very precise”}. The final score of each metric then is the average of all 4×30 summaries. The score of each metric in Table 7.4 is the average of all 4×60 summaries on both domains including the short and long summary. The overall score then is the average over all four metrics for each system on both short and long summaries.

Example Summaries

In order to better compare the results, we depict a few of the generated summaries about David Beckham in this subsection. Due to the limited space, we only show some running examples for our system (including long and short summaries but with similar size), **NIST-Wiki**, **LDA**, **Opinosis**, plus a manually created summary as additional reference.

Our System(long): *David Beckham was born in London in 1975/05/02. He played as Midfielder. He served in Manchester United F.C. (1991-2003), Real Madrid C.F. (2003-2007), Los Angeles Galaxy (2007-). He has won FA Youth Cup (1992), FA Community Shield (1993,1994,1996,1997), Premier League (1996,1997,1999,2000,2001,2003), FA Cup (1996,1999), UEFA Club Football Awards (1999), MLS Cup (2011), FIFA 100, etc.*

Our System(short): *David Beckham was born in London in 1975/05/02. He played as Midfielder. He has played for about 3 clubs and won about 45 honors. In 1991 he joined Manchester United F.C. and served for 13 years. During this time period, he was awarded FA Youth Cup (1992), FA Cup (1996), FA Cup (1999), Intercontinental Cup (1999), UEFA Club Football Awards (1999), etc.*

NIST-Wiki: *David Robert Joseph Beckham, Order of the British Empire (born 1975-05-02) is an England association footballer who plays for Los Angeles Galaxy. David Beckham has played for Manchester United F.C., Preston North End F.C., Real Madrid C.F., A.C. Milan, and the England national football team for which David Beckham holds the appearance record for a Outfield#Association football. David Beckham's career began when David Beckham signed a professional contract with Manchester United, making his first-team debut in 1992 aged 17.*

LDA: *Beckham scored the equaliser and United went on to win the match and the league. Beckham scored 9 goals that season, all in the Premier League. The income from his new contract, and his many endorsement deals, made Beckham the highest-paid player in the world at the time. In the first nine matches David Beckham started, Real lost 7. David Beckham returned to play in the final home match of the season. Beckham is Officers of the Order of the British Empire. Beckham is England expatriates in the United States.*

Opinosis: *David Beckham enjoyed tremendous following. Beckham's right midfield position. Beckham's contract became public knowledge. Beckham's maternal grandfather was Jewish. Beckham's best season as united player and united. Beckham is England under-21 international footballers. Beckham England people of Jewish descent. Beckham's marriage in 2007- -/:. Beckham crumpled hard to the ground. Beckham of the most recognisable athletes throughout the world, not concentrating on the tournament and England 's next match.*

Manual: *David Beckham, born in 2 May, 1975, is a midfielder. Beckham began his career with Manchester United in 1991. During his 13 years career there, he won several honors. He received Premier League 10 Seasons Awards for his contribution from the 1992-93 to 2001-02 seasons. He also played for Real Madrid, LA Galaxy, etc. To honor his contribution, he was named FIFA 100. On 4 July 1999, David married Victoria. They have four children: sons Brooklyn Joseph, Romeo James, and Cruz David; and daughter Harper Seven.*

System	**Diversity**	**Informativeness**	**Coherence**	**Precision**	
Ours	3.97	4.56	4.48	4.58	*Soccer*
NIST-Wiki	3.10	3.71	4.83	4.88	
LDA	3.38	4.06	3.54	4.61	
Opinosis	2.28	3.90	1.95	3.17	
Ours	3.48	4.78	4.13	4.68	*Movie Star*
NIST-Wiki	2.25	3.64	4.34	4.99	
LDA	1.98	3.33	2.26	4.79	
Opinosis	1.47	3.03	1.76	3.48	

Table 7.2.: Long summary.

System	**Diversity**	**Informativeness**	**Coherence**	**Precision**	
Ours	3.64	4.06	4.25	4.22	*Soccer*
NIST-Wiki	2.80	3.00	4.78	4.83	
LDA	2.93	3.55	3.28	4.46	
Opinosis	2.01	3.22	1.81	3.03	
Ours	3.35	4.53	4.37	4.11	*Movie Star*
NIST-Wiki	1.93	3.18	4.33	4.98	
LDA	1.58	2.75	2.23	4.79	
Opinosis	1.33	2.61	1.68	3.40	

Table 7.3.: Short summary.

System	**Diversity**	**Informativeness**	**Coherence**	**Precision**	**Overall**
Ours	**3.61**	**4.48**	4.30	4.40	**4.20**
NIST-Wiki	2.52	3.38	**4.57**	**4.92**	3.85
LDA	2.46	3.42	2.83	4.66	3.34
Opinosis	1.77	3.19	1.80	3.27	2.51

Table 7.4.: Overall score.

Experimental Results

The evaluation results from two experiments on the four metrics are displayed in Tables 7.2 and 7.3. On the average over all four metrics, our system outperforms all baseline systems (see the last column of Table 7.4). More specifically, our system outperforms all the others on diversity and informativeness, but comes slightly shorter on precision and coherence. In contrast, NIST-Wiki is the best on these two metrics, because it just picks the first n sentences from each Wikipedia article which is very similar to a manual short summary.

There is no 100% perfect extraction methodology. Incorrect extractions obviously affect the precision of the generated summary. Furthermore, extraction recall also influences the short summary since we report a statistical number. For example, if we only extract one award of David Beckham, the short summary "David Beckham won about one honor" is incorrect. This error may only happen for extractive summaries when the source data contains mistakes. To reduce such errors, we might consider including also vague statements like "more than". Opinosis compresses the text by considering the sentence redundancy, so the new generated sentences may change the semantics of the original sentences. This holds for semi-structured contents which are presented as natural-language sentences, such as "Rui Costa has won Toulon Tournament in 1992. Rui Costa has won FIFA U-20 World Cup in 1991." Opinosis is able to compress them into one meaningful sentence "Rui Costa has won Toulon Tournament in 1992 and FIFA U-20 World Cup in 1991." While for other sentences in the Wikipedia article, most generated sentences are meaningless and even incorrect. See the running examples of David Beckham in the last subsection. Opinosis generates one sentence "Beckham's marriage in 2007- -/:." However Beckham actually got married with Victoria on 1999-July-04. Also, the sentence "David Beckham enjoyed tremendous following" is meaningless. Such a sentence is given a score of 3. So the overall precision of Opinosis is about 3. Noisy text, such as URL links which are recognized as sentences, is accounted as an error for precision during the evaluation. This is also the reason why the precision of NIST-Wiki is not 100% correct.

NIST-Wiki produces perfect coherence as it returns the contiguous first n sentences from the Wikipedia article. Other extractive methods, like LDA, introduce incoherence into the summary, as the extracted sentences are usually not contiguous. Besides incoherence, they also introduce imprecision when the extracted sentence contains indicative pronouns, such as "after this", and temporal phrases, such as "one year later". If the original previous sentence was not extracted, the extracted sentence would indicate incorrect information. As for the abstractive method, some sentences generated by Opinosis are meaningless, therefore it increases the difficulty of reading the summary. On the contrary, our system exploits simple templates which are easy to understand. Only when too many facts hold the same relation, the generated sentence feels non-fluent. For example, "David Beckham won the Primier League (1996,1997,1999,2000,2001,2003), FA Cup (1996), La Liga (2007), MLS Cup (2011) ...". We currently use a parameter to show only a fixed number of repeated facts.

Notice also that the informativeness and diversity are affected by recall. Our system managed to find the key information. The sentences from the semi-structured input contents facilitate LDA and Opinosis to find this key information as well. Specifically for LDA, those sentences get higher topic saliency than other sentences from the free-text contents in the article for each topic, thus LDA could extract more information from those structured sentences into the final summary. Therefore the score of LDA and Opinosis for informativeness is better than or close to NIST-Wiki (the natural biography), according to Table 7.4. The diversity is not good across all systems. No system managed to extract all information of interest. Looking at the last subsection of the running examples, no system extracted summaries about Beckham's marriage and children. Considering the honors, even if our system extracted all the honors for Beckham, it is difficult to decide which ones are the most important ones to be shown in the summary, since only an expert with strong background knowledge could find out the important honors. Since the LDA-based summarization strategy calculate the saliency in multiple topics, it could get different sentences focusing on different sub-topics for each article. Therefore, as shown in the results, for diversity the LDA-based method could get a score that is close to NIST-Wiki's score.

7.2.6. Conclusions and Outlook

This section has presented a novel abstractive summarization method by leveraging facts from a temporal knowledge base. The experimental results show that our approach may more adequately capture a user's interest and achieves high performance on summarization diversity and informativeness in comparison to other extractive and abstractive methods.

In our approach, the natural language sentences are generated by manually defined templates. Our current experiments mostly focused on a small number of relations over two specific domains. Future work could aim at designing an automatic approach for generating sentence templates for each relation and to generalize our techniques to broad and open domains.

Chapter 8.

Conclusions

Summary. This thesis has presented methods and tools for the automatic construction of temporal knowledge base. It consists of four contributions: the core temporal knowledge base T-YAGO (introduced in Chapter 4) and the knowledge harvesting system PRAVDA (introduced in Chapter 4 and 5) , the knowledge cleaning system T-URDF (introduced in Chapter 5) and the interactive knowledge harvesting system PRAVDA-live (introduced in Chapter 6). These assets, together, constitute a system for constructing, maintaining and expanding a temporal knowledge base. Finally applications of temporal knowledge are discussed in Chapter 7.

Outlook. There are several directions for future work.

1. Currently the system is designed for a single machine. The obvious next step is to extend the system for distributed platforms, aiming at scalable temporal knowledge harvesting.

2. Temporal expressions are recognized by regular expressions. Since it is impossible to create the regular expressions for all valid temporal expressions, only explicit temporal expressions (e.g., 21 December, 2012) are recognized. This could be extended to increase recall. In addition to explicit dates, the implicit temporal expressions, for example, the *French Revolution*, should be recognized as well.

3. Since we focus on closed-domain relation extraction, the relations are manually pre-specified in our system. In contrast, open-domain relation extraction aims at extracting all facts without any limitation of relations.

Methods could be developed to discover relations automatically by combining closed-domain and open-domain relation extraction.

Bibliography

[1] James F. Allen. Maintaining knowledge about temporal intervals. *Commun. ACM*, 26(11):832–843, November 1983.

[2] Rachit Arora and Balaraman Ravindran. Latent dirichlet allocation based multi-document summarization. In *Proceedings of the second workshop on Analytics for noisy unstructured text data*, AND '08, pages 91–97, New York, NY, USA, 2008. ACM.

[3] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. DBpedia: A nucleus for a web of open data. In *Proceedings of the 6th international The semantic web and 2nd Asian conference on Asian semantic web conference*, ISWC'07/ASWC'07, pages 722–735, Berlin, Heidelberg, 2007. Springer-Verlag.

[4] Shumeet Baluja, Rohan Seth, D. Sivakumar, Yushi Jing, Jay Yagnik, Shankar Kumar, Deepak Ravichandran, and Mohamed Aly. Video suggestion and discovery for YouTube: Taking random walks through the view graph. In *Proceedings of the 17th international conference on World Wide Web*, WWW '08, pages 895–904, New York, NY, USA, 2008. ACM.

[5] M. S. Bazaraa, H. D. Sherali, and C. M. Shetty. *Nonlinear programming: theory and algorithms*. Wiley-Interscience series in discrete mathematics and optimization. Wiley, 1993.

[6] Omar Benjelloun, Anish Das Sarma, Alon Halevy, Martin Theobald, and Jennifer Widom. Databases with uncertainty and lineage. *The VLDB Journal*, 17(2):243–264, March 2008.

[7] Klaus Berberich, Srikanta Bedathur, Omar Alonso, and Gerhard Weikum. A language modeling approach for temporal information needs. In

Proceedings of the 32nd European conference on Advances in Information Retrieval, ECIR '10, pages 13–25, Berlin, Heidelberg, 2010. Springer-Verlag.

[8] Barry Bishop and Florian Fischer. IRIS- Integrated rule inference system. In *Proceedings of the 1st Workshop on Advancing Reasoning on the Web: Scalability and Commonsense (ARea2008) hosted by the 5th European Semantic Web Conference (ESWC-08)*, ARea '08. CEUR Workshop Proceedings, 2008.

[9] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *The Journal of Machine Learning Research*, 3:993–1022, 2003.

[10] Branimir Boguraev, James Pustejovsky, Rie Kubota Ando, and Marc Verhagen. TimeBank evolution as a community resource for TimeML parsing. *Language Resources and Evaluation*, 41(1):91–115, 2007.

[11] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, SIGMOD '08, pages 1247–1250, New York, NY, USA, 2008. ACM.

[12] Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R. Hruschka Jr., and Tom M. Mitchell. Toward an architecture for never-ending language learning. In Maria Fox and David Poole, editors, *Proceedings of the 24th AAAI Conference on Artificial Intelligence*, AAAI '10, pages 1306–1313. Association for the Advancement of Artificial Intelligence, 2010.

[13] Andrew Carlson, Justin Betteridge, Richard C. Wang, Estevam R. Hruschka, Jr., and Tom M. Mitchell. Coupled semi-supervised learning for information extraction. In *Proceedings of the third ACM international conference on Web search and data mining*, WSDM '10, pages 101–110, New York, NY, USA, 2010. ACM.

[14] Asli Celikyilmaz and Dilek Hakkani-Tur. A hybrid hierarchical model for multi-document summarization. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 815–824, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[15] Jinxiu Chen, Donghong Ji, Chew Lim Tan, and Zhengyu Niu. Relation extraction using label propagation based semi-supervised learning. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, ACL-44, pages 129–136, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics.

[16] John M. Conroy and Dianne P. O'leary. Text summarization via hidden Markov models. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '01, pages 406–407, New York, NY, USA, 2001. ACM.

[17] Aron Culotta and Jeffrey Sorensen. Dependency tree kernels for relation extraction. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, ACL '04, pages 423–429, Stroudsburg, PA, USA, 2004. Association for Computational Linguistics.

[18] Dipanjan Das and AndrĺȩF. T. Martins. A survey on automatic text summarization. *Literature Survey for the Language and Statistics II course at CMU*, 4:192–195, 2007.

[19] Günes Erkan and Dragomir R. Radev. LexRank: graph-based lexical centrality as salience in text summarization. *J. Artif. Int. Res.*, 22(1):457–479, December 2004.

[20] Oren Etzioni, Michele Banko, Stephen Soderland, and Daniel S. Weld. Open information extraction from the web. *Commun. ACM*, 51(12):68–74, December 2008.

[21] Oren Etzioni, Michael Cafarella, Doug Downey, Stanley Kok, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S. Weld, and Alexander Yates. Web-scale information extraction in KnowItAll: (preliminary results). In *Proceedings of the 13th international conference on World Wide Web*, WWW '04, pages 100–110, New York, NY, USA, 2004. ACM.

[22] Katja Filippova. Multi-sentence compression: finding shortest paths in word graphs. In *Proceedings of the 23rd International Conference on*

Computational Linguistics, COLING '10, pages 322–330, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[23] Michael Fisher, Dov Gabbay, and Lluis Vila. *Handbook of Temporal Reasoning in Artificial Intelligence (Foundations of Artificial Intelligence (Elsevier))*. Elsevier Science Inc., New York, NY, USA, 2005.

[24] Norbert Fuhr. Probabilistic Datalog-a logic for powerful retrieval methods. In *Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '95, pages 282–290, New York, NY, USA, 1995. ACM.

[25] Yoshio Fukushige. Representing probabilistic relations in RDF. In *Proceedings of the 1st International Semantic Web Conference, Workshop 3: Uncertainty Reasoning for the Semantic Web*, pages 106–107. Springer, 2005.

[26] Kavita Ganesan, ChengXiang Zhai, and Jiawei Han. Opinosis: a graph-based approach to abstractive summarization of highly redundant opinions. In *Proceedings of the 23rd International Conference on Computational Linguistics*, COLING '10, pages 340–348, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[27] Udo Hahn and Inderjeet Mani. The challenges of automatic summarization. *Computer*, 33(11):29–36, November 2000.

[28] Tom Heath and Christian Bizer. *Linked Data: Evolving the Web into a Global Data Space*. Synthesis Lectures on the Semantic Web: Theory and Technology. Morgan & Claypool, 1st edition, 2011.

[29] Johannes Hoffart, Fabian M. Suchanek, Klaus Berberich, Edwin Lewis-Kelham, Gerard de Melo, and Gerhard Weikum. YAGO2: Exploring and querying world knowledge in time, space, context, and many languages. In *Proceedings of the 20th international conference companion on World Wide Web*, WWW '11, pages 229–232, New York, NY, USA, 2011. ACM.

[30] Johannes Hoffart, Mohamed Amir Yosef, Ilaria Bordino, Hagen Fürstenau, Manfred Pinkal, Marc Spaniol, Bilyana Taneva, Stefan Thater, and Gerhard Weikum. Robust disambiguation of named entities in text. In *Proceedings*

of the Conference on Empirical Methods in Natural Language Processing, EMNLP '11, pages 782–792, Stroudsburg, PA, USA, 2011. Association for Computational Linguistics.

[31] Hai Huang and Chengfei Liu. Query evaluation on probabilistic RDF databases. In *Proceedings of the 10th International Conference on Web Information Systems Engineering*, WISE '09, pages 307–320, Berlin, Heidelberg, 2009. Springer-Verlag.

[32] Jiewen Huang, Lyublena Antova, Christoph Koch, and Dan Olteanu. MayBMS: A probabilistic database management system. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*, SIGMOD '09, pages 1071–1074, New York, NY, USA, 2009. ACM.

[33] Christian S. Jensen and Richard Thomas Snodgrass. Temporal data management. *IEEE Trans. on Knowl. and Data Eng.*, 11(1), 1999.

[34] Hongyan Jing and Kathleen R. McKeown. Cut and paste based text summarization. In *Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference*, NAACL 2000, pages 178–185, Stroudsburg, PA, USA, 2000. Association for Computational Linguistics.

[35] Dimitris Karampinas and Peter Triantafillou. Crowdsourcing taxonomies. In *Proceedings of the 9th international conference on The Semantic Web: research and applications*, ESWC '12, pages 545–559, Berlin, Heidelberg, 2012. Springer-Verlag.

[36] Richard M. Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972.

[37] Richard M. Karp and Michael Luby. Monte-Carlo algorithms for enumeration and reliability problems. In *Proceedings of the 24th Annual Symposium on Foundations of Computer Science*, SFCS '83, pages 56–64, Washington, DC, USA, 1983. IEEE Computer Society.

[38] Gjergji Kasneci, Shady Elbassuoni, and Gerhard Weikum. MING: Mining informative entity relationship subgraphs. In *Proceedings of the 18th ACM*

conference on Information and knowledge management, CIKM '09, pages 1653–1656, New York, NY, USA, 2009. ACM.

[39] Kevin Knight and Daniel Marcu. Summarization beyond sentence extraction: A probabilistic approach to sentence compression. *Artificial Intelligence*, 139(1):91–107, July 2002.

[40] Oleksandr Kolomiyets, Steven Bethard, and Marie-Francine Moens. Extracting narrative timelines as temporal dependency structures. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1*, ACL '12, pages 88–97, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics.

[41] Markus Krötzsch and Denny Vrandecic. Semantic MediaWiki. In *Foundations for the Web of Information and Services*, pages 311–326. Springer Berlin Heidelberg, 2011.

[42] Liangda Li, Ke Zhou, Gui-Rong Xue, Hongyuan Zha, and Yong Yu. Enhancing diversity, coverage and balance for summarization through structure learning. In *Proceedings of the 18th international conference on World Wide Web*, WWW '09, pages 71–80, New York, NY, USA, 2009. ACM.

[43] Xiao Ling and Daniel S. Weld. Temporal information extraction. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence*, AAAI '10, pages 1385 – 1390. Association for the Advancement of Artificial Intelligence, 2010.

[44] Inderjeet Mani. Summarization evaluation: An overview. In *Proceedings of the NTCIR Workshop 2 Meeting on Evaluation of Chinese and Japanese Text Retrieval and Text Summarization*, pages 77–85. National Institute of Informatics, 2001.

[45] David McClosky and Christopher D. Manning. Learning constraints for consistent timeline extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pages 873–882, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics.

[46] David D. Mcdonald and James D. Pustejovsky. Natural language generation. In *Proceedings of the 9th international joint conference on Artificial intelligence*, pages 799–805. John Wilwy and Sons, 1986.

[47] Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2 - Volume 2*, ACL '09, pages 1003–1011, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics.

[48] Ndapandula Nakashole, Mauro Sozio, Fabian M. Suchanek, and Martin Theobald. Query-time reasoning in uncertain RDF knowledge bases with soft and hard rules. In *Proceedings of the 2nd International Workshop on Searching and Integrating New Web Data Sources*, VLDS '12, pages 15–20. CEUR-WS.org, 2012.

[49] Ndapandula Nakashole, Martin Theobald, and Gerhard Weikum. Scalable knowledge harvesting with high precision and high recall. In *Proceedings of the fourth ACM international conference on Web search and data mining*, WSDM '11, pages 227–236, New York, NY, USA, 2011. ACM.

[50] Marius Pasca. Towards temporal web search. In *Proceedings of the 2008 ACM symposium on Applied computing*, SAC '08, pages 1117–1121, New York, NY, USA, 2008. ACM.

[51] Dragomir R. Radev, Timothy Allison, Sasha Blair-Goldensohn, John Blitzer, Arda Çelebi, Stanko Dimitrov, Elliott Drábek, Ali Hakim, Wai Lam, Danyu Liu, Jahna Otterbacher, Hong Qi, Horacio Saggion, Simone Teufel, Michael Topper, Adam Winkel, and Zhu Zhang. MEAD - a platform for multidocument multilingual text summarization. In *Proceedings of the 4th International Conference on Language Resources and Evaluation*, LREC '04, pages 699–702, Paris, France, 2004. European Language Resources Association (ELRA).

[52] Delip Rao and David Yarowsky. Ranking and semi-supervised classification on large scale graphs using Map-Reduce. In *Proceedings of*

the 2009 Workshop on Graph-based Methods for Natural Language Processing, TextGraphs-4, pages 58–65, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics.

[53] Carl Edward Rasmussen and Christopher KI Williams. *Gaussian processes for machine learning*, volume 1. MIT press Cambridge, MA, 2006.

[54] Christopher Re, Nilesh N. Dalvi, and Dan Suciu. Efficient top-k query evaluation on probabilistic data. In *Proceedings of the 23rd International Conference on Data Engineering*, ICDE '07, pages 886–895, Washington, DC, USA, 2007. IEEE Computer Society.

[55] Dan Roth and Wen-Tau Yih. A linear programming formulation for global inference in natural language tasks. In *Proceedings of the 8th Conference on Computational Natural Language Learning*, CoNLL '04, pages 1–8, Stroudsburg, PA, USA, 2004. Association for Computational Linguistics.

[56] Cristina Sarasua, Elena Simperl, and Natalya Fridman Noy. CrowdMap: Crowdsourcing ontology alignment with microtasks. In *Proceedings of the 11th International Semantic Web Conference*, ISWC '12, pages 525–541, Berlin, Heidelberg, 2012. Springer.

[57] Anish Das Sarma, Martin Theobald, and Jennifer Widom. Exploiting lineage for confidence computation in uncertain and probabilistic databases. In *Proceedings of the 2008 IEEE 24th International Conference on Data Engineering*, ICDE '08, pages 1023–1032, Washington, DC, USA, 2008. IEEE Computer Society.

[58] Sebastian Schaffert, Julia Eder, Szaby Grünwald, Thomas Kurz, Mihai Radulescu, Rolf Sint, and Stephanie Stroka. KiWi - a platform for semantic social software. In *Proceedings of the 4th Semantic Wiki Workshop (SemWiki 2009) at the 6th European Semantic Web Conference (ESWC 2009)*, SemWiki '09. CEUR-WS.org, 2009.

[59] Dou Shen, Jian-Tao Sun, Hua Li, Qiang Yang, and Zheng Chen. Document summarization using conditional random fields. In *Proceedings of the 20th international joint conference on Artifical intelligence*, IJCAI'07, pages 2862–2867, San Francisco, CA, USA, 2007. Morgan Kaufmann Publishers Inc.

[60] Push Singh, Thomas Lin, Erik T. Mueller, Grace Lim, Travell Perkins, and Wan Li Zhu. Open mind common sense: Knowledge acquisition from the general public. In *On the Move to Meaningful Internet Systems, 2002 - DOA/CoopIS/ODBASE 2002 Confederated International Conferences DOA, CoopIS and ODBASE 2002*, pages 1223–1237, London, UK, UK, 2002. Springer-Verlag.

[61] Robert Speer, Catherine Havasi, and Harshit Surana. Using verbosity: Common sense data from games with a purpose. In *Proceedings of the 23rd International Florida Artificial Intelligence Research Society Conference*, FLAIRS '10. AAAI Press, 2010.

[62] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. YAGO: A core of semantic knowledge. In *Proceedings of the 16th international conference on World Wide Web*, WWW '07, pages 697–706, New York, NY, USA, 2007. ACM.

[63] Fabian M. Suchanek, Mauro Sozio, and Gerhard Weikum. SOFIE: A self-organizing framework for information extraction. In *Proceedings of the 18th international conference on World Wide Web*, WWW '09, pages 631–640, New York, NY, USA, 2009. ACM.

[64] Dan Suciu, Dan Olteanu, Christopher RÍę, and Christoph Koch. *Probabilistic Databases*. Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2011.

[65] Yohei Takaku, Nobuhiro Kaji, Naoki Yoshinaga, and Masashi Toyoda. Identifying constant and unique relations by using time-series text. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pages 883–892, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics.

[66] Partha Pratim Talukdar and Koby Crammer. New regularized algorithms for transductive learning. In *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases: Part II*, ECML PKDD '09, pages 442–457, Berlin, Heidelberg, 2009. Springer-Verlag.

[67] Partha Pratim Talukdar and Fernando Pereira. Experiments in graph-based semi-supervised learning methods for class-instance acquisition. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 1473–1481, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[68] Partha Pratim Talukdar, Derry Wijaya, and Tom Mitchell. Coupled temporal scoping of relational facts. In *Proceedings of the 5th ACM international conference on Web search and data mining*, WSDM '12, pages 73–82, New York, NY, USA, 2012. ACM.

[69] Abdullah Uz Tansel, James Clifford, Shashi Gadia, Sushil Jajodia, Arie Segev, and Richard Snodgrass, editors. *Temporal databases: theory, design, and implementation*. Benjamin-Cummings Publishing Co., Inc., Redwood City, CA, USA, 1993.

[70] Benjamin E. Teitler, Michael D. Lieberman, Daniele Panozzo, Jagan Sankaranarayanan, Hanan Samet, and Jon Sperling. NewsStand: A new view on news. In *Proceedings of the 16th ACM SIGSPATIAL international conference on Advances in geographic information systems*, GIS '08, pages 18:1–18:10, New York, NY, USA, 2008. ACM.

[71] Martin Theobald, Mauro Sozio, Fabian Suchanek, and Ndapandula Nakashole. URDF: Efficient reasoning in uncertain RDF knowledge bases with soft and hard rules. Technical Report MPI-I-2010-5-002, Max Planck Institute Informatics (MPI-INF), Saarbruecken, Germany, February 2010.

[72] Tomasz Tylenda, Mauro Sozio, and Gerhard Weikum. Einstein: physicist or vegetarian? summarizing semantic type graphs for knowledge discovery. In *Proceedings of the 20th international conference companion on World Wide Web*, WWW '11, pages 273–276, New York, NY, USA, 2011. ACM.

[73] Octavian Udrea, V. S. Subrahmanian, and Zoran Majkic. Probabilistic RDF. In *IEEE Conference on Information Reuse and Integration*, IRI '06, pages 172–177. IEEE Computer Society, 2006.

[74] Marc Verhagen, Robert Gaizauskas, Frank Schilder, Mark Hepple, Jessica Moszkowicz, and James Pustejovsky. The TempEval challenge: Identifying

temporal relations in text. *Language Resources and Evaluation*, 43:161–179, 2009.

[75] Marc Verhagen, Inderjeet Mani, Roser Sauri, Jessica Littman, Robert Knippen, Seok Bae Jang, Anna Rumshisky, John Phillips, and James Pustejovsky. Automating temporal annotation with TARSQI. In *Proceedings of the ACL 2005 on Interactive poster and demonstration sessions*, ACLdemo '05, pages 81–84, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics.

[76] Luis von Ahn, Mihir Kedia, and Manuel Blum. Verbosity: A game for collecting common-sense facts. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '06, pages 75–78, New York, NY, USA, 2006. ACM.

[77] Xiaojun Wan and Jianwu Yang. Multi-document summarization using cluster-based link analysis. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '08, pages 299–306, New York, NY, USA, 2008. ACM.

[78] Dingding Wang, Shenghuo Zhu, Tao Li, and Yihong Gong. Multi-document summarization using sentence-based topic models. In *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*, ACL '09, pages 297–300, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics.

[79] Jiannan Wang, Tim Kraska, Michael J. Franklin, and Jianhua Feng. CrowdER: Crowdsourcing entity resolution. *Proc. VLDB Endow.*, 5(11):1483–1494, jul 2012.

[80] Yafang Wang, Maximilian Dylla, Zhaochun Ren, Marc Spaniol, and Gerhard Weikum. PRAVDA-live: Interactive knowledge harvesting. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, CIKM '12, pages 2674–2676, New York, NY, USA, 2012. ACM.

[81] Yafang Wang, Maximilian Dylla, Marc Spaniol, and Gerhard Weikum. Coupling label propagation and constraints for temporal fact extraction. In *Proceedings of the 50th Annual Meeting of the Association for Computational*

Linguistics: Short Papers - Volume 2, ACL ’12, pages 233–237, Stroudsburg, PA, USA, 2012. Association for Computer Linguistics.

[82] Yafang Wang, Mohamed Yahya, and Martin Theobald. Time-aware reasoning in uncertain knowledge bases. In Ander de Keijzer and Maurice van Keulen, editors, *Proceedings of the Fourth International VLDB workshop on Management of Uncertain Data (MUD 2010) in conjunction with VLDB 2010*, volume WP10-04 of *CTIT Workshop Proceedings Series*, pages 51–65, University of Twente, The Netherlands, 2010. Centre for Telematics and Information Technology (CTIT).

[83] Yafang Wang, Bin Yang, Lizhen Qu, Marc Spaniol, and Gerhard Weikum. Harvesting facts from textual web sources by constrained label propagation. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, CIKM ’11, pages 837–846, New York, NY, USA, 2011. ACM.

[84] Yafang Wang, Bin Yang, Spyros Zoupanos, Marc Spaniol, and Gerhard Weikum. Scalable spatio-temporal knowledge harvesting. In *Proceedings of the 20th international conference companion on World Wide Web*, WWW ’11, pages 143–144, New York, NY, USA, 2011. ACM.

[85] Yafang Wang, Mingjie Zhu, Lizhen Qu, Marc Spaniol, and Gerhard Weikum. Timely YAGO: Harvesting, querying, and visualizing temporal knowledge from wikipedia. In *Proceedings of the 13th International Conference on Extending Database Technology*, EDBT ’10, pages 697–700, New York, NY, USA, 2010. ACM.

[86] Fei Wu and Daniel S. Weld. Automatically refining the wikipedia infobox ontology. In *Proceedings of the 17th international conference on World Wide Web*, WWW ’08, pages 635–644, New York, NY, USA, 2008. ACM.

[87] Feiyu Xu, Hans Uszkoreit, and Hong Li. A seed-driven bottom-up machine learning framework for extracting relations of various complexity. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*, ACL ’07, pages 584–591, Stroudsburg, PA, USA, 2007. Association for Computational Linguistics.

[88] Limin Yao, Sebastian Riedel, and Andrew McCallum. Collective cross-document relation extraction without labelled data. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, EMNLP '10, pages 1013–1023, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics.

[89] Alexander Yates, Michael Cafarella, Michele Banko, Oren Etzioni, Matthew Broadhead, and Stephen Soderland. TextRunner: Open information extraction on the web. In *Proceedings of Human Language Technologies: The Annual Conference of the North American Chapter of the Association for Computational Linguistics: Demonstrations*, NAACL-Demonstrations '07, pages 25–26, Stroudsburg, PA, USA, 2007. Association for Computational Linguistics.

[90] Hongyuan Zha. Generic summarization and keyphrase extraction using mutual reinforcement principle and sentence clustering. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '02, pages 113–120, New York, NY, USA, 2002. ACM.

[91] Ce Zhang, Feng Niu, Christopher Ré, and Jude W. Shavlik. Big data versus the crowd: Looking for relationships in all the right places. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1*, ACL '12, pages 825–834, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics.

[92] Qi Zhang, Fabian Suchanek, and Gerhard Weikum. TOB: Timely ontologies for business relations. In *Proceedings of 11th International Workshop on the Web and Databases*, WebDB '10, New York, NY, USA, 2008. ACM.

[93] Xiang Zhang, Gong Cheng, and Yuzhong Qu. Ontology summarization based on RDF sentence graph. In *Proceedings of the 16th international conference on World Wide Web*, WWW '07, pages 707–716, New York, NY, USA, 2007. ACM.

[94] Zhu Zhang. Weakly-supervised relation classification for information extraction. In *Proceedings of the 13th ACM international conference on*

Information and knowledge management, CIKM '04, pages 581–588, New York, NY, USA, 2004. ACM.

[95] Guodong Zhou, Min Zhang, Dong-Hong Ji, and Qiaoming Zhu. Tree kernel-based relation extraction with context-sensitive structured parse tree information. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '07, pages 728–736, Stroudsburg, PA, USA, 2007. Association for Computational Linguistics.

[96] Xiaojin Zhu, Zoubin Ghahramani, and John D. Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of The 20th International Conference on Machine Learning*, ICML '03, pages 912–919. AAAI Press, 2003.

Appendix A. Rules and Queries

Aggregation Rules

$A_1:\ \mathrm{joinsYouthClub}(a,b) \wedge \mathrm{duringYouthClub}(a,b) \wedge \mathrm{leavesYouthClub}(a,b)$
$\quad \rightarrow \mathrm{playsForYouthClub}(a,b)$

$A_2:\ \mathrm{joinsSeniorClub}(a,b) \wedge \mathrm{duringSeniorClub}(a,b) \wedge \mathrm{leavesSeniorClub}(a,b)$
$\quad \rightarrow \mathrm{playsForSeniorClub}(a,b)$

$A_3:\ \mathrm{joinsNationalTeam}(a,b) \wedge \mathrm{duringNationalTeam}(a,b) \wedge \mathrm{leavesNationalTeam}(a,b)$
$\quad \rightarrow \mathrm{playsForNationalTeam}(a,b)$

$A_4:\ \mathrm{beginManagesTeam}(a,b) \wedge \mathrm{duringManagesTeam}(a,b) \wedge \mathrm{endManagesTeam}(a,b)$
$\quad \rightarrow \mathrm{managesTeam}(a,b)$

Inference Rules

Players playing for teams are summarized into $\mathrm{playsForTeam}$.

$C_1:\ \mathrm{playsForYouthClub}(a,b) \rightarrow \mathrm{playsForTeam}(a,b)$
$\mathrm{playsForSeniorClub}(a,b) \rightarrow \mathrm{playsForTeam}(a,b)$
$\mathrm{playsForNationalTeam}(a,b) \rightarrow \mathrm{playsForTeam}(a,b)$

If two players play for the same team at the same time, they are teammates.

$C_2:\ \mathrm{playsForTeam}(a,b,t_1) \wedge \mathrm{playsForTeam}(c,b,t_2) \wedge \mathrm{overlaps}(t_1,t_2) \wedge \mathrm{notEquals}(a,c)$
$\quad \rightarrow \mathrm{teammates}(a,c)$

If one player plays for the same team after another player, then the former is a successor of the latter.

$C_3:\ \mathrm{playsForTeam}(a,b,t_1) \wedge \mathrm{playsForTeam}(c,b,t_2) \wedge \mathrm{after}(t_1,t_2) \wedge \mathrm{notEquals}(a,c)$
$\quad \rightarrow \mathrm{successor}(a,c)$

If one player plays for the same team before another player, then the former is an ancestor of the latter.

$C_4:\ \mathrm{playsForTeam}(a,b,t_1) \wedge \mathrm{playsForTeam}(c,b,t_2) \wedge \mathrm{before}(t_1,t_2) \wedge \mathrm{notEquals}(a,c)$
$\quad \rightarrow \mathrm{predecessor}(a,c)$

Players who have played for more than 1460 days (more than 4 years) for a team.

$C_5:\ \mathrm{playsForTeam}(a,b,t_1) \wedge \mathrm{durationMoreThan}(t_1,1460)$
$\quad \rightarrow \mathrm{playedMoreThan4YearsForTeam}(a,b)$

If a coach manages the team when a player is playing for the team, the coach trained this player.

C_6: $\mathtt{managesTeam}(a, b, t_1) \wedge \mathtt{playsForTeam}(c, b, t_2) \wedge \mathtt{overlaps}(t_1, t_2)$
$\rightarrow \mathtt{isCoachOf}(a, c)$

If a coach manages a team, and this is a national team, then he is a coach of a national team.

C_7: $\mathtt{managesTeam}(a, b, t_1) \wedge \mathtt{playsForNationalTeam}(c, b, t_2) \wedge \mathtt{overlaps}(t_1, t_2)$
$\rightarrow \mathtt{isCoachOfNationalTeam}(a, b)$

Queries

Single-fact queries:

For which teams did David Beckham play (and when)?

Q_1: $\mathtt{playsForTeam}(\mathtt{DavidBeckham}, x)$

Which teams has Alex Ferguson managed (and when)?

Q_2: $\mathtt{managesTeam}(\mathtt{AlexFerguson}, x)$

Who are the predecessors of David Beckham?

Q_3: $\mathtt{predecessor}(x, \mathtt{DavidBeckham})$

Chain query:

Who are the coaches of David Beckham, and which teams did they previously play for?

Q_4: $\mathtt{isCoachOf}(x, \mathtt{DavidBeckham}) \wedge \mathtt{playsForTeam}(x, y)$

Which teammates of David Beckham, participated in an activity with Zinedine Zidane?

Q_5: $\mathtt{teammates}(\mathtt{DavidBeckham}, y) \wedge \mathtt{participatedIn}(y, z) \wedge \mathtt{participatedIn}(\mathtt{ZinedineZidane}, z)$

Star query:

Who are the coaches of the England National Football Team, what cups did they win, and which activities did they join?

Q_6: $\mathtt{isCoachOfNationalTeam}(x, \mathtt{EnglandNationalFootballTeam}) \wedge \mathtt{winsCup}(x, y)$
$\wedge\, \mathtt{participatedIn}(x, z)$

Who played for Manchester United for more than 4 years and was a teammate of David Beckham?

Q_7: $\mathtt{playedMoreThan4YearsForTeam}(x, \mathtt{ManchesterUnited}) \wedge \mathtt{teammates}(x, \mathtt{DavidBeckham})$

Clique query:

Who are the successors of David Beckham, and who won the same cup as Beckham?

Q_8: $\mathtt{successor}(x, \mathtt{DavidBeckham}) \wedge \mathtt{winsCup}(x, z, t_1) \wedge \mathtt{winsCup}(\mathtt{DavidBeckham}, z, t_2)$

Table A.1.: Aggregation rules, inference rules, and queries used for the experiments.

Appendix B.

Convexity Proof

We show that the function 4.5 is convex, following the convention of [53], we rewrite the estimated labels of $\widehat{\mathbf{Y}}$ into a vector $\widehat{\mathbf{y}}$ with length $n \times (m+1)$.

$$\widehat{\mathbf{y}} = (\widehat{\mathbf{Y}}_{11}, ..., \widehat{\mathbf{Y}}_{n1}, \widehat{\mathbf{Y}}_{12}, ..., \widehat{\mathbf{Y}}_{n2}, ..., \widehat{\mathbf{Y}}_{1(m+1)}, ..., \widehat{\mathbf{Y}}_{n(m+1)})^T \tag{B.1}$$

In the same way, the initial label assignment matrix $\mathbf{Y}$ can be converted into a vector $\mathbf{y}$. Following the same idea, the graph Laplacian $\mathbf{L}$ is block diagonal with the matrices $\mathbf{L_1}, ..., \mathbf{L_{m+1}}$. In our case, $\mathbf{L_i} = \mathbf{L}$.

$$\begin{bmatrix} \mathbf{L_1} & & & \\ & \mathbf{L_2} & & \\ & & ... & \\ & & & \mathbf{L_{m+1}} \end{bmatrix}. \tag{B.2}$$

And the diagonal matrix $\mathbf{S}$ is represented analogously.

$$\begin{bmatrix} \mathbf{S_1} & & & \\ & \mathbf{S_2} & & \\ & & ... & \\ & & & \mathbf{S_{m+1}} \end{bmatrix}. \tag{B.3}$$

Then the matrix $\hat{\mathbf{C}}$ implied by inclusion constraints has the same form as the unnormalized graph Laplacian, which is $\hat{\mathbf{C}} = \mathbf{D}^{\hat{\mathbf{C}}} - \mathbf{W}^{\hat{\mathbf{C}}}$. The elements $\mathbf{W}^{\hat{\mathbf{C}}}{}_{\nu^{\ell}\nu^{\ell'}}$ and $\mathbf{W}^{\hat{\mathbf{C}}}{}_{\nu^{\ell'}\nu^{\ell}}$ of affinity matrix $\mathbf{W}^{\hat{\mathbf{C}}}$ have the same positive weights $\mathbf{C}_{\ell\ell'} \cdot \mathbf{Y}_{\nu\ell'}$, if an inclusion constraint between labels ℓ and ℓ' is included. The remaining entries of $\mathbf{W}^{\hat{\mathbf{C}}}$ are zeros. Then the degree matrix $\mathbf{D}^{\hat{\mathbf{C}}}$ is a diagonal matrix with $\mathbf{D}^{\hat{\mathbf{C}}}{}_{ii} = \sum_j \mathbf{W}^{\hat{\mathbf{C}}}{}_{ij}$.

With the new formulation of the problem, the optimization problem including inclusion constraints takes the form

$$\begin{aligned}&\min_{\widehat{\mathbf{y}}} \big[(\widehat{\mathbf{y}}-\mathbf{y})^{\mathsf{T}}\mathbf{S}(\widehat{\mathbf{y}}-\mathbf{y}) + \mu_1\widehat{\mathbf{y}}^{\mathsf{T}}\mathbf{L}\widehat{\mathbf{y}}\\ &+ \mu_2(\widehat{\mathbf{y}}-\mathbf{r})^{\mathsf{T}}\mathbf{I}(\widehat{\mathbf{y}}-\mathbf{r}) + \mu_3\widehat{\mathbf{y}}^{\mathsf{T}}\hat{\mathbf{C}}\widehat{\mathbf{y}}\big] \quad \text{s.t. } \widehat{y}_v^{l} \geq \widehat{y}_v^{l'}\end{aligned}$$

The Hessian matrix of the objective function with respect to $\widehat{\mathbf{y}}$ is

$$\mathbf{S} + \mu_1\mathbf{L} + \mu_2\mathbf{I} + \mu_3\hat{\mathbf{C}} \tag{B.4}$$

which is positive definite if $\mu_2 > 0$ ($\hat{\mathbf{C}}$ is positive semi-definite following the same reason as for the unnormalized graph Laplacian). Finally, as the feasible region is defined by linear constraints, the optimization problem is convex.

List of Figures

List of Tables