UNIVERSITÄT DES SAARLANDES

DOCTORAL THESIS

Human Computing and Crowdsourcing Methods for Knowledge Acquisition

Sarath Kumar Kondreddi
Max Planck Institut für Informatik

A dissertation presented for the degree of
Doctor of Engineering

in the

Faculty of Natural Sciences and Technology
UNIVERSITY OF SAARLAND

Saarbrücken, May 2014

Promotionskolloquium

Datum	06 Mai 2014
Ort	Saarbrücken
Dekan der Naturwissenschaftlich-Technischen Fakultät I	Prof. Dr-Ing. Markus Bläser *Universität des Saarlandes*

Prüfungskommission

Vorsitzender	Prof. Matthias Hein
Gutachter	Prof. Dr.-Ing. Gerhard Weikum
Gutachter	Prof. Peter Triantafillou
Gutachter	Dr.-Ing. Klaus Berberich
Beisitzer	Dr.-Ing. Sebastian Michel

UNIVERSITÄT DES SAARLANDES

Thesis Abstract

Database and Information Systems Group

Max-Planck Institute for Informatics

Doctor of Engineering

Human Computing and Crowdsourcing Methods for Knowledge Acquisition

by Sarath Kumar KONDREDDI

Abstract

Ambiguity, complexity, and diversity in natural language textual expressions are major hindrances to automated knowledge extraction. As a result state-of-the-art methods for extracting entities and relationships from unstructured data make incorrect extractions or produce noise. With the advent of human computing, computationally hard tasks have been addressed through human inputs. While text-based knowledge acquisition can benefit from this approach, humans alone cannot bear the burden of extracting knowledge from the vast textual resources that exist today. Even making payments for crowdsourced acquisition can quickly become prohibitively expensive.

In this thesis we present principled methods that effectively garner human computing inputs for improving the extraction of knowledge-base facts from natural language texts. Our methods *complement* automatic extraction techniques with human computing to reap benefits of both while overcoming each other's limitations. We present the architecture and implementation of HIGGINS, a system that combines an information extraction (IE) engine with a human computing (HC) engine to produce high quality facts. Using automated methods, the IE engine compiles dictionaries of entity names and relational phrases. It further combines statistics derived from large Web corpora with semantic resources like WordNet and ConceptNet to expand the dictionary of relational phrases. It employs specifically designed statistical language models for phrase relatedness to come up with questions and relevant candidate answers that are presented to human workers. Through extensive experiments we establish the superiority of this approach in extracting relation-centric facts from text. In our experiments we extract facts about fictitious characters in narrative text, where the issues of diversity and complexity in expressing relations are far more pronounced. Finally, we also demonstrate how interesting human computing games can be designed for knowledge acquisition tasks.

KURZFASSUNG

Mehrdeutigkeit, Komplexität sowie Vielfältigkeit im Ausdruck stellen die automatische Extraktion von Wissen aus natürlichsprachlichen Texten vor grosse Herausforderungen. Infolgedessen sind aktuelle Methoden zur Informationsxtraktion (IE) von Entitäten sowie deren wechselseitigen Relationen aus unstrukturierten Daten oft fehleranfällig. Durch die Methodik des *Human Computing (HC)* kann eine Vielzahl von schwierigen Problemen mit Hilfe menschlicher Eingaben adressiert werden. Wenngleich Problemstellungen des textbasierten Wissenserwerbs auch durch HC unterstützt werden, kann die Wissensextraktion aus sehr umfangreichen Textsammlungen nicht alleine durch diesen manuellen Ansatz gelöst werden. Weiterhin sind, im Rahmen eines Bezahlungsmodells, die durch Vergütung der von menschlichen Anwendern ausgeführten Kleinstaufgaben entstehenden Kosten unbezahlbar.

In dieser Arbeit stellen wir Methoden vor, die Algorithmen zur automatischen Extraktion mit den durch Human Computing gewinnbaren Informationen kombinieren. Wir stellen die Architektur und Implementierung des HIGGINS-Systems vor, das IE und HC synergetisch verbindet mit dem Ziel hochwertiger und umfassender Wissensakquisition aus Texten. Die IE-Komponente von HIGGINS konstruiert zunächst umfangreiche Sammlungen von Entitätsbezeichnungen und relationalen Paraphrasen. Weiterhin werden aus Webkorpora gewonnene statistische Informationen mit semantischen Ressourcen wie WordNet und ConceptNet kombiniert, um die gewonnenen relationalen Phrasen zu expandieren. Spezifisch definierte statistische Modelle werden zur Bestimmung der semantischen Ähnlichkeit von Phrasen eingesetzt. Auf diese Weise generiert die IE-Komponente sowohl Fragen für HC als auch relevante Antwortmöglichkeiten. Die HC-Komponente erzeugt daraus kleine Aufgaben für Crowdsourcing oder Onlinespiele und sammelt das daraus resultierende Nutzerfeedback. Eine umfassende experimentelle Evaluation belegt die Praktikabilität und Vorteile dieser kombinierten IE/HC-Methodologie.

SUMMARY

Automatic information extraction (IE) from text corpora and Web sources enables the construction of large knowledge bases (KB) of relational facts with millions of entities. However, fact extraction from natural language text is challenging as automated IE techniques need to cope with textual sentences that have complex structure, contain pronouns and other anaphoras, and use ambiguous wording. In processing such inputs, they often produce erroneous or incomplete facts, or miss out on extracting relevant facts. As a result, the quality and coverage of the resulting KB suffers. This motivates the need for exploiting human intelligence and knowledge for assessing the validity of extracted facts, correcting errors and for gathering additional knowledge.

For textual inputs, humans still outperform state-of-the-art computational methods in understanding language and context, in resolving ambiguity and performing objective reasoning. Therefore human computing (HC) and crowdsourcing methods offer a natural alternative to overcome the limitations of automated IE. Challenging IE tasks such as compiling relationships between entities can be converted into a questions in an HC game, soliciting answers from interested users. Alternatively, IE tasks could be set up as human intelligence tasks (HITs) on platforms like Amazon Mechanical Turk or CrowdFlower. Despite these opportunities, scaling up is an issue; HC game players cannot solely bear the burden of extracting facts from vast textual corpora, and paid crowdsourcing at scale quickly becomes prohibitively expensive.

In consideration of individual benefits and limitations of automated IE and HC, this thesis presents methods that enable effective fact acquisition by combining both of them. The key idea is to use automatic IE to generate questions and candidate answers and evaluate them through HC inputs. To this end, we designed and built the HIGGINS system that employs this combination to generate relational facts from Web corpora. As a running application we utilize movie and book narratives and compile relationships between character roles in story plots.

The architecture of the HIGGINS system couples an automated IE component with an HC component. The IE component is recall-oriented: it applies techniques from computational linguistics and uses additional heuristics to build large lexicons of entity names and relational phrases from narrative corpora. To increase coverage over the large variety of relations seen in narratives, the relational phrases are combined with phrases from semantic resources like WordNet and ConceptNet. Furthermore, to obtain highly specific phrases, additional phrase mining is performed using co-occurrence statistics derived over large Web corpora. All these phrases are combined using a mixture model and used in specifically designed statistical (translation) language models that rank candidate relations for entity pairs, taking as query the contexts in which the pairs occur. In the HC component the ranked candidates are aggregated, diversified and transformed into crowdsourced HITs or questions of an HC game.

Our extensive experimentation on HIGGINS , using crowdsourced HITs, consistently shows high precision and recall, demonstrating the synergistic benefits of combining IE and HC. HIGGINS obtains a larger number of facts compared to state-of-the-art IE system OLLIE, and outperforms a purely HC approach in terms of both precision and recall. Through our experiments we also corroborate that the combination of semantic resources and statistically derived phrases achieves higher output quality, compared to using either of them individually. Finally, our experiments show that questions in HIGGINS generate good inter-annotator agreement with only a handful of judgements – a fact that directly translates into reduced costs for human engagement.

In this work we also demonstrate how the HIGGINS system can support the construction of different HC games for fact acquisition. We map the objectives of game players onto the generation of correct relational facts. We constructed the MOVIEWIZARD and BOOKWIZARD games, where individual players compete to achieve high scores by pointing out appropriate relations between character pairs among the candidates presented to them. We also built MOVIEGURUS , an interactive two-player game in which a player helps a randomly-paired partner to guess the movie title by typing out relationships between anonymized character roles. The HIGGINS system harnesses relational facts through these player interactions.

Zusammenfassung

Die automatisierte Informationsextraktion (IE) aus Textkorpora und dem World Wide Web ermöglicht die Generierung umfangreicher Wissensbasen in Form von relationalen Fakten über Millionen von Entitäten. Aufgrund der komplexen Struktur von natürlichsprachigen Sätzen, Anaphorik (z. B. in der Form von Pronomen), sowie mehrdeutiger Formulierung, ist die zuverlässige Extraktion von Fakten aus Texten eine große Herausforderung für IE-Methoden. Dies führt zu fehlerhaften sowie unvollständigen Fakten und hat negativen Einfluss auf die Qualität und Vollständigkeit der generierten Wissensbasis. Diese Problematik motiviert die Untersuchung von Möglichkeiten der Einbindung menschlicher Intelligenz und menschlichen Wissens in den Extraktionsprozess, mit dem Ziel der Validierung und Fehlerkorrektur extrahierter Fakten sowie der Gewinnung zusätzlicher Fakten.

Methoden des Human-Computing (HC) und Crowdsourcing-Ansätze sind eine naheliegende Alternative, um die Defizite automatisierter Verfahren zu beheben. Schwierige IE-Aufgaben können zum Beispiel durch Antworten auf Fragen im Rahmen eines HC-Spiels adressiert werden. Alternativ können entsprechende Bearbeitungseinheiten, sogenannte HITs (Human Intelligence Tasks), auf Crowdsourcing-Plattformen wie Amazon Mechanical Turk oder Crowdflower angeboten werden. Für sehr große Textsammlungen ist jedoch die Zahl der benötigten Spieler viel zu hoch, und die Kosten für die Vergütung von Arbeitern beim Crowdsourcing sind unbezahlbar.

Diese Arbeit stellt Methoden vor, mit denen durch Kombination von IE und HC die Wissensakquisition für Computer wesentlich verbessert werden kann. Die Kernidee des Verfahrens liegt im Einsatz von IE-Methoden zur Generierung von Fragen sowie Antwortmöglichkeiten und dem anschließenden Einsatz von HC in Form des Sammelns von Antworten durch Nutzer. Wir stellen das HIGGINS-System vor, in dem dieser Lösungsansatz realisiert wurde. Als Beispielanwendung in dieser Arbeit dient die Extraktion von relationalen Beziehungen zwischen Charakteren in Zusammenfassungen von Romanen oder Filmen.

Die Architektur des HIGGINS-Systems koppelt eine automatisierte IE-Komponente mit einer HC-Komponente. Die IE-Komponente zielt auf hohe Ausbeute und basiert auf computerlinguistischen und heuristischen Methoden, um in einem ersten Schritt umfangreiche Verzeichnisse von Entitätsbezeichnungen und relationalen Paraphrasen zu erstellen. Die so gesammelten Paraphrasen werden mit semantischen Resourcen wie WordNet und ConceptNet kombiniert, um eine bessere Abdeckung der vielfältigen Relationen zu erreichen. Relationale Phrasen werden mit Hilfe eines statistischen Modells zueinander in Beziehung gesetzt und als Evidenz für spezifische Relationen gewichtet. In der HC-Komponente wird dieses Modell verwendet, um eine Rangliste von Phrasen bzw. Relationen zu konstruieren, aus denen die Antwortmöglichkeiten für Crowdsourcing-HITs oder Spielsituationen erstellt werden.

Unsere experimentelle Evaluation des HIGGINS-Systems auf einer Crowdsourcing-Plattform untermauert die Verbesserungen in Präzision und Ausbeute, die durch die Synergie von IE und HC erreicht werden. Im Vergleich mit dem IE-System OLLIE, einem der führenden vollautomatischen Softwarewerkzeuge, liefert HIGGINS eine größere Anzahl von Fakten mit hoher Qualität. Die Experimente zeigen, dass die Kombination von semantischen Ressourcen und mit statistischen Methoden gesammelten Phrasen wesentlich bessere Ergebnisse liefert als jede der beiden Basiskomponenten alleine. Die Experimente zeigen auch ein hohes Maß an Übereinstimmung zwischen verschiedenen Nutzern, was sich direkt in geringeren Kosten niederschlägt.

Desweiteren zeigen wir, wie mit dem HIGGINS-System verschiedene HC-Spiele zur Wissensakquisition erstellt werden können. Das Ziel eines Spielers ist dabei mit der Gewinnung korrekter relationaler Fakten verknüpft. Wir präsentieren zwei Spiele, MovieWizard und BookWizard, in denen Spieler durch die Auswahl zutreffender Relationen zwischen Charakteren hohe Punktzahlen anstreben. MovieGurus ist ein weiteres Spiel, für zwei Personen, bei dem ein Spieler seinem zufällig zugewiesenen Partner durch Auflistung von Beziehungen zwischen anonymisierten Charakteren hilft, einen Filmtitel zu erraten. Die Interaktionen zwischen Spielern werden von HIGGINS genutzt, um relationale Fakten zu akquirieren.

Acknowledgements

Foremost, I would like to express my deepest gratitude to Prof. Gerhard Weikum for giving me the opportunity to pursue doctoral studies under his guidance. His constant motivation, excellent scientific advice and outstanding support has made this work possible. For this, I would like to sincerely thank him. I am also indebted to Prof. Peter Triantafillou for his valuable advice on many aspects of this work. I thank him for his patience, his willingness to look into details and generous guidance. I would also like to thank Dr. Klaus Berberich for examining and reviewing this thesis. I would like to acknowledge the authors and editors of en.wikipedia.org and www.sparknotes.com/lit/ for making their data available. Also, I would like to acknowledge the authors of the Stanford CoreNLP software and the OpenIE tools, ReVerb, PATTY and OLLIE, for making their data and source code available.

I am extremely thankful to my colleagues at the Max Planck Institute for Informatics for the stimulating ideas and discussions, among them Avishek Anand, Stephan Seufert, Johannes Hoffart and Ndapandula Nakashole. I would also like to acknowledge and thank the members of the Database and Information Systems group for critically reviewing my work, participating in my game-based case studies and providing valuable suggestions. Furthermore, I would like to express my sincere thanks to the staff at Max Planck Institute for being very kind and providing assistance when needed. I owe many thanks to the International Max Planck Research School for my financial support, which allowed me to focus on my research. Last but not the least, I am greatly indebted to my family and friends for their constant encouragement and support throughout the years.

Contents

1 CHAPTER

Introduction

1.1 Motivation

Building computer programs that reason and use knowledge to solve complex problems is the goal of Artificial Intelligence. This goal can only be fully realized if computing systems are made aware of real-world knowledge. Knowledge Acquisition (KA), a sub-field in AI, deals with extracting, structuring and organizing data from various sources, therefore enabling computers to gain knowledge of entities and their relationships. Structured information thus harnessed is stored in machine-readable form, typically as entity-relation-entity triples or facts, which are instances of binary relationships. Information from such repositories of facts, known as knowledge bases, is readily interpretable by machines. Factual knowledge from knowledge bases help machines to interpret character strings as entities, and provide information on their types, their semantic classes or their relationships with other entities. Given their tremendous utility, large KBs such as Freebase [1, 2], DBPedia [3], YAGO [4], OpenCyc [5], Probase [6] and NELL [7, 8] are increasingly being built and used in areas that include Web search, machine translation, summarization, question answering, information integration and expert systems.

Knowledge Acquisition from Text.

Much of the human knowledge is expressed in textual form; either in books, or as documents on the World Wide Web or elsewhere. Knowledge acquisition from textual sources therefore becomes crucial to build knowledge bases with high coverage. However it is a difficult endeavour to extract factual data from millions of text documents and reconcile it with already extracted knowledge. Employing humans solely for this purpose is impracticable and prohibitively expensive. For these reasons, KA critically relies on information extraction (IE) technology, which provides the capability to capture fact triples from text in an automated way. Developing automated IE techniques that (i) understand the grammatical arrangements as well as the semantics of natural language, and (ii) work at scale, is an active research area in AI.

Limitations of Automatic Extraction Methods.

To extract fact triples from text, IE techniques combine methods from pattern matching, computational linguistics and statistical learning. Traditional techniques of IE [9–15] make use of prespecified sets of relations to extract relational tuples from text, relying on extensive human labour to generate extraction rules or training samples for supervised methods. These IE techniques cannot extend to heterogeneous corpora where the relations are diverse and their number is large. For instance, narratives from movies, books or news carry a large number of diverse relations. Moreover each relation can be expressed in different ways (often complicated) within the text: for example, “Bond shot Xenia dead” or “Bond quickly got rid of Xenia” for the relation “Bond killed Xenia”. As there are many such infrequently occurring expressions in text, there is not much hope to achieve good coverage by specifying extraction rules or by painstakingly generating training data.

Methods under the Open IE paradigm overcome these limitations and perform scalable extraction with no human involvement. *Open IE* methods [16–20] can derive a wide diversity of relational facts between entities based on detecting and analyzing noun phrases for entities and verb-centric phrases for relations, such as: “Vesper” “*was planted to trap*” “Bond”, “Bond” “*saves*” “Vesper” and “Vesper” “*finally falls for*” “Bond” (from the movie story *Casino Royale*).

However, there are fundamental limitations of IE technology. IE methods can yield noisy or semantically meaningless entity-relation triples such as “Vesper” “*certainly has*” “Bond” (from the sentence “Vesper certainly has Bond sized up.”), or, miss out on interesting relations (from the sentence “Vesper and Bond confess their love for each other.”, assuming “and” is not detected as a relation). This occurs because automated IE generally faces obstacles: input sentences with complex structures, use of pronouns and other anaphoras, and ambiguous wording. The following snippet (from `imdb.com`) is a daunting example: "He quickly grabs Vesper and they kiss in the stairway entrance to cover themselves."

Leveraging the Crowds.

The idea of Human Computing (HC) is to leverage capabilities of humans to generate solutions for tasks that are computationally hard. Human Computing and commercial crowdsourcing have been successfully employed to help with tasks where fully automated solutions are deemed inadequate [21, 22]. A typical approach is to place so-called human intelligence tasks or *HITs*, specifically designed micro-tasks, on platforms like Amazon's Mechanical Turk [23] or CrowdFlower [24]. Human contributors on such platforms (referred as the "crowd") accomplish these HITs to be compensated through payments. HC has been applied to various tasks including quality assessment of rankings, summaries, and IE results [21, 25, 26], query answering [27–29], entity matching [30, 31], ontology alignment [32], image search and annotation [33–35], and acquiring commonsense properties [36, 37] and taxonomies [38]. For certain problems, HC has been successfully leveraged in game form [39]: correcting OCR errors, generating image annotations, translating words to foreign languages, and so on.

Our thesis is that HC is a natural alternative to overcome the fundamental limitations of automated IE. It can tap human intelligence and knowledge to assess candidate facts, to correct errors, and to add new facts. Human *intelligence* can help to resolve pronouns in complex sentences or to identify erroneous paths in the dependency-parsing of natural language. Human *knowledge* on special topics such as movies, books, or medicine can add new facts, that may be entirely missing from the text, or help derive entity-relation triples, e.g., about movie or book characters, that are virtually impossible to extract automatically, as they may not be explicitly mentioned. Despite this great potential, to our knowledge there have been no previous attempts to employ HC and crowdsourcing platforms for the difficult KA task of extracting entity-relation triples.

1.2 Our Approach

The goal of our work is to combine HC methods with automated IE for effective acquisition of relational facts. We propose using automatic IE techniques to generate questions and candidate answers (for multiple-choice questions), for a HC game or for crowdsourced HITs. Our hypothesis is that this improves the quality of user contributions and lowers the cost of crowdsourcing.

1.2.1 Challenges

In our work, we consider compiling relational facts from complex textual inputs such as story narratives and movie plots. While these inputs contain a rich variety of relations, they are expressed in diverse and complex ways. We use IE techniques to process the input sentences and generate possible relations for an entity pair, which are then validated using human inputs.

Overcoming sparseness. Our IE methods should be able to pick meaningful relation candidates by analyzing phrases from the textual sentences. However, relevant patterns for advanced relations tend to be rare. For example, phrases like “imprison”, “orders to assassinate”, or “pretends to fall in love” are infrequent even in a large corpus of movie plots and book summaries. Overcoming this sparseness issue is key to detect appropriate relations and achieve good coverage.

Limiting HC costs. For applications with human experts (e.g., movie aficionados, or book lovers, or experts in diseases and medicine, etc.), one would expect that HC can be nicely cast into game form, thus enticing more users to contribute on the KA task. However, despite the inherent promises of HC for KA, humans alone cannot carry this burden. First, the number of real experts is typically limited. Second, these experts are not so likely to participate in online games. Hence, inevitably, HC output will contain a wide range from high-quality to highly noisy and incorrect facts. One may think that these HC errors could be compensated by large-scale crowdsourcing, with redundant HITs and statistic reasoning over many contributors. However, there is still the issue of the total cost: each HIT may cost a few cents only, but paying for hundreds of thousands or millions of HITs quickly becomes prohibitive.

1.3 Thesis Contributions

Addressing the challenges outlined above, this thesis presents methods that combine automatic IE with human computing, harnessing benefits of both while addressing their individual limitations. A brief synopsis of its contributions is presented here:

- **Architecture for Combining Human Computing with IE.** The first contribution of the dissertation is a system for knowledge acquisition, whose architecture combines automated extraction techniques with human computing inputs. Automated fact extraction methods use a combination of pattern-based, learning or reasoning approaches to achieve scalable extraction from textual sources. However the facts extracted may be noisy or incorrect owing to the complexity of natural language text. In our architecture, the extraction system combines statistical evidence in large corpora with semantic resources to generate large number of relational fact candidates. Moreover it couples this extraction phase with ranking of the candidates based on their likelihood to produce correct facts. These candidates form the basis of human involvement either as human intelligence tasks for crowdsourcing platforms (called HITs) or questions in a human computing game. The key idea is to *complement* high-recall fact extraction methods with high-quality human judgements, reaping benefits of both. This system architecture, named HIGGINS, was presented at the World Wide Web Conference 2013 [40].

- **Automated HIT Generation for Crowdsourced Fact Acquisition.** The second contribution of the thesis is a principled framework for discovering and extracting relationships between entities in narrative text. Based on the HIGGINS architecture, our system consists of an Information Extraction (IE) component and a Human Computing (HC) component. The IE component compiles entity-name and relation phrase dictionaries making use of methods in computational linguistics and additional heuristics. To overcome sparseness issues, it additionally assimilates phrases from semantic resources, taps into co-occurring phrases in large Web copora, and combines all of them using a mixture model. Taking into account the textual contexts of the entities, it ranks the phrases in the relational lexicon using a statistical translation model. The HC component aggregates and diversifies these candidate entity-pair relation triples to generate questions with relevant multiple-choice answers. These questions, casted as HITs, are evaluated by workers of crowdsourced platforms to generate final facts. Through extensive experiments we establish the high quality of the facts generated by the system. When compared to state-of-the-art fact extraction systems, our system delivers superior performance, especially on recall. By tapping into statistics derived from large corpora and semantic resources, and through judicious ranking, this system substantially reduces money costs at the crowdsourcing stage, which can be prohibitively expensive when pure HC is employed. This work was presented at the IEEE Conference on Data Engineering 2014 [41].

- **HIGGINS Games for Knowledge Acquisition.** The third contribution of the thesis is a framework for designing human computing games for KA. The HIGGINS architecture allows for wrapping various KA tasks as factoid-based question answering games. To demonstrate the viability of HIGGINS methods for relation-oriented fact extraction, we built the single-player MOVIEWIZARD (BOOKWIZARD) games, and the two-player MOVIEGURUS game. These games are designed to acquire relations between characters in movie and book plots. As part of the game play, the player's objective is to gain high scores by correctly pointing out the relations, interacting either with the system or with a human partner. As a by-product of playing the HIGGINS games, new facts are acquired and assessed. KA games based on HIGGINS were demonstrated at the ACM Conference on Information and Knowledge Management 2013 [42].

1.4 Thesis Organization

The rest of this thesis is structured as follows: **Chapter 2** summarizes the state-of-the-art on automatic extraction techniques for knowledge base construction, and outlines its limitations. It then presents background on the field of human computing and efforts made for knowledge acquisition through human computing. **Chapter 3** presents the architecture of HIGGINS, a system that couples information extraction with human computing to overcome their individual limitations; it outlines the key idea and the design principles, and introduces its components. **Chapter 4** discusses in detail the components that make up the HIGGINS system, and their implementation. **Chapter 5** provides detailed experiments on the performance of HIGGINS and its components. **Chapter 6** discusses the design and construction of HC games for KA. It presents the MOVIEWIZARD, BOOKWIZARD and MOVIEGURUS games that enable extraction of relations from narratives. **Chapter 7** provides conclusions and directions for future work.

2 CHAPTER

Background and State-of-the-Art

2.1 Overview

Automated information extraction (IE) is at the heart of knowledge acquisition, allowing factual information to be harvested from a large variety of data sources [9–15]. However, even state-of-the-art IE methods often fall short in recognizing factual relationships expressed in natural language. Human computing (HC) is a relatively new field that casts hard computational problems as intelligence tasks for humans. Our work proposes the use of human computing to overcome challenges in extracting relational facts from textual sources. This chapter first provides background on knowledge acquisition and knowledge representation. Then it presents an overview on the existing work in IE and HC fields.

2.2 Knowledge Acquisition and Representation

In the field of Artificial Intelligence, knowledge acquisition (KA) refers to the transfer and transformation of problem-solving expertise from a knowledge source to a program. In order to enable a computer program perform meaningful inference, real-world knowledge is acquired from data sources; and subsequently structured and organized as inter-related concepts. These concepts and their relationships are stored as logical assertions in large information repositories called knowledge bases (KBs).

Ontology. In information sciences, ontologies refer to KBs that possess “semantic awareness” or the ability to perform logical reasoning with its underlying knowledge. Ontologies represent knowledge using machine-interpretable definitions of concepts, usually within a domain, and relations among them. Using rules, constraints, and procedures, they are not only able deduce new knowledge from what exists within them but also incorporate external knowledge. Ontologies form crucial components of natural language understanding and knowledge-based problem solving methods. There are widely used for query answering and information integration purposes. In scientific research they have been employed for problems such as word sense disambiguation, document classification, query expansion, natural language question answering and machine translation.

Ontology building is a difficult process and ontologies generally limit their knowledge to predefined scope of interest or *domain* (animal, disease, protein, bio-medicine etc.). Invariably the fundamental knowledge constituents of ontologies are *facts*, made up of *entities* and their *relations*. For example, an ontological fact ⟨Barack Obama –PRESIDENTOF→ USA⟩, has {Barack Obama, USA} as entities with the relation {PRESIDENTOF} between them. Ontological entities belong to one or more *classes*, such as politicians, presidents etc. We briefly describe these components of ontological facts here:

- ***Entities.*** In an ontology, all concepts, whether abstract or concrete, are represented as entities. Although there is no formal nomenclature, entity representations in ontologies are unique. This holds even if the entity in real-world is referred in various different ways. For example {*Bond*}, {*007*} or {*James Bond*} could refer to the one ontological entity James Bond. Nevertheless all entities in the ontology must be distinguishable from one another.

- ***Classes.*** Entities that share common properties are grouped into classes (for example presidents, scientists, buildings, rivers etc.). Every ontological entity belongs to one or more classes. Classes themselves are represented as ontological entities. Moreover subsets of classes are identified and represented using the subclass relation, for example presidents –SUBCLASSOF→ politicians, politicians –SUBCLASSOF→ persons and so on. All entities in a class are referred to as its *instances*.

- ***Relations.*** For many applications, relations between entities form the most useful source of knowledge in an ontology. Typically ontologies define set of relations that entities participate in, based on domain knowledge. ISA and SUBCLASSOF are most commonly found ontological relations. Any arity of relations can be supported, but in practice many current-day ontologies use binary relations. Each relation has domain and range to which the entities should adhere, for the ontological fact to be valid. For example, the relation BORNIN can have person and location as its domain and range. Relations themselves are entities in the ontology, and as with entities their representation is unique (WASBORNTO, GAVEBIRTHTO or HASPARENT could all be represented as PARENTOF).

2.2.1 Knowledge Base Construction.

Traditionally, KBs have been constructed manually, typically by experts from a domain. These KBs serve to support decision-making process of humans or machines. However using human experts to construct KBs is not a scalable approach which can be employed to create KBs that do not grow or update frequently. For example WordNet [43] is a human-compiled linguistic KB consisting of words in the English language, their senses and relationships. Other examples of similar KBs are OpenCyc [5], SUMO [44], UMLS [45] and GeneOntology [46].

To build general purpose domain-independent KBs that contain large number of concepts and their relations, human effort has to be minimized and automated methods have to take over. Towards this end many automated IE methods have been developed which follow different strategies based on the input they start with. Several automated methods exploit semi-structured sources like Wikipedia to create huge KBs such as DBPedia [3] and YAGO [4]. Some approaches focus on extracting concepts and relationships directly from textual sentences. DIPRE [11], Snowball [12], KnowItAll [47], TextRunner [48], ReVerb [17], OLLIE [49] are examples of automated text-based IE approaches, some of which we discuss in detail in Section 2.3. Other approaches like PORE [50] and Kylin [51] combine and consolidate extracted information from semi-structured text with extractions from natural language text.

If costs were not an issue, allowing humans to participate directly yields KBs of a higher accuracy. However this requires uniformity checks and quality control measures. For example, Freebase [1] harvests data from Wikipedia, MusicBrainz and other directories, as well as individually contributed data from its users.

2.2.2 Knowledge Representation

There exist numerous models that encode ontological knowledge. Among them, the Resource Description Framework or the RDF data model, standardized by the World Wide Web Consortium, is widely in use today. This model defines the notion of resources and uses Unique Resource Identifiers (URIs) to identify them. RDF supports statements about resources in the form of subject-predicate-object (SPO) triples. From ontological perspective each triple denotes a fact. For example, one way to represent the assertion "Microsoft has headquarters in Redmond" as an RDF triple is: Microsoft –HASHEADQUARTERSIN→ Redmond. RDFS (RDF Schema) extends RDF to incorporate type information. Hence classes become resources in RDFS, allowing statements about classes and sub-classes. Web Ontology Language (OWL) further extends RDF and RDFS by providing terminology so that classes and their properties can be defined.

The RDFS/OWL model is simple and expressive enough for many purposes but there is no built-in representation for additional information such as extraction source, time-period of validity or confidence scores. As a consequence, many systems developed their own data models by extending RDFS/OWL model to suit their needs [4, 52].

2.2.3 Tasks for Ontological Fact Acquisition from Text.

IE techniques help build large ontologies with little or no human intervention. In transforming text to ontological facts, there are several steps, each of which has its own challenges. We list them below.

1. **Entity Extraction**: There are three steps performed to map textual strings to entities in the ontology. First, the IE method identifies character sequences as

mention(s) of an entity and classifies them into members of semantic classes such as persons, locations, organizations and so on (*named entity recognition*). Second, alternate references to entities such as pronouns and other references are identified (*coreference resolution*). Finally, entity strings and their references are mapped to the corresponding entity in the ontology (*named entity disambiguation*).

2. **Relation Extraction**: The different relationships that entity pairs participate in are extracted from textual sentences and stored as relational facts in the ontology. Naturally this task assumes that the necessary entity recognition, anaphora resolution and entity disambiguation has been performed beforehand.

2.3 Techniques for Discovering Relational Facts in Text

In this section, we present prior methods for relational fact discovery from text. These techniques work on a best-effort basis, dealing with varying nature and size of the input data, balancing precision and recall, while tackling issues of efficiency and scalability. Their underlying methods are based on rules and patterns (e.g., [53–56]), linguistic analysis (e.g., [17, 57–59]), statistical learning (e.g., [6, 7, 18, 19, 60, 61]), consistency reasoning (e.g., [62, 63]) and often combinations of all these elements. Traditional methods focused on harvesting tuples that satisfy prespecified relations from a domain, while the more recent OpenIE systems are data-driven, aiming to capture all relational tuples from heterogeneous sources such as the World Wide Web. We briefly discuss some of these methods below.

2.3.1 DIPRE

Dual Iterative Procedure for Relation Expansion (DIPRE) [11] is a seminal method for finding relations on Web-scale corpora. The method relies on the duality of relations and their occurrence patterns. An iterative procedure is followed for discovering relations in the following manner: Starting with a bootstrap tuple-set of the target relation, occurrences of all tuples in the corpus are extracted along with their context. In the next step, patterns are generated based on the tuple occurrences and their surrounding contexts. Finally, the sample is expanded with the newly found patterns and searched again. Exploiting the redundancy and structure in large document collections, and through careful expansion of patterns, DIPRE retrieves tuples that participate in a target relation with high quality.

As the above approach requires no supervision, we use a similar iteration procedure in HIGGINS to mine relational phrases from large corpora. However, we address two main challenges that DIPRE faces. As DIPRE benefits from redundancy of structure, its full potential can only be realized by running it over large text collections. Therefore, to overcome scalability bottlenecks, HIGGINS uses a Map-Reduce style implementation. Another drawback with DIPRE is that the patterns during expansion phase can produce noise and hence digress significantly from the target relation. HIGGINS adds syntactic constraints on the patterns to get rid of noisy patterns, and performs one iteration only so as to avoid unrelated patterns.

2.3.2 Snowball, StatSnowball & Espresso

DIPRE's approach of iterative extraction with bootstrapping was adopted by many systems given its benefits of minimal supervision. All these methods try to address the tricky issue of pattern/tuple selection during the expansion phase. Ideally the newly-found patterns during expansion need to have high *coverage*, and at the same time be *selective* to avoid incorrect tuples. Snowball [12], StatSnowball [61], Espresso [64] and PORE [50] are notable works that deal with this issue.

Snowball [12] tries to achieve coverage by using type-lifted entity arguments (e.g., LOCATION based ORGANIZATION instead of Irving based Exxon Corp.). For selectivity, the components of the patterns are represented in vector-space using occurrence frequency as weights, and near-similar patterns are clustered. Similarly, tuples that generate noisy patterns are avoided by assigning confidence scores, based on selectivity and the number of patterns that generated them. Such control mechanism increases the overall recall while retaining the precision. However, pattern selectivity through entity type-lifting helps in case of certain type combinations (e.g., ORGANIZATION-LOCATION), while for some combinations it does not work well (e.g., PERSON-PERSON, which can produce large number of different patterns). HIGGINS instead uses noun phrases rather than typed-lifting to achieve high recall.

The StatSnowball [61] system which improves over Snowball defines patterns over features instead of matching terms, and uses an L^1-norm regularized Maximum Likelihood Estimator for pattern selection. The method, however, uses shallow features and can miss relations when presented with sentences of arbitrary complexity. On the other hand HIGGINS does not depend on word-level features but employs dependency parsing. Also it does not discard patterns but uses ranking model for phrases to narrow down to the correct relation.

The Espresso [64] algorithm follows a conservative expansion approach by ranking patterns and including top-k patterns, incrementing k in each round. The patterns are generated by querying web search engines with wild-card expansions. Generic patterns that have high coverage but low precision are dealt with using Point-wise Mutual Information (PMI) scores, a collocation measure between seed and the newly-found patterns. However, it is not clear if PMI scores, or collocation measures in general, boost good but rare patterns; and scaling to many relations remains a question.

2.3.3 SOFIE & PROSPERA

SOFIE [63] extracts ontological facts from natural language documents. Starting with a set of target relations, it extracts patterns and performs consistency checks using a weighted MAX-SAT model to ensure correctness of the extractions. The system takes into account type-specific, functional and domain rules to verify newly acquired knowledge against the existing facts in the ontology. Although this results in very precise facts, SOFIE is very slow in practice and does not scale to Web-size corpora. PROSPERA [65] extends SOFIE's pattern analysis with n-gram itemsets and carries over the confidence scores from this phase as weights to the reasoner's input. Both phases are organized to support distributed processing for better scalability. The SOFIE/PROSPERA approaches, however, require target relations to be known upfront.

2.3.4 TextRunner

TextRunner [48], the first of systems developed under the OpenIE paradigm, pursues an aggressive approach by capturing verb-centric relations between pairs of noun-phrases. Its design does not require knowing relations upfront and brings higher scalability as it is able to efficiently extract relational facts from Web-scale corpora. In the first of its three-step process, TextRunner builds a self-supervised Bayes classifier that runs on dependency parse tree paths built from sentences, and assigns "trustworthiness" to extractions. In the second step, it performs a single pass on the corpus to pick candidate tuples, and labels them as good or bad using the classifier built in the first step. Finally, the system retains tuples that are labeled good and have high corpus frequencies. The downside of TextRunner's approach is that labeling needs a sufficiently large sample. Moreover word-level trust labels cannot be directly extended to phrases as the trustworthiness decreases with its length, irrespective of words that comprise them (e.g., Bill –SUCCEEDED⟶ Hillary and Bill –SUCCEEDED IN NOMINATING⟶ Hillary).

2.3.5 ReVerb

ReVerb [17] extends TextRunner by enforcing syntactic and lexical constraints on patterns to reduce noisy and incoherent patterns. Syntactic constraint enforces a verb-phrase based regular expression over parts-of-speech tags to eliminate noisy and uninformative patterns. In addition lexical constraint gets rid of very specific phrases by looking at frequency counts over a global corpus. Finally, ReVerb trains a logistic classifier to assign confidence scores to the extractions, allowing it to retain precise extractions and trade-off recall if necessary.

Experimental evaluation shows that by considering only verb-centric phrases, ReVerb does not lose out on too many relational extractions. However ReVerb does not capture noun-centric relations, or relations spread over long range in sentences. For complex sentences it may capture factual assertions with incomplete relational patterns. The authors also report that incorrect identification of arguments leads to high error rate in Reverb, either by missing the extraction or producing incorrect extractions. Despite all these drawbacks, the ReVerb method scales well, provides high recall and does not make any assumptions on the input text.

2.3.6 PATTY

The PATTY [66, 67] system builds a large lexicon of binary relations which are semantically typed and organized into a taxonomy of semantic subsumptions. PATTY enriches pattern representations with syntactic, lexical and ontological (SOL) information (POS tags, wild-cards and ontological types of the arguments — for example ⟨person⟩'s [adj] performance * ⟨event⟩). PATTY defines syntactic matches to the SOL pattern as textual strings that can be mapped to pattern in an order-preserving manner. Also it establishes a notion of semantic generalization of SOL patterns by looking at the set overlap of their entity arguments. The SOL pattern model is able to capture fine-grained relations, enabling pattern generalization for high coverage and efficient processing for scalable mining.

PATTY also constructs a taxonomy of subsuming patterns, looking at set of argument pairs of each the pattern. However it avoids pairwise comparison of set of argument pairs by constructing a prefix-tree of these sets, where patterns with common argument pairs have the same prefix path. It also adds node-to-node links across different paths, connecting nodes that have the same argument pair. By traversing the prefix-tree bottom up following these links, all paths that are subsumed by each pattern are mined, yielding a taxonomy of patterns.

2.3.7 OLLIE

OLLIE [49] system builds pattern templates on dependency tree paths. This allows OLLIE to capture noun-form and adjectival patterns as well, unlike ReVerb and TextRunner, that only capture relations mediated by verbs. In addition, by allowing both syntactic and semantic/lexical patterns over dependency paths, and performing relaxed pattern matching, OLLIE yields high recall while maintaining good precision. Another feature of OLLIE is the ability to detect non-factual assertions, which is achieved by performing context analysis (e.g., in the case of "Early astronomers believed that *earth* is the center of the *universe*."). These non-factual assertions can be attributional, hypothetical or conditional. To detect attributional relations, the contextual verb in the dependency parse is matched to a list of communication and cognition verbs in VerbNet [68] such as {"*believe*" or "*talk about*"}. In addition the prefix of the adverbial clause modifier in the parse structure is checked against terms such as {if, although, when, because etc.}. While these measures cover most cases, a supervised logistic classifier (trained over extractions from Wikipedia, news and biology) is used to reduce confidence scores if the context is not likely to be factual.

OLLIE achieves higher yield compared to ReVerb-like systems owing to use of dependency parse structures, allowing relational terms to positionally occur outside their arguments. On the flip side, the approach heavily relies on dependency parsers, which are prone to errors (over 30% of their errors in their experiments). Other sources of errors are aggressive generalization of patterns, errors in detecting contexts, and inability to detect n-ary relations. Nonetheless, OLLIE obtains a higher number of precise extractions than existing state-of-the-art OpenIE systems. Moreover, so far it is the only automatic OpenIE system that detects non-factual assertions which are only hypothetically or conditionally true.

2.4 Human Computing Systems

Human computing (HC) refers to the paradigm of designing procedures that employ large crowd of users to solving computationally hard problems. This underlying idea is to leverage the innate cognitive and intellectual capabilities of humans for certain tasks, especially in cases where algorithmic solutions fare poorly; for example, tasks that involve understanding content in images, audio or videos. Human computing is increasingly being applied in areas that include, but not limited to, information accessibility and security, computer vision, content recognition, data filtering, and translation.

HC has generated massive interest in research communities given its wide range of applications. Academics in the areas of data and knowledge management have started principled research on how to systematically exploit collective human intelligence [21, 22]. Recent works have studied application of HC in quality assessment of rankings, summaries, and IE results [21, 25, 26], query answering [27–29], entity matching [30, 31], ontology alignment [32], image search and annotation [33–35], and acquiring commonsense properties [36, 37] and taxonomies [38].

User participation in human computing tasks needs incentives. Three main streams of incentivization exist: Most prominent stream is of users who earn direct incentives in the form of monetary payments for working on tasks requested by others over online "crowdsourcing" platforms. Secondly altruistic Web users motivated by information sharing, collaborate to create and maintain online knowledge repositories and make them available for everyone's use. Thirdly, HC games can be designed that disguise tasks, soliciting user's contributions implicitly and exploiting their desire to be entertained. Classified based on the type of incentivization, we discuss here the important HC works and their attributes.

2.4.1 Crowdsourcing

Crowdsourcing refers to the practice of employing a large pool of humans, typically from online Web users, to work on microtasks – tasks that can be done by any user in a short amount of time (seconds or few minutes). In crowdsourcing terminology, microtasks are often referred to as Human Intelligence Tasks or HITs and the users as "workers" or "turkers". Typical HITs are labeling, ranking or classifying items, recognizing objects in multimedia, answering survey questions, or testing websites. More involved HITs include transcribing audios or videos, summarizing text passages, or developing and correcting text descriptions. Upon completion of one or more HITs, the worker gets micropayments in the tune of cents or few dollars. Most HITs are designed so that they can be accomplished by any worker in the population with no assumption on their background knowledge on the subject. Easy-to-accomplish (and high paying) HITs attract a larger subset of the worker population and result in lesser turn-around time for the requester.

Many crowdsourcing platforms have emerged that enrol internet population as workers for crowdsourcing tasks. There are many benefits of the requester-worker symbiotic environment that these platforms provide. The requesters benefit by bypassing the tedious process of recruiting employees, while the workers can choose and work on tasks that suit them. However the contrasting goals of the requester trying to acquire good results, and the workers maximizing their earnings has certain pitfalls. Relying on a single worker often leads to low quality or erroneous solutions. Crowdsourcing platforms employ two strategies to overcome this effect. They produce redundant HITs and aggregate the workers' inputs (here the workers are chosen randomly to minimize collusion). Majority voting is the de-facto method for this purpose; different variants of this method and other aggregation methods are subject of active research [69, 70]. Another widely employed strategy is to assign trust or confidence scores to workers and weed out solutions of unreliable workers by subjecting them apriori to control tests [71].

Amazon Mechanical Turk [23] and CrowdFlower [24] are forerunners among over 1000 commercial crowdsourcing platforms that exist today. They employ millions of workers world-wide for tasks in areas related to design, data gathering, idea generation, search and surveillance, documentation and proofreading, and many more [1]. Given the wide range of its application areas, crowdsourcing has generated massive interest among researchers [72] and has been the focus of both DB and IR communities for the past few years.

- **Applications in Vision.** Vision applications that require detecting objects and patterns in images and other multimedia have greatly benefited from crowdsourcing. Techniques to use crowds for labeling images for content search [35, 73], correcting OCR errors, annotating video content for transcription, classification [74] and detecting events [75] have been proposed. Most prominent among these systems is reCaptcha [76] which uses OCR errors to discern humans from programs. Users who correctly identify these errors are allowed access to the Web resources that use this tool, resolving OCR errors as a by-product.

- **Applications in Database Management.** Within data management, CrowdDB [27], Qurk [77, 78], and [28, 29] attempt to extend database systems by incorporating crowd functionality. CrowdDB augments databases with new data that humans can easily find (say utilizing search engines) and aims to answer fuzzy-comparison queries, such as different text sentences or photos referring

[1] http://en.wikipedia.org/wiki/List_of_crowdsourcing_projects

to the same entity. It models human participation in the task of query answering, extending SQL with operators reflecting these data-adding and comparison needs. Qurk [77] aims at managing the work-flow and the returned answers by humans to HITs. It extends SQL and represents human answers as multivalued attributes and aggregation functions are defined (such as majority voting) to declare the eventual single attribute value. It also allows the addition of new operators, such as filters, sorts, and joins. For instance, join operators are employed whereby humans are called to find the same entities in different "tables" (e.g. identify the same person in two different sets of photos). In [78] authors show how the human-based sort and join operators are implemented and optimizations are presented regarding the generation of HITs and their costs. Similarly, [28] attempts to involve humans in data management tasks, presenting a new query model which allows for predicates to be evaluated by humans, the DB, and/or external algorithms.

- **Applications in Information Retrieval.** Crowdsourcing is increasingly being employed also for Information Retrieval tasks [25, 26]. CrowdSearch, [34], for example, tackles the problems associated with the low-quality results achieved when searching for images, especially when using mobile phones where image quality is typically poor. It adds a human-validation phase, whereby the AMT infrastructure is employed and human intelligence improves search result quality. CrowdSearch addresses the issues of selecting the results for which validation is sought and how to validate them, using models for the delay-accuracy-cost trade-offs involved in the crowdsourced tasks. Crowds have also been successfully employed for relevance assessments: Alonso et al. [25] show that crowdsource workers can perform relevance assessment as well as TREC experts.

2.4.2 Collaborative Knowledge Building

With increasing usage of the World Wide Web in the recent years, there has been tremendous development in internet-based platforms for collaborative sharing of knowledge.

Wikipedia. Wikipedia, a collaboratively-edited online encyclopedia, is one of the largest and most popular efforts in this direction. As of date, it contains over 30 million articles in 287 languages (over 4.3 million in English) – written, structured, and edited by volunteers on the Web. Its high quality, freely available and dynamically-updated content is widely used for research purposes. Though rendered in HTML, Wikipedia articles are authored in the *Wiki markup language* which provides a loose structure and makes its content particularly amenable for harvesting [3, 4]. There are various informative components of a Wikipedia article that can be directly tapped into for extraction. A typical article contains summarized information on important attributes of the entity (such as birth-date, profession, nationality for articles of the PERSON class) in a table called *infobox*. The rest of the article contains information in the form of unstructured text organized into sections. Often they contain lists and tables. Important entities in the text are linked to their respective articles through Wikipedia *intra-wiki* links. Each article is also assigned *categories* based on the topics to which it belongs. Each article is linked to its equivalent in other languages through *inter-wiki* links. Different surface string formulations are manually disambiguated to point to the correct article (for e.g., {*Obama*} & {*Barack Obama*} point to the same article while {*Amazon (river)*} & {*Amazon (warrior)*} point to different articles). While Wikipedia infoboxes, links and categories have been extensively utilized for fact harvesting, exploiting their rich textual content is still an active area of research.

Wiki Farms & Community Wikis. Platforms such as wikia.com, sparknotes.com, cliffnotes.com etc. enable close-knit communities to share information from specific domains such as movies, video games, history, sports or literature. These repositories vary in richness and organization depending on the community size, time period and existence and update activity. Like previous systems, HIGGINS exploits community-gathered knowledge for its purposes. In HIGGINS we harvest movie articles from Wikipedia and book story articles from Sparknotes [79].

2.4.3 Human Computing Games.

Human computing games, pioneered by Luis von Ahn [39], transform or mask computationally challenging tasks into interesting games. Here humans inadvertently indulge in solving instances of a computational problem, only to fulfill their desire to be entertained. The outcome of these games help gather data, which is otherwise hard to acquire (for example, by applying computer vision techniques). Such data is invaluable and can directly serve for various purposes (e.g., image search) or indirectly as training data for machine learning approaches.

Luis von Ahn et al. introduced “Games With A Purpose” or “GWAP” [39], and proposed several games that aid in solving hard AI problems such as image labeling, object recognition and commonsense fact acquisition. In all of these games, the game interaction is tied up with the task to be accomplished. During the interaction, the player tries to achieve a fully specified goal following pre-defined rules which are carefully designed to partially or fully solve the computational task. Three general strategies for game play were outlined under GWAP:

a. **Output-agreement Games:** Two randomly paired players are given the same input and the goal is to produce the same output. ESP game [33] (later adopted as Google Image Labeler), is a popular game that adopts this strategy. The system shows an identical image to a pair of players and they provide keywords description about the image. The game is won if a common keyword is provided by both. These keywords are used to tag the input images and consequently utilized for enhancing Web image search.

b. **Input-agreement Games.** Two randomly paired players are given individual inputs known only to the system. They are shown the data generated by their each other and the goal is to verify if the system provided them with the same input. For example, in TagATune [80], two sound clips are used as inputs and the players describe to each other their content or features. These descriptions form important audio metadata and can consequently be used for audio search and retrieval.

c. **Inversion-problem Games.** Two randomly paired players are given different roles. A player under the ‘narrator’ role is given the input which he describes to the partner, who assumes the ‘guesser’ role. The goal is achieved when the guesser correctly identifies the input based on the descriptions created by the partner. The Verbosity [36] game follows this strategy. In each of several rounds, the narrator is given an object (e.g., apple, grass, milk etc.) based on which he fills in template sentences. Using these sentences as hints the guesser tries to detect the object. Both players win the round when the guesser is successful in identifying the object. The system uses the sentence templates to construct common sense facts.

2.5 Human Computing for Knowledge Acquisition

Finally, with respect to knowledge acquisition, crowdsourcing techniques also have enjoyed successes. An early representative example are common sense knowledge bases, with the Open Mind Common Sense (OMCS) project [81], being a prototypical example. OMCS relies on volunteers to provide statements of common sense regarding real-world objects, people, and events. In parallel to such 'brute-force' crowdsourcing efforts, games with a purpose, have shown that knowledge can be acquired, as a 'side-affect', when humans play carefully designed games. Verbosity [36] is an interesting example game, which was successful in terms of both the number of people players and the knowledge acquired. Interestingly, OMCS and Verbosity have been combined [37] in order to increase OMCS's 'entertainment value' and thus its human involvement. This is facilitated by the fact that the statements acquired by Verbosity resemble greatly the facts in ConceptNet [82] (OMCS's semantic network representation).

Very recently, crowdsourcing has also been employed for the creation of taxonomies [38], collecting seed facts as labeled training data [83], entity resolution [31], and ontology alignment [32]. However, all above efforts rely solely on human input for knowledge acquisition.

3 CHAPTER

The HIGGINS System for Combining Information Extraction and Crowdsourcing

3.1 OVERVIEW

In this chapter, we present an architecture that combines techniques for automated information extraction (IE) with human computing (HC), for acquiring knowledge base facts. Termed HIGGINS, this architecture blends an automated IE engine with a crowdsourced-based (or game-based) HC engine. The IE engine harvests fact candidates from free text by compiling large entity and relation lexicons and using statistics and semantic resources for relational phrases. From the large pool of generated fact candidates, the HC engine uses ranking based on statistical language models to provide most likely candidates for human judgements, casting them either in the form of crowdsourced HITs or questions of a HC game.

3.2 ARCHITECTURE OF HIGGINS

The key idea of HIGGINS is to use automatic IE to generate questions and relevant candidate answers (for multiple-choice questions), enabling effective collection of human inputs. Our expectation is that this can improve the quality of user contributions and reduce the overall cost of crowdsourcing. Therefore in HIGGINS ,

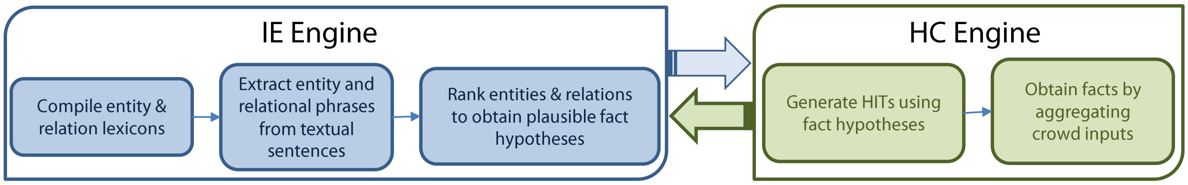

IE Engine
Compile entity & relation lexicons
Extract entity and relational phrases from textual sentences
Rank entities & relations to obtain plausible fact hypotheses
HC Engine
Generate HITs using fact hypotheses
Obtain facts by aggregating crowd inputs

FIGURE 3.1: HIGGINS workflow

the strategy for acquisition of entity-relation triples has two components (shown in Fig. 3.1):

1. IE engine: We employ automated IE on Web corpora, in order to derive candidates for entity-relation-entity triples, with an open set of potential relations. We use a suite of techniques from computational linguistics, including dependency parsing (with the Stanford Parser) and pronoun resolution (with our own customized method). The resulting triples are usually of mixed quality, necessitating the second stage.

2. HC engine: The sets of candidates from the IE engine and their underlying patterns are then used to generate HITs in game form. Abstractly, each HIT presents the user with a *knowledge quad* of the form $(c, e1, r, e2)$ where $e1$ and $e2$ are entities, r is a relation, and c is a cue or textual context. One or more of the components c, $e1$, r, and $e2$ can be empty slots (variables) to be filled by the user; we may present a multiple-choice list to the user to pick the missing value. The quads are presented in the form of questions, with relevant candidate answers and additional free-text fields for entering further values. We formally define the notion of knowledge quads in the following section.

3.2.1 Definitions.

HIGGINS generates fact candidates from text through automated IE and validates them through HC producing factually correct knowledge triples and quads. In this work, we look at the following definitions for knowledge triples and quads.

Definition 3.2.1. A KNOWLEDGE TRIPLE is a triple $\langle e1, r, e2\rangle$ consisting of entities $e1$ and $e2$ and a relation r, where each of the three slots may also be a variable (starting with a question mark and indicating a missing value).

Definition 3.2.2. A KNOWLEDGE QUAD is a quadruple $\langle c, e1, r, e2\rangle$ where c is a *context entity* and the other three components form a knowledge triple within the scope of c. All slots may also be variables.

In both triples and quads, HIGGINS allows *semantic classes* in the place of the $e2$ entity. In this case, r is the *type* relation (an example being *Michael_Corleone type MafiaGangster*).

Definition 3.2.3. BOUND/UNBOUND SLOTS. The IE engine of HIGGINS generates fact hypotheses as quads $z = \langle c, e1, e2, r\rangle$, where one or more of c, $e1$, $e2$ and r can be bound. A slot is bound if the HIGGINS IE engine extracts a candidate value for the slot. For example, $i = \langle$ *The Godfather, Michael Corleone, Vito Corleone,* r $\rangle$ has c, $e1$ and $e2$ bound and r is unbound.

In our notation, a quad instance and its constituent slots are denoted using small case and bold-face shows bound items. For example, a quad instance $z \in Z$, $z = \langle \mathbf{c}, \mathbf{e1}, \mathbf{e2}, r\rangle$, has $\mathbf{c}$, $\mathbf{e1}$ and $\mathbf{e2}$ bound and r is unbound.

3.2.2 Functionality of HIGGINS Components.

The HIGGINS IE and HC engines in combination take textual sources as input and produce knowledge quads by formulating acquisition or validation tasks. In this section we outline the steps followed and provide an overview of the functional integration of HIGGINS ’ components.

In the following procedures we focus on the acquisition task where the relation is unbounded. For this task, the HIGGINS IE engine takes as input the narrative text and produces fact hypotheses H. These fact hypotheses are fed to the HC engine and transformed into fully qualified knowledge quads via human inputs.

In the IE stage, the procedure shown in Higgins.IEEngine is followed for the relation extraction task shown as t. For each context (for example, a movie, book or a news

Procedure Higgins(t)

```
Input: Example Task $t : \langle \mathbf{c}, \mathbf{e_1}, \mathbf{e_2}, r \rangle$, Narrative Collection: $M$
Output: Knowledge quad instances $K$
$H \leftarrow$ Higgins.IEEngine $(t, M)$;
$K \leftarrow$ Higgins.HCEngine $(H)$;
return $K$;
```

article) in the narrative collection M, we retrieve from the entity phrase lexicon, the entity phrases that are of interest in the context. From the narrative text, other entity phrases occurring in the same sentence are paired. Based on the text in these sentences, a set of relation candidates are chosen from the dictionary of relations based to their relevance to the text. The resulting set of fact hypotheses H are collected and passed to the HC engine, where each $h \in H$ is a quad comprising of the context, the entity pair and candidate relations that potentially apply to the pair.

Procedure Higgins.IEEngine(t,M)

```
Input: Task $t : \langle \mathbf{c}, \mathbf{e_1}, \mathbf{e_2}, r \rangle$, Narrative Collection: $M$
Output: Set of HITs $H$
$H = [\,]$
for $c \in M$ do
    $E \leftarrow$ getEntitiesOfInterest $(\mathbf{c})$
    for $e_1 \in E$ do
        $e_2 \leftarrow$ getPairingEntity $(\mathbf{c}, \mathbf{e_1}, E)$
        $R \leftarrow$ generateRelationCandidates $(\mathbf{c}, \mathbf{e_1}, \mathbf{e_2})$
        $h \leftarrow \langle \mathbf{c}, \mathbf{e_1}, \mathbf{e_2}, R \rangle$
        $H \leftarrow H$.append $(h)$
return $H$
```

The HC engine, shown in Higgins.HCEngine, generates natural language questions Q on the basis of the fact hypotheses H. These questions when ratified via a human input become fully qualified knowledge quads K.

Procedure Higgins.HCEngine(H)

```
Input: HITs : $h \in H$, where $h = \langle \mathbf{c}, \mathbf{e_1}, \mathbf{e_2}, R \rangle$
Output: Knowledge quad instances $K$
$K = [\,]$
for $h \in H$ do
    $q \leftarrow$ generateQuestion($h$)
    $\mathbf{h} := \langle \mathbf{c}, \mathbf{e_1}, \mathbf{e_2}, \mathbf{r} \rangle \leftarrow$ obtainHumanInput ($q$)
    $K \leftarrow K$.append ($\mathbf{h}$)
return $K$
```

3.3 DESIGN OF HIGGINS COMPONENTS

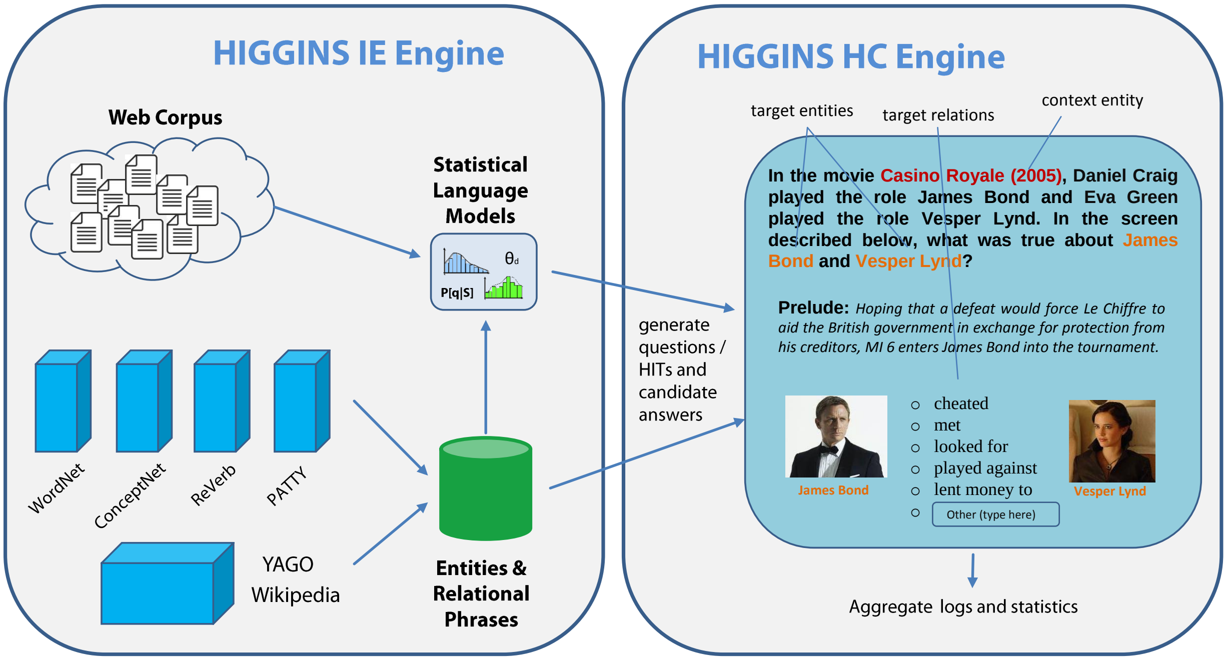
In designing HIGGINS components, two important issues need to be considered. Firstly the IE engine should generate all likely fact hypotheses taking into consideration the diversity and sparseness of relational phrases. Secondly, in order not to overwhelm the users at HC stage, the IE engine needs to come up with only a handful of relevant fact candidates for HITs. Therefore we follow the design principles of HIGGINS listed below:

1. The IE engine is tuned to work aggressively (aiming for high recall), capturing as many relational patterns as possible, and we expand this set by specifically designed *statistical (translation) language models (LM's)*.

2. We use statistics and heuristics to generate interesting questions about *important* entities and *salient but not obvious* relationships. Candidate answers for multiple-choice input are judiciously *ranked*, using corpus-collected statistics. An additional *diversification* step serves to avoid boring the user with near-duplicate choices.

3. The statistically derived relational phrases for candidate answers are complemented by phrases from *semantic resources*, specifically WordNet [84], ConceptNet [82], ReVerb [17], and PATTY [66]. All this information is combined by a mixture model that generates, expands, and ranks relationships for a given context using a statistical language model.

Figure 3.2 depicts the **system architecture** and main components of HIGGINS. It also shows a sample question from a game instance.



In the movie Casino Royale (2005), Daniel Craig played the role James Bond and Eva Green played the role Vesper Lynd. In the screen described below, what was true about James Bond and Vesper Lynd?

Prelude: *Hoping that a defeat would force Le Chiffre to aid the British government in exchange for protection from his creditors, MI 6 enters James Bond into the tournament.*

FIGURE 3.2: Overview of the HIGGINS Architecture

3.4 HIGGINS IE ENGINE

The HIGGINS IE approach follows the rationale of *Open IE* [6, 16, 17, 20] to achieve high recall. The IE engine aggressively gathers noun phrases for entities and verb-centric or role-centric phrases for relationships. Given a corpus, e.g., a set of movie narratives, we process all its documents in three phases:

1. identifying entity occurrences,
2. gathering relational phrases from sentences that contain two interesting entity names, using light-weight NLP techniques like part-of-speech tagging,
3. pruning the set of potential candidate relationships, by using more expensive NLP techniques like dependency parsing.

3.4.1 Entity occurrences

Entities. We consider all proper noun phrases (names, used without article) as entities. In addition, we consider a specific set of general noun phrases and occurrences of personal pronouns. If possible, we relate these phrases to canonical entities registered in a knowledge base like Freebase or Yago (or to the lists of actors, characters in Wikipedia/IMDB). The phrase-entity pairs form the entity part of the HIGGINS dictionary.

Entity descriptions. In addition, we have identified general noun phrases near each character and mapped them to the type system of the Yago knowledge base, using the same heuristics that Yago uses for mapping Wikipedia category names onto WordNet synsets [4, 85] and keeping only subtypes of the Yago class *person*. This way, we have compiled knowledge triples that provide us with fine-grained types of characters such as ⟨ *Captain Jack Sparrow, isa, pirate lord* ⟩ and ⟨ *Will Turner, isa, blacksmith* ⟩. This gives us a way of connecting role names occurring in narrative texts, e.g., "...the pirate lord", to the corresponding entities.

Pronouns. Finally, we have developed a heuristic method for resolving personal pronouns in such texts. We focus on pronouns in singular form, and we exploit gender information for characters. We map a male (female) pronoun to the closest preceding male (female) name occurring as grammatical subject. The gender properties for entities are derived from character descriptions using a classifier (over features such as frequencies of male vs. female pronouns, prior information on male vs. female first names, etc.). All these techniques serve to increase the number of sentences that we can tap into for extracting relationships. We relay this gender information to state-of-the-art co-reference resolution tool [86] to increase its resolution accuracy. If any errors are produced at this IE stage, the HC stage serves to capture them. Without these additional steps, the number of sentences that contain two explicitly and fully named entities and an interesting relation would often be disappointingly low.

These techniques can be applied, with minor variations, to other Web sources, and can be generalized to other kinds of entities. For example, for people's biographies as context entities, we can extract surface names of related entities from href anchor texts in Wikipedia and other sources.

3.4.2 Relations and Relational Phrases

As surface expressions for binary relations, HIGGINS uses lexico-syntactic patterns like verbal phrases occurring in sentences that mention two entities. For mining these, we give ourselves a head-start by using verbs from WordNet [43] and ConceptNet [82], and more general phrases from ReVerb [17] and PATTY [66].

These semantic resources already provide us with a fairly big set of phrases. However, when considering human relationships in movies or books, one sometimes comes upon very special phrases. Therefore, we manually compiled a set of 500 relations, and mined additional paraphrases through co-occurring entity pairs in the Wikipedia corpus. We limit phrases to one of the following two cases:

i) verb constructs or verbal phrases ending with a preposition (e.g. "*falls in love with*"), or

ii) noun phrases ending with a preposition (e.g., "*wise counselor of*").

Examples of knowledge triples with phrase-relation pairs obtained this way are:

⟨ "had an affair with" means romance_with ⟩,
⟨ "stabbed from behind" means attacked ⟩,
⟨ "cut his head off" means murdered ⟩.

We have a total of 116,471 relational phrases compiled in this way. We also use the co-occurrence frequencies of phrases with entity pairs in the Wikipedia corpus to estimate the *semantic relatedness* between phrases, which is later used by the HC engine for ranking and diversification in questions and candidate answers. This statistical estimation is discussed in detail in Section 4.5.

3.4.3 Pruning of Candidates

By detecting entity mentions in sentences and matching phrases from the dictionary in the same sentences, we can compile a huge set of candidates for relationships between entities. This gives us a very rich pool for generating HITs at the HC stage.

However, the ratio of spurious relation instances is very high. This is due to the complexity of the input sentences and the aggressiveness of our IE methodology. We experimented with more conservative approaches as well, but they tend to miss out many interesting candidates and lose big in recall.

To reduce the number of false positives, we employ a more expensive post-processing phase on the collected relationship candidates by running a dependency parser on all relevant sentences. Specifically, we use the Stanford Parser [87, 88] and identify all dependency paths in a sentence connecting a pair of entities that occurs in a candidate from the previous phases. If no such path exists, the candidate is discarded. Otherwise, we retain all candidates whose relational phrases overlap with words on at least one of the dependency paths. For example, in the sentence "Le Chiffre abducts Vesper Lynd and uses her as bait to capture James Bond.", to detect the relation between Le Chiffre and James Bond, we keep the top-k relations that match the dependency path `[uses as bait to capture]`.

3.5 HIGGINS HC ENGINE

The IE engine of HIGGINS generates *hypotheses* $h = \langle c, e1, e2, r \rangle$. These hypotheses are transformed into HITs by the HC engine of HIGGINS. When the players fill in values for the unbound arguments, the hypotheses turn into fully bound knowledge quads. Thus, a set of HITs H is eventually turned into a set of knowledge quads K. HITs come in two flavors. If instance $h \in H$ contains unbound arguments, the HIT is termed as an *acquisition task*. The player's objective is to come up with the correct binding(s). The IE engine may additionally provide candidate values for unbound arguments, in which case, the player sees a *multiple-choice question* for the correct binding. If the instance $h \in H$ contains only bound arguments, the HIT is called a *validation task*. Here the player's objective is to confirm or refute the hypothesis.

3.5.1 HIGGINS HC tasks

If the IE engine provides candidate values for unbound arguments, the worker/-player at the HC stage chooses the correct binding. If all the instances $i \in I$ contain bound arguments, the HIT is called a *validation* task. Here the player's objective is to confirm or reject the instance.

Acquisition Tasks. An acquisition task transforms a partially bound quad instance generated by the IE engine into a fully bound instance by completion of all bindings via inputs from the human players. The players choose from the provided list of candidates or type in their response.

In Table 3.1, we enumerate the different types of HITs depending on the unbound arguments, and show an example each.

HITs with unbound slots	Example HIT
$\langle \mathbf{c}, \mathbf{e1}, \mathbf{e2}, r \rangle$	In the movie *The Godfather*, what happened between *Michael Corleone* and *Sonny Corleone*?
$\langle \mathbf{e1}, \mathbf{e2}, \mathbf{r}, c \rangle$	In which movie did *Lord Aragorn* marry *Elf-Queen Arwen*?
$\langle \mathbf{c}, \mathbf{e1}, \mathbf{r}, e2 \rangle$ $\langle \mathbf{c}, \mathbf{e2}, \mathbf{r}, e1 \rangle$	In the *1992 presidential elections*, whom did *Bill Clinton defeat*?
$\langle \mathbf{c}, \mathbf{r}, e1, e2 \rangle$	In the JK Rowling's book *Harry Potter and the Half-Blood Prince*, who *killed* whom?
$\langle \mathbf{c}, \mathbf{e1}, r, e2 \rangle$ $\langle \mathbf{c}, \mathbf{e2}, r, e1 \rangle$	In the movie *Rambo: First Blood*, what is true about *Colonel Sam Trautman*?
$\langle \mathbf{r}, \mathbf{e1}, c, e2 \rangle$ $\langle \mathbf{r}, \mathbf{e2}, c, e1 \rangle$	Whom did *Casanova fall in love* with?
$\langle \mathbf{c}, e1, e2, r \rangle$	What of the following incidents happened in Shakespeare's play *Macbeth*?
$\langle \mathbf{r}, c, e1, e2 \rangle$	Which biographies involve *assassinations*?
$\langle \mathbf{e1}, c, e2, r \rangle$ $\langle \mathbf{e2}, c, e1, r \rangle$	Which of the following are true about *Nicolas Sarkozy*?

TABLE 3.1: Acquisition Tasks

Validation Tasks. A validation task confirms or rejects a fully bound quad instance generated by the IE engine via inputs from the human players. Table 3.2 shows an instance of the validation task.

HITs with unbound slots	Example HIT
$\langle \mathbf{c}, \mathbf{e1}, \mathbf{e2}, \mathbf{r} \rangle$	In the movie *The Batman Begins*, *Joker killed Batman*. Confirm/Reject.

TABLE 3.2: Validation Tasks

3.5.2 Question Generation

The key issue for the HC engine is to generate the questions for a game (or crowdsourced HITs) and, if it is a multiple-choice question, a set of candidate answers. For simpler explanation, here we focus on the case of hypotheses where only the relation argument is unbound.

In principle, we could randomly pick instances from the pool H of hypotheses generated by the IE engine. However, this would inevitably yield poor results, because many instances would not be suited for a specific user. Examples are entities that are unfamiliar to the user and/or obscure entities involving peripheral characters in a movie or book.

To avoid this pitfall, we tailor the questions to i) relate to context entities that match the user's interests (e.g., movies she knows well), and ii) refer to relations between salient entities (e.g., main characters of a movie) of popular movies. For the latter, we exploit knowledge sources like Wikipedia and IMDB for identifying popular movies and their important entities.

Once a hypothesis $h \in H$ is chosen as a question for a particular user, it merely needs to be cast into the surface form that the game uses, e.g., by adding cues from text about the context entity and the sentence that led to h and its surrounding paragraph. Figure 3.2 contains a screen-shot example. Analogously, for crowdsourced HITs we group the questions so as to allow human contributors to choose movies of their interest.

3.5.3 Generating Candidate Answers

In a game, HIGGINS could simply ask the user to fill in the missing value for a relationship, by providing a free-text form field. However, experience shows that it is easier to engage a broad set of users with multiple-choice questions and candidate answers to choose from. Therefore, we generate a small number of candidate answers (usually 5) and additionally offer a free-text field for entering other relationships. For the game effectiveness, it is important that the offered candidate answers i) are reasonable, that is, exclude nonsensical relations for the given context entity (e.g., exclude "*is the grandmother of*" for a question about James Bond and Le Chiffre), ii) include a good answer, if known from the original sentence that generated the HIT, and iii) are sufficiently diverse so that users see actual choices, as opposed to proposing only candidates that are so close that only a very sophisticated user could distinguish them (e.g., "*orders to kill*" vs. "*hired goons to eliminate*" - which would be considered the same by most players).

To satisfy these desiderata, the possible answers for a HIT are first ranked, based on the statistical language models, and then diversified. The details are presented in Chapter 4.

- **Ranking:** For this step, we start with the relational phrase from the original sentence that generated the HIT question. We use the statistical language model for phrase relatedness to enumerate semantically similar phrases in descending order of relatedness scores. For example, when starting with "*falls in love with*", highly related phrases would be "*loves*", "*love affair with*", "*has romantic passion for*", "*engages in romance with*", etc.
- **Diversification:** For diversification, we aim to remove near-duplicates from the top-k candidates according to the relatedness ranking. To this end, we remove all phrases that have significant overlap with a phrase that is ranked higher. Overlap in words is considered significant if there is a noun or verb contained in both phrases (excluding light verbs such as has, was etc.). In the above example, the phrases "*loves*" and "*love affair with*" are viewed as near-duplicates. Same is the case for "*has romantic passion for*", and "*engages in romance with*". In addition to this syntactic form of diversification, we use randomization by picking the final phrases for the multiple-choice question from a larger pool of top-ranked and de-duplicated candidates (typically picking 5 out of a pool of 20).

3.5.4 Rewards for HC

HIGGINS uses predefined question templates and plugs in the context, the entities, and the relation(s), to present a HIT to a game player or a crowdsourced worker.

In the game form, points are awarded to the player upon providing a response. The player can choose to either respond or skip a question that is presented. Notably, the game can be configured to give additional cues (from the context) upon request by the player. This is strategic: when the presented choices do not satisfy the player, the player can see the sentence for which the entity-relation fact occurs. In this way, noisy relations (that may occur for instance due to errors in pronoun resolution) can be corrected by players. To do this, players simply engage human intelligence using which correct resolution is much easier. In this case, the player can obtain only half the points for an answer. And, we enhance our KB quality using the intelligence of humans who may not be experts in a specific domain. When players can know/guess the answer without the cue, they gain full bonus points, as in this case we additionally tap into human knowledge.

For crowdsourced HITs, each response secures monetary rewards to the worker. The workers are also provided sentences of the source text, from which the IE engine generated the fact candidates. When none of the candidate answers hold, the worker types-in the correct relation/entity based on the text provided.

3.6 SUMMARY

The HIGGINS architecture aims at the synergistic benefits of combining automatic fact extraction methods with human computing systems. Its IE engine combines phrase statistics mined from large corpora with semantic resources to discover relations that manifest from complex and diverse phrasal expressions. Designed for achieving high recall, the IE engine may permit incorrect fact candidates or noise. Statistical language model based ranking of the resulting fact candidates (specifically its unbound slots) is performed to present the top-ranked ones at the HC stage. This helps eliminate noisy, incorrect and irrelevant candidates and as a result reduces the overall cost of human judgements.

4 CHAPTER

Candidate Fact Extraction and Ranking

4.1 Overview

In this chapter we present details of the IE engine in the HIGGINS framework. Specifically we describe the extraction methods employed to extract entities and their relational phrases from text. We describe the ranking models for relational phrases which help prune the fallacious quads, enabling effective crowdsourcing by producing only a handful of choices for the humans. Figure 4.1 pictorially summarizes the workflow of the HIGGINS IE engine.

4.2 Building The Entity Lexicon For HITs: Extraction, Resolution and Alias detection

HIGGINS compiles a dictionary of canonical entity names, their mentions and descriptive noun-phrases for its purposes. For canonical entity names, we tap into the character lists (and tables) in movie or book articles. Fig. 4.2 shows an example screen-shot from Wikipedia. We harvest names, aliases and nicknames of the characters from the descriptions in these lists using a regular expression based wrapper (see Appendix A). Examples of knowledge triples (all in the context of the Godfather movies) extracted in the above manner are: $\langle$ *Vito_Corleone knownAs "Don Corleone"* $\rangle$, $\langle$ *Santino_Corleone knownAs "Sonny"* $\rangle$, $\langle$ *Kay_Adams knownAs "Michael's non-Italian girlfriend"* $\rangle$, $\langle$ *Tom_Hagen knownAs "the Consigliere"* $\rangle$, $\langle$ *Virgil_Sollozzo knownAs "the Turk"* $\rangle$.

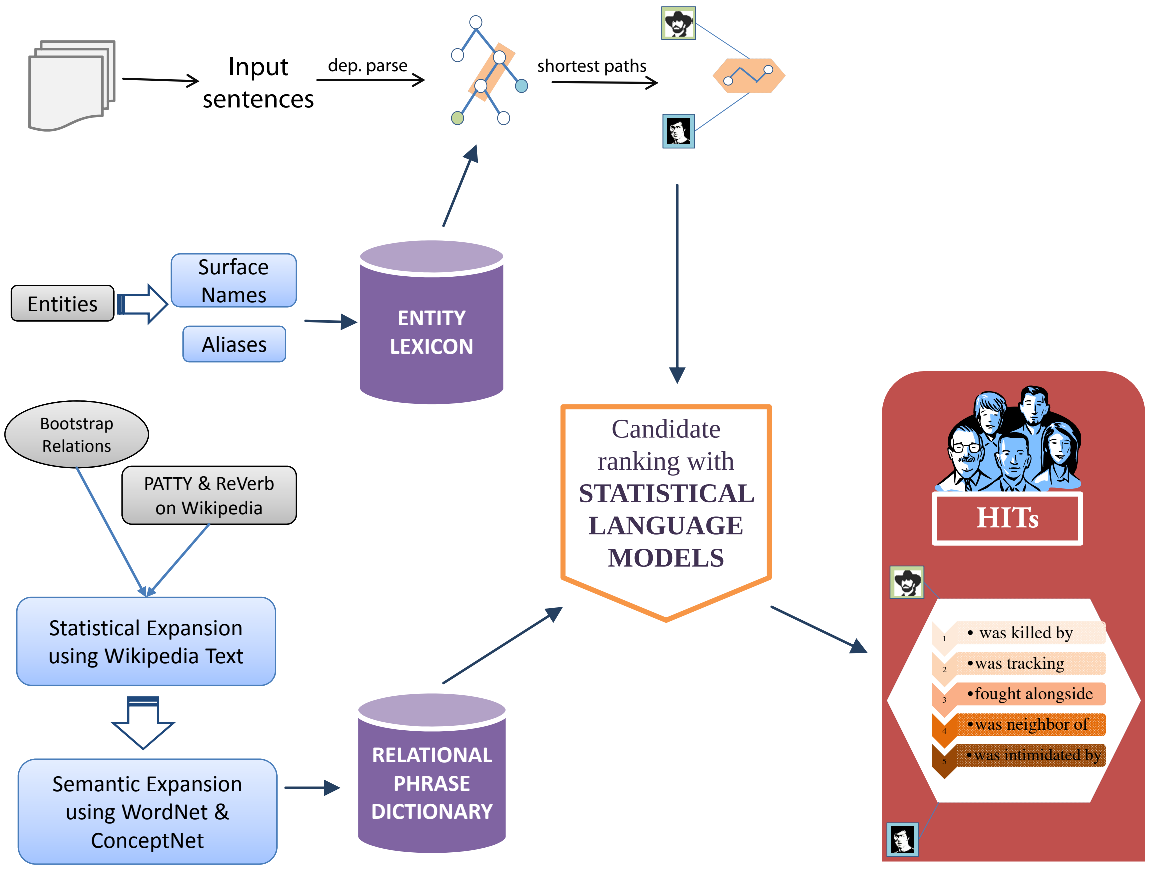
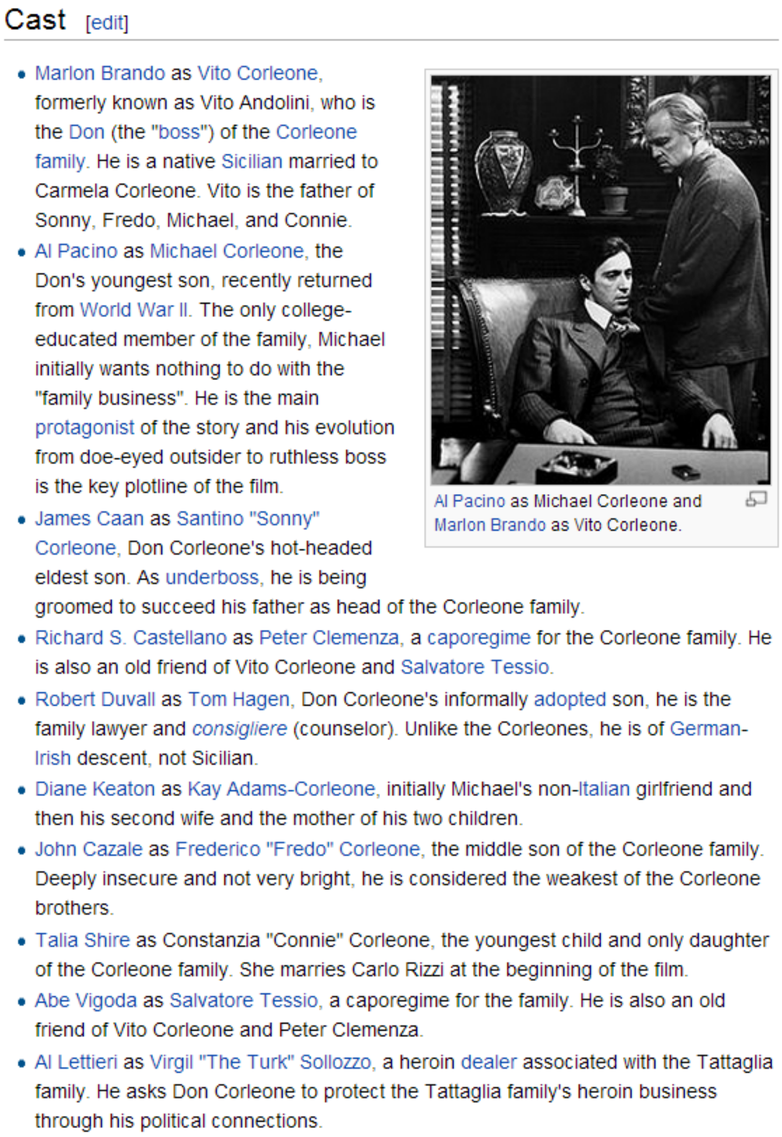

Input sentences
dep. parse
shortest paths
Entities
Surface Names
Aliases
ENTITY LEXICON
Bootstrap Relations
PATTY & ReVerb on Wikipedia
Statistical Expansion using Wikipedia Text
Semantic Expansion using WordNet & ConceptNet
RELATIONAL PHRASE DICTIONARY
Candidate ranking with STATISTICAL LANGUAGE MODELS
HITs
• was killed by
• was tracking
• fought alongside
• was neighbor of
• was intimidated by

FIGURE 4.1: Generating HITs with HIGGINS

Often these sections contain brief character potrayal snippets that provide descriptive noun phrases for the characters (for example, in Fig. 4.2, the phrases "*informally adopted son*" or "*family lawyer*" in the description of Tom Hagen). For snippet of each character, we extract these phrases using regular expressions on part-of-speech tags, listed in Appendix A. The common nouns in the extracted phrases are identified by the 'NN' tag in their parts-of-speech. By looking up the common nouns contained in the noun phrases of each character ("*son*", "*lawyer*", etc.) in the type system of YAGO ontology (for example, in the subtypes of the type PERSON) we identify phrases that describe these characters. This way we harvest fine-grained information about the entities such as ⟨ *Vito_ Corleone, isa, boss* ⟩, ⟨ *Tom_Hagen, isa, lawyer* ⟩ and so on. Using this information and by generating term combinations of



Cast [edit]

- Marlon Brando as Vito Corleone, formerly known as Vito Andolini, who is the Don (the "boss") of the Corleone family. He is a native Sicilian married to Carmela Corleone. Vito is the father of Sonny, Fredo, Michael, and Connie.
- Al Pacino as Michael Corleone, the Don's youngest son, recently returned from World War II. The only college-educated member of the family, Michael initially wants nothing to do with the "family business". He is the main protagonist of the story and his evolution from doe-eyed outsider to ruthless boss is the key plotline of the film.
- James Caan as Santino "Sonny" Corleone, Don Corleone's hot-headed eldest son. As underboss, he is being groomed to succeed his father as head of the Corleone family.
- Richard S. Castellano as Peter Clemenza, a caporegime for the Corleone family. He is also an old friend of Vito Corleone and Salvatore Tessio.
- Robert Duvall as Tom Hagen, Don Corleone's informally adopted son, he is the family lawyer and *consigliere* (counselor). Unlike the Corleones, he is of German-Irish descent, not Sicilian.
- Diane Keaton as Kay Adams-Corleone, initially Michael's non-Italian girlfriend and then his second wife and the mother of his two children.
- John Cazale as Frederico "Fredo" Corleone, the middle son of the Corleone family. Deeply insecure and not very bright, he is considered the weakest of the Corleone brothers.
- Talia Shire as Constanzia "Connie" Corleone, the youngest child and only daughter of the Corleone family. She marries Carlo Rizzi at the beginning of the film.
- Abe Vigoda as Salvatore Tessio, a caporegime for the family. He is also an old friend of Vito Corleone and Peter Clemenza.
- Al Lettieri as Virgil "The Turk" Sollozzo, a heroin dealer associated with the Tattaglia family. He asks Don Corleone to protect the Tattaglia family's heroin business through his political connections.

Al Pacino as Michael Corleone and Marlon Brando as Vito Corleone.

FIGURE 4.2: Cast section in the 'The Godfather' movie article in Wikipedia (figure shows a truncated snapshot)

full names using heuristics, we identify entity mentions in the storyline text.

Finally we perform coreference resolution using the Stanford Coreference Resolution Tool [86] to recognize anaphoras. The accuracy of the tool can be enhanced by using gender information of the entities. We developed a heuristic method for resolving personal pronouns in such texts for this purpose. We focus on pronouns in singular form, and gather gender-resolving pronouns for characters. We map a male (female) pronoun to the closest preceding male (female) name occurring as grammatical subject. The gender properties for entities are derived from character descriptions using a classifier (over features such as frequencies of male vs. female pronouns, prior information on male vs. female first names, etc.).

4.3 Relations & Relational Phrases

The relationships between humans or fictional characters seen in narrative text are diverse and multifarious. While some of these relations are common place (for e.g., "*sister of*", "*criticized*", or "*murdered*"), many are less frequent but equally interesting (for example, "*tries to overthrow*", "*was the inspiration for*" or "*nearly lost to*"). Moreover authors of narrative text use a variety of phrases to express such relations (for example, the relation "*criticized*" may be expressed using phrases such as "*knocked*" or "*picked apart*"). For achieving a decent recall over such relations, it is essential that the IE engine has knowledge of phrases that express them. We employ a two-staged approach for the collection of relations and relationship phrases. We first establish a dictionary of relations utilizing existing IE resources. We then collect phrases that express these relations using co-occurrence statistics on a large corpus. To increase coverage, these phrases are expanded further by using semantic resources such as WordNet and ConceptNet.

4.3.1 Building the Dictionary of Relations

As surface expressions for binary relations we construct a dictionary of relations using noun and verb-form lexico-syntactic patterns. They come from the following sources:

1. **Extractions of OpenIE systems.** We include relational instances from high quality extractions of ReVerb [17, 89] and PATTY [66, 67].

   - ReVerb is an OpenIE system that identifies and extracts verb-mediated binary relations from Web-scale textual collections. It captures relational phrases that adhere to a syntactic constraint enforced using regular expressions over part-of-speech (POS) tags, to reduce incoherent and uninformative phrases. To remove highly specific phrases from the resulting set, filtering based on a lexical constraint is applied. This is done by retaining phrases if their POS pattern is observed with at least one distinct argument pair in ClueWeb09 corpus. Unlike other OpenIE systems, ReVerb achieves higher coverage by allowing phrases with light-verb construction (such as "*had a deal with*") and phrases with multiple verbs ("*refuses to negotiate with*"). Such characteristic phrases often occur between character roles in narrative text. We make use of ReVerb extractions on the Clueweb09[1] dataset, made available by the authors (http://reverb.cs.washington.edu/).

- PATTY extracts semantically typed binary relations from Web-scale corpora and organizes them into a subsumption taxonomy. It transforms relational phrases found using dependency parse trees into syntactic-lexical-ontological (SOL) patterns, where the arguments have ontological type signatures. With the arguments forming support sets, the SOL patterns are organized into subsumption hierarchy by determining inclusion, mutual inclusion and independence among the support sets. Unlike the ReVerb system, PATTY makes use of arbitrary patterns which include noun-mediated relational phrases. Therefore we make use of PATTY extractions on the English Wikipedia corpus, made available by the authors (http://www.mpi-inf.mpg.de/yago-naga/patty/).

ReVerb performs general extraction with no type restrictions on the arguments while PATTY uses ontological types from YAGO. Not all of these argument types are useful in the context of relations that occur between character roles in movies or books. We restrict the OpenIE extractions to the arguments with the Freebase and YAGO types (and their sub-types) shown in Table 4.1. From 14.7 million ReVerb extractions on Clueweb09, this step produced 192,291 surface phrases for relations. Similarly 199,855 phrases from PATTY were produced.

2. **Manual Compilation of Relations.** In order to ensure the presence of salient relations in our dictionary, we manually produced a set of 544 relations. Appendix B lists these relations.

[1] http://lemurproject.org/clueweb09.php/

Freebase Types	YAGO types
/people/person	person
/people/appointee	living_thing
/people/appointed_role	imaginary_being
/people/appointed_nomination	fictional
/people/deceased_person	film_characters
/people/family_name	social_group
/people/measured_person	
/tv/tv_character	
/film/film_character	
/book/book_character	
/theater/theater_character	
/fictional_universe/fictional_character	
/celebrities/celebrity	

TABLE 4.1: Freebase & YAGO types for narrative characters

In the spirit of [17], we perform additional checks based on heuristic to prune out noisy and incorrect OpenIE phrases. They are:

i) presence of at least one common noun or a verb

ii) multi-word phrases that do not end with a verb must end with a preposition.

All the phrases are then lemmatized and POS lifted to generate lexico-syntactic forms. The duplicates are then merged, providing us a bootstrap set of **116,471** relations. All corresponding surface forms of the relational phrases are retained to be used in HIT generation at the HC stage.

4.4 PHRASE DISCOVERY FOR RELATIONS

The dictionary of relations provides us a fairly large set of potential relations, using which the HIGGINS system can automatically discover participating entities in text – thus procuring fact candidates. However, the number of candidate instances wherein the exact lexical forms of the relations are observed in the document text can be very limited. To overcome this, we generalize the relational phrases by allowing wild-cards in their matching criteria. As in PATTY, words within the phrases that do not alter their semantic sense such as pronouns, determiners, adjectives and modifiers can be relaxed to within any member of their class (for example, the pattern "*be* ⟨det⟩ *inspiration for*" in place of "*was the inspiration for*", or, the pattern "*files* ⟨prp⟩ *report about*" in place of "*files his report about*").

However given the diversity of expression in textual sources such as narratives, pattern generalization over lexical forms does not suffice. Many of the relations are expressed in semantically similar phrases are unrelated in their lexical forms. For example, the relation "ADMIRED" may be expressed using phrases such as "*looked up to*" or "*repeatedly praised*", and loosely as "*eulogized*", "*idolized*", or "*was an austere devotee of*". Similarly the noun-form relation "ACCOMPLICE" can be expressed using "*confederate*", "*aid of*" or in the verb-form as "*conspired along with*" or "*provided support to*". If such associated phrases for the relations are not taken into account, overall recall of the system would suffer.

To collect phrases for relations in the dictionary, we mine patterns using co-occurring entity pairs in Wikipedia article text. This process is detailed below.

4.4.1 Statistical Mining of Relational Phrases

Our mining procedure hinges on the idea that a phrase p is associated with relation r, if many entity pairs in a large corpus are observed with both r and p. For example the phrase "*entered a wedlock*" is associated with the relation "MARRIEDTO" if many entity pairs (e.g., (Kate, William), (Jigme, Jetsun), (Victoria, Daniel) and so on) co-occur with both these phrases in the corpus documents. Our procedure allows for general noun phrases as arguments, to capture a larger proportion of phrase expressions (for e.g., *Blair's son .. ties the knot with .. his long-term girlfriend*). Regex-based rules on parts-of-speech tags listed in A.2 of Appendix A are used for noun-phrase detection.

We use the full text of 4.1 million English Wikipedia articles, which are split into sentences, lemmatize and POS tag using Stanford CoreNLP processing tool [90].

4.4.1.1 Mining Procedure.

The dictionary of relations, D, compiled in 4.3.1 is used to bootstrap the mining process. The mining procedure has two steps:

i. *Finding noun-phrase arguments:* For each relation $r \in D$, the bootstrap set, find contexts C (we use sentence-level text as contexts) in the corpus where r occurs. For each context in C, extract the set of noun-phrase pairs E that surround r.

ii. *Extracting phrases:* For each noun-phrase pair in $e \in E$, find contexts C' where e occurs and extract encompassing phrases P.

From the above two steps we collect relational phrases, their co-occurring noun-phrase pairs, and co-occurrence statistics. We also collect overall corpus frequency counts of the noun-phrase pairs and the mined patterns.

4.4.1.2 Statistical Relatedness of Phrases.

We use the co-occurrence statistics of noun-phrase pairs with a dictionary relation and a mined relational phrase to quantify the strength of association between the relation and the phrase. For a relation r and a phrase p, stronger semantic closeness between p and r is indicated by the presence of a large number of different noun-phrase pairs in the corpus that co-occur with both r and p. We quantify this degree of *closeness* by assigning relatedness scores to each (p, r) pair. We studied several existing statistical relatedness measures for this purpose.

Statistical measures based on word-based collocation in large corpora have extensively been studied in the field of natural language processing (see Chapter 5 in [91] for an overview). We experimented with several different measures by adapting them to our setting – Pointwise Mutual Information (PMI), normalized PMI, Mutual Information, Pearson's chi-square test and cosine of PMI. For a dictionary relation r, a mined relational phrase p, and the set of co-occurring noun-phrase pairs N, Table 4.2 lists these measures.

Measure	Evaluation
PMI	$PMI(p,r) = \log \frac{\Pr(p,r)}{\Pr(p)*\Pr(r)}$
normalized PMI	$nPMI(p,r) = \frac{\log \Pr(p,r)/\Pr(p)*\Pr(r)}{-log \Pr(p,r)}$
Mutual Information	$MI(\mathrm{p},\mathrm{r}) = \sum_{p\in\mathrm{p}} \sum_{r\in\mathrm{r}} \Pr(p,r) * \log \frac{\Pr(p,r)}{\Pr(p)*\Pr(r)}$
Pearson's chi-square test	$X^2 = \sum_{p,r} \frac{(\mathbf{O}_{p,r}-\mathbf{E}_{p,r})^2}{\mathbf{E}_{p,r}}$
cosine of PMI	$cPMI(p,r) = \frac{\sum_{n\in N} PMI(p,n)*PMI(r,n)}{\sqrt{\sum_{n\in N} PMI(p,n)^2}*\sqrt{\sum_{n\in N} PMI(r,n)^2}}$

TABLE 4.2: Collocation measures for phrase 'p' and dictionary relation 'r'

We used these collocation measures to establish strength of dependence between p and r based on the sets of noun-phrase pairs that they co-occur with (see Appendix B for details on estimation of probabilities in each measure). In practice however, given the sparseness of phrases in the underlying corpus, we observed in our experimentation that these measures perform poorly. Also we observed that phrase mining on the much larger ClueWeb09 corpus produced a higher degree of noisy phrases without alleviating the phrase sparseness issue. Therefore, for relatedness scores, we adopted the Jaccard index, a simpler similarity measure which showed robust results.

Jaccard Index for Statistical Phrase Relatedness.

The Jaccard index is a statistical degree of similarity, defined as the ratio of the intersection and union of two sets. In our setting, the noun-phrase pairs co-occurring with two relational phrases form the sets. However we take into consideration the occurrence frequencies with each phrase, and treat occurrences of noun-phrase pairs as bags (rather than sets). For two relational phrases p and r and their respective noun-phrase pair bags $[\![N(p)]\!]$ and $[\![N(r)]\!]$, Jaccard's index is given as,

$$J(p,r) = \frac{|[\![N(p)]\!] \cap [\![N(r)]\!]|}{|[\![N(p)]\!] \cup [\![N(r)]\!]|} \tag{4.1}$$

This is equivalent to a weighted version of Jaccard index, which for a noun-phrase pair $n \in [\![N(p)]\!] \cap [\![N(r)]\!]$ is evaluated as,

$$J(p,r) = \frac{\sum_n (min(\mathbf{count}[\![N(p)]\!]n, \mathbf{count}[\![N(r)]\!]n))}{\sum_n (max(\mathbf{count}[\![N(p)]\!]n, \mathbf{count}[\![N(r)]\!]n))} \tag{4.2}$$

where $\mathbf{count}[\![N(p)]\!]n$ and $\mathbf{count}[\![N(p)]\!]n$ denote frequencies of the pair n in the bags $[\![N(p)]\!]$ and $[\![N(r)]\!]$ respectively.

The principal benefits of using statistical phrase mining is discovery of relational phrases that do not strictly have syntactic or lexical similarity with the initial set of relations. Based on co-occurring noun-phrase pairs, relational phrases such as "*dethroned*" or "*successfully organized a coup against*" or "*end the reign of*" for the relation "*successor of*" can be mined while such information cannot be directly obtained by exploiting thesauri-like linguistic resources. Sophisticated entailment phrases that result from corpus idiosyncrasies such as "*wanted to destroy*" and "defeated" for the relation "*confronted*" are also produced as a result of the mining process.

Majority of the relations in our dictionary yielded phrases that are infrequent in the corpus. We observed that discarding infrequent phrases based on thresholding resulted in decrease of recall. These infrequent but highly specific relational phrases such as "*wrote songs in praise of*", or "*disregarded the advice of*" often form crucial evidence to the relations in our dictionary. So we retain all mined phrases regardless of their co-occurrence frequencies.

4.4.2 Scalable Mining

To achieve decent coverage over both the salient as well as the highly specific relational phrases, the mining procedure described in 4.4.1.1 needs to be run over a large text corpus. The procedure involves book keeping of relations, noun-phrase pairs, relational phrases that occur with each noun-phrase pair and their corresponding co-occurrence frequencies in each step. A centralized implementation hence would be memory-intensive, requiring to keep millions of phrases in main memory for efficient statistics aggregation. Alternatively a Map-Reduce style implementation is a natural fit wherein the statistics aggregation can be performed in the reduce phase. Therefore we developed the mining procedure as a sequence of three Map-Reduce jobs: DICTRELATIONSEARCH, NOUNPHRASEPAIRSEARCH and RELATIONALPHRASESEARCH.

- **DictRelationSearch.**

  In DictRelationSearch, we start with the dictionary relations, look for their occurrences in the corpus and capture co-occurring noun-phrase pairs. The mapper takes dictionary relations as keys, performs corpus search and constructs composite keys consisting of noun phrase arguments and the relation. The composite keys are grouped by the noun-phrase pair prior to the reduce phase, making the aggregation efficient. The reducer produces aggregate occurrence counts of relations and their co-occurrence counts with the noun-phrase pairs.

- **NounPhrasePairSearch.**

  In NounPhrasePairSearch, we search for occurrences of noun-phrase pairs captured in the previous job and co-occurring relational phrases. The noun-phrase pairs form the keys to the mapper which triggers a corpus search for each pair and extracts their encompassing relational phrases. The mapper also performs a phrase filtering step to constrain phrases to the same form as our dictionary relations. The filtering step therefore weeds out phrases that do not contain at least one common noun or verb and multi-word phrases that do not end with a verb or preposition. As in the previous job, composite keys are constructed using the noun-phrase pairs and the co-occurring relational phrases. The output of the mapper are grouped by the relational phrases before aggregation at the reduce stage. The reducer produces aggregate occurrence counts of noun-phrase pairs and their co-occurrence counts with the relational phrases.

- **RelationalPhraseSearch.**

  Finally, in the RelationalPhraseSearch job, we capture global co-occurrence frequencies of the relational phrases with all noun-phrase pairs of the corpus. The relational phrases form the keys to the mapper which performs a corpus search and extracts co-occurring noun-phrase pairs. The reducer produces aggregate occurrence counts of the phrases and their co-occurrence counts with the noun-phrase pairs.

We use the Hadoop implementation for the above Map-Reduce jobs and collect frequency statistics by maintaining global counters in each job. To make the entire mining process faster we collect all $\langle noun-phrase\rangle\langle * \rangle\langle noun-phrase\rangle$ triples in the corpus and materialize them beforehand, thus avoiding corpus search at each step.

4.4.3 Semantic Expansion of Relational Phrases

Statistical phrase mining yielded us a large number of relational phrases that have varying degrees of co-occurrence similarity with our dictionary of relations. For the 116,471 relations in the dictionary, we obtained 985,069 phrases from Wikipedia articles. Nevertheless the output quality and coverage is highly dependent on the richness of the text corpus, and it is difficult to assess if most phrases that express a certain relation have been covered. Therefore, to complement the two phrases obtained via statistical mining, we exploit manually crafted relations from two semantic resources: WordNet and ConceptNet.

4.4.3.1 WordNet Relations

WordNet [43, 84] is a large lexicon of nouns, verbs, adjectives and adverbs which are grouped based on semantic similarity. Each such group, called a synset, represents a distinct concept and different synsets are interlinked by means of conceptual-semantic and lexical relations. Two synsets can be interlinked by an *isA* relationship, which includes hyponyms (specialization, as in '*half sister*' hyponym of '*sister*') and hypernyms (generalization, as in '*female sibling*' hypernym of '*sister*'). Such WordNet relationships are directly added into out list of relational phrases in case they have not been obtained from corpus mining. Also the mined phrases can be reconciled by substituting the participating common nouns and verbs with WordNet hyponyms and hypernyms. For example, a phrase "*stabbed with a knife*" can be expanded into other meaningful phrases "*stabbed with a hunting knife*" or "*stabbed with a sharp metal blade*" using hyponyms and hypernyms respectively.

4.4.3.2 ConceptNet Relations

ConceptNet [82, 92] is a semantic network of common-sense facts collected from human contributors, to make available "unstated" knowledge to NLP and computer vision applications. The ConceptNet graph contains concepts as nodes (which could be nouns or short phrases) and relationships as edges, for example "*uncle* → isA → *male*", "*husband* → relatedTo → *married man*" or "*hanging* → isA → *killing*". From the ConceptNet graph we incorporate into our relational phrases the concepts that take part in the following relationships: IsA, SimilarTo, DerivedFrom, RelatedTo and ConceptuallyRelatedTo. These concepts from ConceptNet supplement worldly knowledge to our relational phrases such as "*award* → isa → *recognition*", "*account* → relatedTo → *credit card*" or "*record producer* → conceptuallyRelatedTo → *music group*". As in the case with WordNet relations, these phrases can be directly added to our list when absent, or reconciled by substituting the participating nouns or verbs in the statistically mined phrases.

By combining 985,000 phrases obtained from statistical mining with 1,668,042 phrases from WordNet and 205,432 phrases from ConceptNet, we obtain a total of 2,858,474 unique relational phrases. Given a textual sentence from any narrative text, these phrases form the basis of identifying if one of the relations from our dictionary hold between any entity pair of that sentence. To narrow down on the relevant phrases we employ phrase pruning using dependency parse of the sentence. Thereafter, taking the resultant phrases into account we rank the relations in our dictionary and provide them as candidate answers at the HC stage. We use statistical language models for ranking relations taking the phrases into account. Specifically, we adopt the statistical translation model [93] for this task. The details on language model based ranking are provided in the section 4.5.

4.4.4 Pruning Phrase Candidates.

Given a snippet of narrative text such as movie/book plot story, any sentence-level context that contain at least a pair of entity mentions is of interest for extracting relations. However these sentences from narratives often contain multiple entity mentions and express relations in a convoluted way. Therefore, to eliminate false positives, we employ dependency parsing on all relevant sentences. The nodes (words) on the shortest path in the dependency parse tree between a pair of entity mentions give a cue on the relation between them [94]. As these nodes do not capture semantic relations in their entirety we augment them with adjective/adverb modifiers and negations, and also take conjunctions into account. Specifically we allow all nodes in the parse tree that have grammatical relations in {agent, aux, cc, ccomp, complm, det, infmod, mark, neg, nn, npadvmod, poss, possessive, prt, punct, rcmod}. Also {prep} are allowed if they are descendents of the subject entity in the pair. In addition, we restrict ourselves to the mention pair being grammatical subjects or objects of the clauses in the sentence.

We use the Stanford Dependency Parser [87, 88] and identify all dependency paths in a sentence connecting a pair of entities that occurs in a candidate from the previous phases. If no such path exists, the candidate is discarded. Otherwise, we retain all candidates whose relational phrases overlap with words on at least one of the dependency paths. For example, in the sentence "*Shortly after the meeting between Vincent and Lucchesi, Altobello travels to the small village of Montelepre, where he hires Mosca (Mario Donatone), a veteran hitman, to assassinate Michael.*", to detect the relation between Altobellow and Mosca, we keep the top-k relations that match the dependency path `[travels to the small village of Montelepre hires to assassinate]`.

4.5 Ranking Relation Candidates

In order to generate candidate answers for HITs, we determine the relevance of the relational phrases to a query context. For this purpose, given a query sentence, we use a statistical language model to rank the relations in our dictionary. Ranked in the descending order of relevance, these relations are presented to the human contributor at the HC stage.

4.5.1 Background: Statistical Language Models

Statistical Language Models [95] (LM), in the field of information retrieval (IR), form a principled approach to address the central problem of query-based ranked retrieval on documents. LMs, therefore, have been used for ranking in different contexts: ranking documents for keyword queries, ranking passages as a first stage in question answering, ranking entities for semantic search, ranking sentences for summarization, etc. Albeit based on sound probabilistic principles, LMs have rarely been used for problems other than IR tasks. Our approach here is a novel way of harnessing statistical LMs for knowledge acquisition.

4.5.2 Language Model for Ranking Relations

A language model is a probability distribution over words or bigrams or variable-length sequences, often using a multinomial model. Each document, each query, each passage, etc. is associated with an LM, where the parameters are estimated from the document, query, or passage, respectively, and are additionally smoothed by statistics over an entire corpus or query log or other background information. The principle for ranking is that the best document for a given query is the one whose LM has the highest likelihood of generating the query (and analogously for other IR tasks). This is typically cast into computing conditional probabilities $\Pr(q|d)$ or information-theoretic scores like the Kullback-Leibler divergence $KL(q|d)$ (aka. information gain or relative entropy).

In our problem setting, we are interested in which of the relations in our dictionary apply given a query sentence. For the sentence as a query we consider only the paths from the dependency tree between the entity pair as explained in 4.4.4.

Given an entity pair $E = \{e_1, e_2\}$ occurring in a sentence s, a set of candidate relations G_r we need to rank the relations in G_r according to their plausibility to relate e_1 and e_2. This probability $\Pr(r)$ is given by

$$\Pr(r) = \alpha * \Pr(r|s) + (1 - \alpha) * \Pr(r|c) \tag{4.3}$$

where s is a sentence and c is the corpus. $\Pr(r|c)$ is the Jelinek-Mercer based smoothing factor. Therefore, each dictionary relation holds a non-zero probability provided it occurs within the corpus. In our case, smoothing using a corpus of documents only partially alleviates the problem of data sparseness by taking into account the occurrences of the relation in the corpus. Due to variations in expressing relations in natural language, corpus-based smoothing of the relations in our model does not ensure that these variations obtain a non-zero probability. We overcome this problem by using our relational phrases and a specifically relevant form of LMs called statistical translation model.

4.5.3 Translation Model for Phrase Relatedness

Statistical translation models offer a principled approach to capture semantic translations of one word sequence to another. The basic idea is to estimate the likelihood of translating a word into another word in the vocabulary, or, the likelihood of translating the document to a query posed in a different language. They have been successfully employed for crosslingual information retrieval and question answering systems. Such settings typically consider probabilities of the form $\Pr(q|d) = \sum_{w \in q} \left(\sum_{v \in d} \Pr(w|v) \Pr(v|d) \right)$, where '$w$' is a query term (say, in French), and the document 'd' to be retrieved contains terms belonging to a different vocabulary (for example, in English). The $\Pr(w|v)$ terms constitute the translation model, allowing the French term 'w' to be translated to semantically related words $v \in d$.

In our setting, we use this principle in a generalized way for "translating" one relational phrase into a different paraphrase. For example, we are interested in the probability $\Pr($"*consigliere*"|"*adviser*"$)$. This is the basis for the phrase relatedness scores. By using a translation model for relation ranking, we allow the relation to be expressed in any of its variations during its likelihood estimation (the likelihood of one expression of a relation into another).

In Eq. 4.3, we use the translation model to estimate $\Pr(r|s)$, the probability of a relation r being applicable to an entity pair in the query sentence s. Intuitively, the idea of the translation model is to allow the relation to occur in any of its variations. If $G_r = \{r_1, r_2, .., r_n\}$ is the set of different relations in our dictionary that occur between the entities in the text and $P_i = \{p_{i1}, p_{i2}, .., p_{im}\}$ are m different phrases for the relation r_i, we use the translation model to obtain the probability $\Pr(r_i|s)$ as

$$\Pr(r_i|s) = \sum_{j=1}^{m} \Pr(r_i|p_{ij}) \Pr(p_{ij}|s) \tag{4.4}$$

In the above equation, to estimate $\Pr(p|s)$, we use the Dice coefficient over bigrams of the phrase p and the dependency path in sentence s. However, the key part of the translation model is the estimation of the relation translation probability $\Pr(r|p)$. For crosslingual query likelihood, Brown et al [96] estimated this word-word probability using the EM algorithm over parallel corpora of two languages. In mono-lingual cases, Berger et al [93] created synthetic word queries as training data, i.e., queries artificially created from the document to make it relevant. In our setting, we use the scores for the relational phrases using a mixture model, described in the section below.

4.5.4 Mixture Model for Combining Statistical and Semantic Components

The translation probabilities $\Pr(r|p)$ in our model encode the likelihood of a dictionary relation r to be translated to the relational phrases. Recall that the relational phrases are collected through co-occurrence based phrase mining from a text corpus and later expanded with phrases from WordNet and ConceptNet. Since we combine these statistics with semantic resources for phrases, we need to combine different kinds of relatedness scores as well.

If p_i are all the phrases of the dictionary relation r, the probability $\Pr(r|p)$ is estimated as

$$\Pr(r|p) = \frac{relscore(r,p)}{\sum_i relscore(r,p_i)} \tag{4.5}$$

where the relatedness score between r and p are obtained using a mixture model with a linear combination of scores, given as,

$$\begin{aligned} relscore(r,p) \quad = \quad & \alpha_{stats} \times score_{stats}(r,p) \\ & + \beta_{WordNet} \times score_{WordNet}(r,p) \\ & + \gamma_{ConceptNet} \times score_{ConceptNet}(r,p). \end{aligned}$$

Here, the scores for the statistics component are the Jaccard overlap scores discussed in section 4.4.1.2.

For resources like WordNet or ConceptNet, there are explicit, manually crafted relations that connect the phrases. The ISA relation from WordNet and the SIMILARTO relation from ConceptNet establish equivalence among participating phrases while DERIVEDFROM and CONCEPTUALLYRELATEDTO relations establish loose relationships among the phrases. Therefore these relations offer the degree of similarity between the phrases, which needs to be taken into account for scoring them. We extend the mixture model for fine-grained control over the phrases that arrive from different WordNet/ConceptNet relations and flexibility in their combination. Weighing is achieved by associating a coefficient with each type of WordNet/ConceptNet relation between phrases, that is set to a fixed value between 0 and 1. For example, when two phrases are in the DERIVEDFROM relation in ConceptNet, we assign a fixed relatedness score of $\gamma_{ConceptNet_{DerivedFrom}}$. All these scores are combined into the mixture-model score, first for WordNet and ConceptNet separately, and finally combined with the statistical Jaccard scores. So the final relatedness score between r and p becomes

$$\begin{aligned}
relscore(r,p) \quad = \quad & \alpha_{stats} \times score_{stats}(r,p) \\
& + \beta^{1}_{WordNet_{hypernym}} \times score_{WordNet}(r,p) \\
& + \beta^{2}_{WordNet_{hyponym}} \times score_{WordNet}(r,p) \\
& + \gamma^{1}_{ConceptNet_{IsA}} \times score_{ConceptNet}(r,p) \\
& + \gamma^{2}_{ConceptNet_{SimilarTo}} \times score_{ConceptNet}(r,p) \\
& + \gamma^{3}_{ConceptNet_{DerivedFrom}} \times score_{ConceptNet}(r,p) \\
& + \gamma^{4}_{ConceptNet_{RelatedTo}} \times score_{ConceptNet}(r,p) \\
& + \gamma^{5}_{ConceptNet_{ConceptuallyRelatedTo}} \times score_{ConceptNet}(r,p).
\end{aligned}$$

4.6 SUMMARY

The HIGGINS IE engine makes use of entity names, entity descriptive phrases, a lexicon of relations and a large number of relational phrases. The generation of candidate answers for multiple-choice questions is based on the relations from our dictionary, which is done by ranking and diversifying them given a specific question (HIT). Recall that the dictionary is built from a variety of sources, including semantic resources like WordNet or ConceptNet and statistics collected by Web mining. For each pair of relational phrases, we compute a *relatedness score* using a mixture model that combines multiple kinds of statistical language models (LM's).

5 CHAPTER

Generating Crowdsourced Human Intelligence Tasks for Fact Acquisition

5.1 Overview

The previous chapter presented in detail the compilation of entity and relation lexicons, and the underlying ranking model that the HIGGINS IE engine employs to facilitate crowdsourcing tasks at the HC stage. This chapter discusses the construction of crowdsourced HITs in HIGGINS. We present experimental evidence of the superior performance of HIGGINS, in three parts:

A. First, we present experiments demonstrating the effectiveness of HIGGINS in generation of knowledge-base facts. The notion of effectiveness centers on

a) the *quality* of worker-evaluated fact triples compared to the ground-truth, as shown by precision and recall measurements,

b) the *costs* incurred, as measured by the agreement among the responses of the workers to the HITs

B. Second, we compare HIGGINS against pure automated IE approaches and pure HC based approaches to demonstrate its benefits.

C. Finally, we experimentally investigate the influence of different components of the HIGGINS IE engine and verify the right combination of its ingredients.

5.2 Generating HITs for Crowdsourced Fact Acquisition

We conducted experiments using plots and character descriptions of books stories from www.sparknotes.com/lit/ [79] and movies articles from Wikipedia. We compiled data for about 3,374 movies and 352 books, with a total of 52,700 characters. In total, we generated about 27,980 questions for the HC stage. We outline the question generation process for HITs below.

5.2.1 Building HITs

First, we extract character names, their alternative names, titles and descriptional phrases from the character descriptions of the movie/book articles, thus building an entity phrase dictionary. We use the text from movie or book story plots to extract relations between the character roles. To effectively identify characters in such text, we generate character name combinations from the entity phrase dictionary. These combinations are then resolved in the plot sentences and replaced with their full canonical names from the entity dictionary. For example all occurrences of "*Vito*", "*Vito Corleone*" or "*Don Corleone*" are replaced by "*Don Vito Corleone*" in the plot text. However, in case of conflicts in name combinations (for e.g., whether "*Corleone*" seen in the text "*... the Corleone family ...*" is "*Don Vito Corleone*" or "*Michael Corleone*"), we do not perform replacements. Combined with the extensive name phrases in our dictionary (such as "*Don Corleone*" nickname of "*The Godfather*", or, "*Sonny*" nickname of "*Santiano Corleone*"), we observed that this heuristics-based character name resolution works well in most cases.

Second, we split the text from the story plot section in the movie/book articles into sentences. To deal with the complexity of long sentences in the narrative plots, we used dependency parsing and extracted the shortest paths between occurrences of the movie/book characters. All words on the these paths used as query to the statistical language model described in Section 4.5 which rank the dictionary of relations in relevance to this query. For question generation we considered all sentences that mention the pair of character roles, obtain the ranked list generated by each sentence and finally aggregate the lists. The aggregation procedure is described below.

Obtaining aggregate candidates. If $S = \{s_1, s_2, ..s_n\}$ are the sentences that contain mentions of the character pair, we rank the relational phrases from the dictionary D using the mixture model (described in Chapter. 4) for each sentence s_i and obtain the ranked lists $R = \{L_1, L_2, .., L_n\}$ respectively.

Processing all the lists for each pair, the aggregate score for each relation $r \in D$ is obtained as $\sum\limits_{i=1}^{n} \frac{1}{pos_i(r)}$, where $pos_i(r)$ is the position of r in the ranked list L_i. The relations are ranked by the aggregate score and the top-5 are presented as choices to the player.

The top-5 options thus generated always include the most likely candidate phrases. However, because of errors in dependency parsing, pronoun resolution, and other steps, even the most likely phrase could be incorrect. For some sentences and choice of entity pairs, actually no relationship would hold. Therefore, the interface for HC contributors always included the options NORELATION and OTHER. In the latter case, the users could enter a relationship as free text. Furthermore, for some entity pairs, multiple relationships could hold and the top-5 candidates could include more than one of these.

Fig. 5.1 shows two example questions generated by the system and the aggregated answer candidates.

5.3 Experimental Setup

For evaluation, we consider two samples drawn from the pool of all questions: a) a **prominent** set of questions drawn from popular movies & books; b) a **random** set of questions drawn randomly from the entire pool, excluding the set of movies & books from the prominent sample.

In addition, to test how HIGGINS performs with varying complexity of text, we experimented with two different choices of input texts: i) plots and story summaries of movies and books (shown as MoviePlots & BookPlots respectively), containing convoluted sentences, often containing more than two character mentions in a sentence and ii) character descriptions of main characters in the movies & books (shown as

EXAMPLE 1

Book: The Kite Runner
Character One: Assef
Character Two: Sohrab
Sentences*: "Sohrab threatens Assef with his slingshot and when Assef lunges at him, Sohrab shoots him in the eye, allowing Amir and Sohrab to escape."*
o Sohrab *threatens* Assef
o Sohrab *shoots* Assef
o Sohrab *lunges at* Assef
o Sohrab *kills* Assef
o Sohrab *finds* Assef

EXAMPLE 2

Movie: Lord of the Rings: Return of the King
Character One: Frodo Baggins
Character Two: Gollum
Sentences*: "Gollum betrays Frodo, leaving him in the lair of giant spider Shelob."*
"When Sam and Frodo are captured by the Rangers of Ithilien, Frodo reveals Gollum's presence to spare his life; Gollum nevertheless feels betrayed and begins plotting against the new master."
"Frodo and Sam take pity on him, understanding the burden of the ring."
o Frodo Baggins *was captured by* Gollum
o Frodo Baggins *was forced to endure* Gollum
o Frodo Baggins *made life miserable for* Gollum
o Frodo Baggins *left* Gollum
o Frodo Baggins *betrays* Gollum

FIGURE 5.1: Sample HITs

MovieCast & BookCast respectively), made up of relatively simpler sentences. Table 5.1 summarizes the test samples.

	Prominent Sample		**Random Sample**	
	No. of movies/books	No. of questions	No. of movies/books	No. of questions
MovieCast	20	109	59	75
MoviePlots	20	130	58	75
BookCast	20	135	49	75
BookPlots	20	167	61	75

TABLE 5.1: Datasets for evaluation

5.3.1 Crowdsourcing Setup

Each HIT shows the sentences from which the options were generated to the worker. When deploying the game on CrowdFlower [24], as there is no provision to swap the character roles, we generated twice the number of options in a question by considering both orders of the character roles. Each question is answered by 5 different workers on CrowdFlower.

This section presents results on the precision and recall of acquiring relationships with HIGGINS , as well as insights on inter-judge agreement and the overall cost. All results here are from the full configuration that integrates all assets described in the previous chapter. The mixture-model parameters (see Section 4.5) were set to $\alpha_{stats} = 0.991$, $\alpha_{WordNet} = 0.0049$, $\alpha_{ConceptNet} = 0.0039$.

5.3.1.1 Conservative and Liberal settings

We manually constructed ground-truth answers for the two datasets. Since some sentences produce multiple relationships, we consider all correct answers for the ground-truth. In cases where a relation exists in the sentence but is not captured by any of the generated answers, OTHER is considered as ground-truth. For character role pairs that do not describe any relation in the sentences, we consider NORELATION as the ground-truth. This naturally leads us to two different settings for evaluating the HC results.

In the **conservative** evaluation, we only consider questions from the ground-truth that do not have OTHER or NORELATION as answers, thereby evaluating the goodness of the options generated by our system. In the **liberal** evaluation, we calculate precision and recall over all the answers including NORELATION and OTHER.

5.4 Measures of quality: Precision and Recall

5.4.1 Precision

The precision measure in the field of IR is defined as the fraction of answers retrieved by the system that are relevant or correct. The generated HIGGINS questions, however, produce multiple correct answers for each question. Therefore, we define the following notion of precision to account for multiple possible answers for a question.

Let $Q = \{q_1, q_2, .., q_i, ..q_n\}$ be the set of generated questions. We present the options to the worker in terms of facts f constructed using the entity pair and the generated top-5 relations. For each q_i we have $\{f_1, f_2, ...f_j, ..f_{12}\}$ options provided to the players, where $\{f_1, .., f_5\}$ are facts obtained top-5 candidate answers and $\{f_6, .., f_{10}\}$ are obtained by swapping the entity roles. f_{11} and f_{12} are the NORELATION and OTHER options, respectively. For each $f_j \in q_i$ we have,

$$f_j = \begin{cases} k & \text{count of workers who chose } f_i \\ 0 & \text{otherwise} \end{cases}$$

Let t_j be a boolean variable that represents the correctness of a fact to question q_i. We have $t_j \in \{0, 1\}$ where $t_j = 1$ if the answer is in the ground-truth of correct answers, and 0 otherwise.

The precision p_i for questions q_i is calculated as,

$$p_i = \frac{\sum_j f_j * t_j}{\sum_j f_j}. \tag{5.1}$$

The overall precision P we report is the average over all questions in the set Q,

$$P = \sum_i^n \frac{p_i}{n}. \tag{5.2}$$

5.4.2 Recall

Conventionally, recall is defined as the fraction of relevant answers retrieved over the total relevant answers that exist. However, it is not feasible to enumerate all relational phrases in our vast dictionary of relations (~116000 relations), that apply to each of the entity pairs. Hence we generated ground-truth – the set of correct answers generated by HIGGINS for each question, using expert human judgements. We define our recall measures as the fraction of correct answers in the ground-truth that the crowdsourced HITs obtained.

We use two different measures of recall when aggregating the workers' answers.

5.4.2.1 Recall-Any

Recall-Any measures the fraction of relations in the ground-truth chosen by at least one worker. For question q_i we set a boolean $g_j = 1$ if at least one worker chose the fact f_j with $t_j = 1$. The recall-any per question is

$$r_i = \frac{\sum_j t_j * g_j}{\sum_j t_j}. \tag{5.3}$$

5.4.2.2 Recall-Majority

Recall-Majority measures the fraction of relations in the ground-truth that were chosen by majority of the workers. For question q_i we set a boolean $g_j = 1$ if the "majority" of the workers who answered q_i chose a the fact f_j with $t_j = 1$. The recall-majority per question q_i then is

$$r_i = \frac{\sum_j t_j * g_j}{\sum_j t_j}. \tag{5.4}$$

The overall Recall-Any and Recall-Majority are averaged over all questions q_i in Q,

$$R = \sum_i^n \frac{r_i}{n}. \tag{5.5}$$

Note that our setting allows multiple correct answers per question. Therefore, our notion of majority allows that different users choose different answers as long as any of the chosen answers is correct.

5.5 Quality of Higgins results

The results are shown in Tables 5.2, 5.3 and 5.4. We see that Higgins generally achieved high precision and, even in the majority mode, decent recall.

	Prominent Sample		**Random Sample**	
	Conservative	Liberal	Conservative	Liberal
MovieCast	0.661	0.795	0.732	0.848
MoviePlots	0.558	0.678	0.633	0.715
BookCast	0.600	0.754	0.529	0.699
BookPlots	0.695	0.781	0.601	0.723

TABLE 5.2: Precision Measurements for HIGGINS

	Prominent Sample			
	Conservative		Liberal	
	Recall-Any	Recall-majority	Recall-Any	Recall-majority
MovieCast	0.673	0.578	0.963	0.862
MoviePlots	0.625	0.616	0.927	0.880
BookCast	0.633	0.578	0.956	0.867
BookPlots	0.743	0.663	0.948	0.855

TABLE 5.3: Recall Measurements for HIGGINS with Prominent Sample

	Random Sample			
	Conservative		Liberal	
	Recall-Any	Recall-majority	Recall-Any	Recall-majority
MovieCast	0.727	0.720	0.961	0.950
MoviePlots	0.711	0.547	0.978	0.840
BookCast	0.591	0.520	0.921	0.853
BookPlots	0.689	0.603	0.971	0.850

TABLE 5.4: Recall Measurements for HIGGINS with Random Sample

5.6 CROWDSOURCING COSTS

Regarding the cost of crowdsourcing, we looked at the inter-judge agreement as a function of the number of contributors for a question. We broke this down by the number of workers. The results reflected by the Fleiss Kappa measure and the average precision are shown in Table 5.5. We see that the agreement increases with more users per questions, but quickly saturates. So for high-quality HC contributions, we do not need a large number of users per question. Of course, this holds when the questions and candidate answers are generated in a meaningful way. This is exactly the contribution of our IE engine for a well-prepared and cost-effective HC phase.

	Prominent			**Random**		
No. of workers	**3**	**4**	**5**	**3**	**4**	**5**
Fleiss Kappa	0.470	0.470	0.460	0.510	0.500	0.510
Avg. Precision	0.722	0.748	0.742	0.729	0.739	0.739
Avg. Recall-Any	0.843	0.907	0.914	0.863	0.910	0.950

TABLE 5.5: Inter-Judge Agreement

5.7 Baseline comparisons – IE and HC Only

Our main hypothesis in this paper is that the combination of IE and HC yields synergies and thus provides benefits that neither an IE-only nor an HC-only method can achieve. For experimental insight in this hypothesis, we compared HIGGINS against two baselines that use solely IE or solely HC.

5.7.1 IE-only Baseline

While there exist several OpenIE methods, we consider the OLLIE [49] system for performance comparison as its extraction functionality comes close to HIGGINS.

OLLIE is a high performance OpenIE system that performs bootstrap learning of patterns in dependency parse trees to detect factual triples. The OLLIE extractor further analyzes the structure of the dependency trees enabling it to detect non-factual assertions such as beliefs, hypothetical or conditional sentence constructs. We ran OLLIE on our datasets and manually identified all correct extractions computed by this IE-only method. Since OLLIE produces relations directly from the textual input while HIGGINS presents the relations from its dictionary of relations, we cannot easily define a notion of recall to compare the two systems. Even the comparison based on the standard notion of precision would be debatable or unfair in this setting, as each system may capture relations in different representations. Rather than using heuristics for thresholding on extraction confidence measures, we compared the set of OLLIE extractions directly against the extractions by HIGGINS and vice versa. This is done as follows.

Let $H \subseteq Q$ be the set of questions for which HIGGINS provided at least one truly correct answer. Let $I \subseteq Q$ be the set of questions for which OLLIE generated a correct fact. Now we can compare the performance of OLLIE relative to what HIGGINS returned, and vice versa. Table 5.6 shows values for $|H|$, $|I|$, $|H \setminus I|$, $|I \setminus H|$, and $H \cap I$ respectively.

	Prominent					**Random**				
DataSet	$\|H\|$	$\|I\|$	$\|H \setminus I\|$	$\|I \setminus H\|$	$\|H \cap I\|$	$\|H\|$	$\|I\|$	$\|H \setminus I\|$	$\|I \setminus H\|$	$\|H \cap I\|$
MovieCast	76	18	61	3	15	58	27	33	2	25
MoviePlot	81	30	57	6	24	54	16	40	2	14
BookCast	100	45	61	6	39	47	10	38	1	9
BookPlot	128	48	84	4	44	51	19	34	2	17

TABLE 5.6: Comparison of HIGGINS (H) and OLLIE (I)

The numbers in Table 5.6 clearly show that HIGGINS provides many more correct facts, compared to OLLIE (shown by $|H \setminus I|$). Conversely, OLLIE provided only very few correct facts that were not acquired by HIGGINS (shown by $|I \setminus H|$). This shows that the joint IE-and-HC method of HIGGINS outperforms state-of-the-art IE-only methods.

5.7.2 HC-only Baseline

As a baseline for an HC-only method, we generated candidate answers for questions in an uninformed random manner but using heuristics to avoid meaningless candidates. For each pair of character roles in a question, we picked the pool of candidates from the dictionary of relations that have a non-empty word-level Jaccard overlap with the words on the dependency path between the characters. A sample of five relations chosen randomly from this pool were presented to the worker. The precision and recall measurements are summarized in table 5.7 and 5.8.

	Prominent Sample		**Random Sample**	
	Conservative	Liberal	Conservative	Liberal
MovieCast	0.316	0.601	0.196	0.594
MoviePlots	0.155	0.508	0.118	0.459
BookCast	0.111	0.700	0.430	0.708
BookPlots	0.316	0.583	0.149	0.490

TABLE 5.7: Precision Measurements for HC-only Method

		Prominent Sample		Random Sample	
Dataset		Conservative	Liberal	Conservative	Liberal
MovieCast	Recall-Any	0.822	0.953	0.789	0.960
	Recall-Majority	0.355	0.383	0.236	0.272
MoviePlot	Recall-Any	0.674	0.952	0.635	0.898
	Recall-Majority	0.116	0.193	0.148	0.202
BookCast	Recall-Any	0.895	0.926	0.706	0.800
	Recall-Majority	0.118	0.119	0.400	0.480
BookPlot	Recall-Any	0.709	0.962	0.640	0.902
	Recall-Majority	0.206	0.255	0.173	0.200

TABLE 5.8: Recall Measurements for HC-only Method

5.8 INFLUENCE OF HIGGINS COMPONENTS

HIGGINS uses a variety of components, and we were interested in identifying which components are essential for high-quality output and which ones merely contribute marginal benefits. We studied different configurations of HIGGINS, and how this influenced the generated candidate answers presented to the user and how this in turn influenced the overall quality of the HC stage. We compared the *full* system against two variants that used i) only the semantic components (WordNet and

ConceptNet) for the language model, or ii) only the statistical assets (excluding WordNet and ConceptNet). These two variants are referred to as *sem-only* Higgins and *stat-only* Higgins, respectively.

The results over the random sample are shown in Tables 5.9 and 5.10 for the precision and recall, respectively. In all cases, we see that the *full* combination outperforms the other configurations by a large margin. This demonstrates that combining semantic resources and statistics is vital for generating meaningful answer candidates and obtaining high-quality results.

	stats-only		sem-only		full	
Dataset	Conservative	Liberal	Conservative	Liberal	Conservative	Liberal
MovieCast	0.605	0.754	0.713	0.797	0.732	0.848
MoviePlot	**0.252**	0.536	**0.299**	0.516	**0.633**	0.714
BookCast	0.544	0.630	0.477	0.649	0.529	0.699
BookPlot	**0.254**	0.611	**0.318**	0.456	**0.601**	0.723

Table 5.9: Impact of Higgins Variants: Precision Measurements

		stats-only		sem-only		full Higgins	
Dataset		Conservative	Liberal	Conservative	Liberal	Conservative	Liberal
MovieCast	Recall-Any	0.623	0.888	0.663	0.878	0.727	0.951
	Recall-Majority	0.623	0.836	0.720	0.875	0.729	0.960
MoviePlot	Recall-Any	0.339	0.868	0.401	0.795	0.711	0.978
	Recall-Majority	**0.240**	0.613	**0.284**	0.595	**0.547**	0.840
BookCast	Recall-Any	0.528	0.821	0.514	0.918	0.591	0.920
	Recall-Majority	0.513	0.716	0.534	0.794	0.520	0.853
BookPlot	Recall-Any	0.260	0.948	0.399	0.872	0.689	0.971
	Recall-Majority	**0.173**	0.640	**0.253**	0.547	**0.603**	0.850

Table 5.10: Impact of Higgins Variants: Recall Measurements

5.9 Strengths, Limitations, Lessons Learned, and Outlook

As the above results substantiate, Higgins delivers high performance. It consistently and significantly outperforms both the IE-only and HC-only approaches in terms of recall and/or precision. Perhaps even more encouraging is the result that high inter-annotator agreement per question can be reached, even with only a small number of participants. This is crucial, as it implies that the high-quality output of HIGGINS can be achieved at low capital costs (which are a typical concern whenever crowdsourcing is involved).

Limitations: One would expect that HC-only methods can achieve near-perfect results as the number of workers per question is increased, thus aggregating the "wisdom of crowds" over more independent people. Apart from the fact that this may quickly become prohibitively expensive, our experience with a small subset of questions even makes us sceptical whether the user inputs actually converge to an agreed-upon fact in (nearly) all cases. Some relationships between characters in movies or books are very subtle, and the sentences that express them in narrative descriptions are sometimes quite sophisticated. An example is: "*With a single gunshot, Blondie severs the rope, dropping Tuco face-first onto his share of the gold. Blondie smiles and rides off as Tuco curses him in rage, shouting, 'Hey Blonde! You know what you are? Just a dirty son of a bitch!'*", from the movie "Good, the Bad, and the Ugly". Should this lead to a fact ⟨*Blondie helped Tuco*⟩ or ⟨*Blondie saved Tuco*⟩ or ⟨*Blondie likes Tuco*⟩ or ⟨*Tuco likes Blondie*⟩? Such situations pose a daunting challenge to both humans and automated IE. It is widely open whether a pure HC method or a combined IE & HC approach are suitable for such difficult inputs. This could be an interesting direction of future work.

IE for HC: All our experiments confirmed that, in most cases, having a smart IE phase before embarking on HC is very beneficial. The quality of the candidate answers generated for multiple-choice questions is substantially improved by our combination of statistical (language-model-based) and semantic (ontology or thesaurus-based) assets. We believe that our judicious combination of the right assets and controlling their interplay is one of the main contributions of HIGGINS.

HC for IE: In addition to IE boosting the performance of HC, there is also a virtuous feedback from HC to IE. As we acquired more and more correct relationships from the HC stage of HIGGINS, we can use these to improve the semantic dictionary and the language-model statistics used in IE to drive the generation of candidate answers. Most notably, the translation LM that estimates the relatedness of different phrases is crucial. Texts sometimes use subtle or rare formulations (sometimes even with irony) to denote a relationship. For example, phrases like "*develops feelings for*" or "*is drawn closer to*" are rare, so their relatedness with "*falls in love with*" cannot be estimated that well by IE techniques alone. Once we have HC results for such cases, we can feed them back into the IE statistics and improve the estimates of the translation LM's.

Cost of Crowd-sourced Experimental Research: In our experimental studies, we tried to scale up as much as possible. We realize, though, that we did not reach out to the level of large game communities or big online groups. We believe that our approach has the potential for further scaling, but at this point, the total cost of experiments has been a limiting factor. In total, our experiments involved more than 12,800 HITs on CrowdFlower. Although each HIT costs only 5 cents, the total cost is a concern. In addition and even more troublesome, we had to create ground-truth output for all instances that were evaluated. In total, these were more than 20,000 question-answer pairs. We are not aware of similarly intensive studies on crowdsourcing for knowledge acquisition. All our experimental data is made available for further research in the community at http://www.mpi-inf.mpg.de/yago-naga/higgins/.

Outlook: HIGGINS manages to produce high quality knowledge at low crowd costs. This begs the interesting question, at least from a theoretical point of view: If money were not an issue, how much can we improve our KA engines, say by engaging a much greater number of users per question? Put differently: is gathering the "wisdom of the crowds" by engaging large numbers of humans, capable of improving substantially the quality of KA? Perhaps surprisingly at first sight, our preliminary evidence is not entirely supportive of this. The complex relationships we are seeking to discover are by their very nature very subtle and they typically appear in narratives in various sophisticated ways. As such, they typically escape the attention of many. Intrigued by this observation, our future plans include the study of robust KA engines (for our problem domain space) which will be able to predict, subject to cost constraints, in which situations extensive IE involvement is needed and what will be its expected quality improvement/impact.

6 CHAPTER

Building Human Computing Games for Knowledge Acquisition

6.1 Overview

The previous chapter provided experimental proof on the effectiveness of the HIGGINS system by constructing and evaluating HITs for crowdsourcing platforms. On the other hand, the output of the HIGGINS IE phase can also form the basis for factoid-based human computing games. In this chapter we present HC games for fact-based knowledge acquisition, discuss how different games for KA can be designed with HIGGINS generating the game questions in the background. We present MOVIEWIZARD, BOOKWIZARD and MOVIEGURUS games that demonstrate the viability of HIGGINS for game-based relational fact acquisition.

6.2 Game Design for Fact Acquisition

A *knowledge acquisition game* is a *sequence of interactions* between the system and one or more players, centered around the notion of knowledge triples or knowledge quads (if context is included). In each interaction, the system generates a knowledge triple or quad, presents it to the player(s), and asks the player(s) to complete the missing slot values. In this case, the player inputs the values for the missing slots or chooses from a list of candidate values provided by the system, to generate a fully bound quad. Alternatively the system provides a fully bound quad and determines the truthfulness of the quad through the player's inputs.

This basic game structure can be extended in several ways:

- *Anonymized slots:* The entity slots in a knowledge quad ($e1$ and $e2$ in the quad $\langle c, e1, r, e2 \rangle$) may be anonymized with codenames provided by the system, and consistently used across multiple interaction steps of a game.
- *Multiple players:* Triples or quads can be shown to a single player or to multiple players simultaneously. In the latter case, we either aim to achieve agreement among players or we assign roles to players so that one player gives hints and the other player(s) attempts to guess the missing values.
- *Clues and callbacks:* Players may actively ask for clues; these would be provided by the system or another player. The clues are completely filled knowledge triples or quads: without variables, but possibly with anonymized slots.

System Perspective versus Player Perspective.
KA Games are perceived differently by the system and by the players. The system understands the notion of knowledge triples and quads and how they can be built from an underlying database of entities, semantic classes, relations, and relation instances. Players, on the other hand, need to view these items in a simpler and intuitive form, either as natural language text with gaps to be filled or as form fields. Moreover, humans may refer to entities in shorthand form or by role names rather than using "official" entity names, e.g., using "*Tom*" or "*the family lawyer*" instead of "Tom Hagen", or using "it Lady Di" or "*the queen of hearts*" instead of "Diana, Princess of Wales". This holds for relations as well, for example, the players may provide "*ordered his goons to assassinate*" or "*gave the order to eliminate*" instead of simply "*ordered to kill*". This is where we can exploit the vast entity name dictionaries, the lexicon of relations and the phrase translation model in HIGGINS.

6.3 HIGGINS GAMES

In the HIGGINS system, the IE engine generates fact hypotheses in the form of incomplete (partially bound) or unverified (fully bound) $\langle c, e1, r, e2 \rangle$ knowledge quads. Based on which of the slots are unbound in these fact hypotheses, different game scenarios at the HC stage are possible. In these cases, the system designs questions by withholding one of the c, $e1$, r or $e2$ values and presenting alternatives to the player(s). The player chooses the appropriate alternative, providing a completed knowledge quad to the system. Alternatively a fully bound quad can be presented to the player to be validated.

6.3.1 Game Scenarios

The following list describes different possible game scenarios by leaving out different slots of the fact hypotheses as variables, and presents the benefits to the system in each of these scenarios.

- *Variable* $e1$ *or* $e2$*:* In this case the system leaves out $e1$ or $e2$ to be determined by the player. This game scenario is helpful for the coreference resolution task. Candidates for the unknown coreference are generated by HIGGINS and presented as multiple-choice to the player.
- *Variable* r*:* By leaving r unbounded, the player is tasked to determine the appropriate relation in which the two entities participate within the context. This game scenario can be applied by the system for the relation extraction task. Here's an example question: "In the movie `Shawshank Redemption`, how are Andy Dufrense and Ellis Boyd Red related?". HIGGINS generates possible *relevant* relation phrase candidates for P by processing the dependency path between $e1$ and $e2$ and ranking the phrases in its dictionary.
- *Variable* c*:* An interesting game scenario can be generated when the context c is unbound. A set of $\langle e1, r, e2\rangle$ triples for the same c (movie or book) are presented to the player. The player, in a series of interactions with the system or another player, tries to determine C. The key idea here is to consistently anonymize $e1$ and $e2$ when presented to the partner. In a two-player setting, one player constructs $\langle e1, r, e2\rangle$ triples with anonymized $e1$ and $e2$ and a randomly-paired partner tries to guess c. The constructed $\langle e1, r, e2\rangle$ triples from the game are used by the system to form new facts.

Using movie and book narratives, we build two games, MOVIEWIZARD and BOOKWIZARD, that are based on the scenarios where r or $e1/e2$ are unbound. MOVIEGURUS, a two-player game, demonstrates the scenario where c is unbound. These games are available here: http://higgins.mpi-inf.mpg.de.

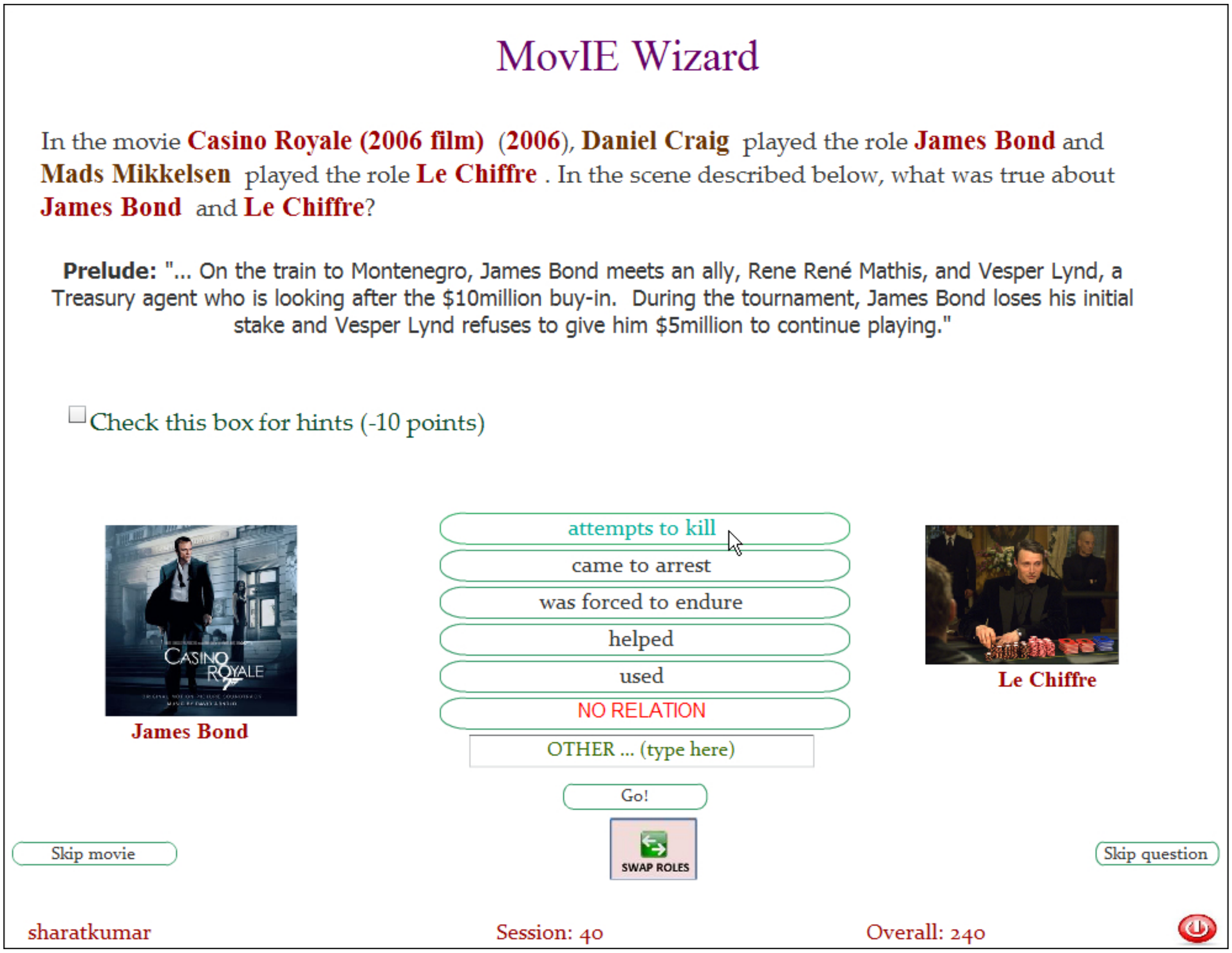
6.3.2 Player Reward Strategy for HIGGINS Games

HIGGINS has no gold standard solutions to directly evaluate the correctness of the players' responses. In a single-player environment where players play the game independently, we deal with this issue by awarding *future agreement bonuses* based on the fraction of the players that agreed with the player's responses. The objective of each player thus is to maximize her long-term overall score and obtain a top rank among all players. And the objective from the system's point of view is to place more faith in the highly-agreed responses, as we expect that these will yield a better KB.

6.4 MOVIEWIZARD & BOOKWIZARD GAMES

The MOVIEWIZARD game aims to establish relations between movie character roles by posing factoid questions to the player based on its storyline (BOOKWIZARD is a functionally identical game on book stories). The player starts the game by picking a movie of his choice from the list of movies provided. The game server provides a question from the selected movie which consists of three parts (see Fig. 6.1). The first part states the question on two prominent characters in the movie. The second part contains hidden cues which are sentences about the character-pair in the plot. The third part contains five relation phrases presented in multiple-choice, an option for no relation between the pair, and an optional input-field. The task of the player is to choose an appropriate relation that holds between the two characters. If there exists a relation and none among the multiple-choice holds, the player types in a relational phrase. Otherwise, the option for no relation has to be chosen. The player may choose to use the hidden cues to arrive at an answer.

Scoring scheme. The player score has two components: a) fixed points, called base points, for responding with an answer to the question and b) bonus points, awarded to the player based on the fraction of players who responded with the same answer. If the player chooses to utilize the hidden cue, the base points for the question are halved. Upon skipping the question no points are added to the score. The game server presents only unanswered questions to the player. The overall score of the player is determined by aggregating the base and bonus points. The position of the player among all players, ranked on overall score, is shown at the end of a game session. The objective of the player is to maximize the score and achieve top ranking among all the players.



MovIE Wizard

In the movie **Casino Royale (2006 film)** (**2006**), **Daniel Craig** played the role **James Bond** and **Mads Mikkelsen** played the role **Le Chiffre** . In the scene described below, what was true about **James Bond** and **Le Chiffre**?

Prelude: "... On the train to Montenegro, James Bond meets an ally, Rene René Mathis, and Vesper Lynd, a Treasury agent who is looking after the $10million buy-in. During the tournament, James Bond loses his initial stake and Vesper Lynd refuses to give him $5million to continue playing."

□ Check this box for hints (-10 points)

FIGURE 6.1: MOVIEWIZARD game

6.5 Game-based User Study

We conducted a user-study of the MovieWizard game using Wikipedia movie plot sections from 14 popular movies. In total, we generated 220 multiple-choice questions from individual sentences in the plot. We ran the game over the generated questions with student volunteers in our lab. None of the players was involved with the Higgins project. In total, 14 players participated. In the game mode, the player is given the choice of selecting movies that she/he is familiar with. In addition to the top-5 candidate answers and the options NONE and OTHER, users also have an option to swap the roles of the two entities in a question. This is useful, when the orientation of a suggested relation is wrong (e.g., with a candidate answer "James Bond tortures Le Chiffre").

In the game, players are shown a prelude text to provide context about a specific situation in the movie. The subsequent sentence is the one from which the question is generated, but the sentence is not directly shown but offered as a hidden cue. Players can earn points when a question is answered. When they ask to reveal the cue, the points to be earned are halved.

We manually constructed ground-truth answers for the 220 questions. Since some sentences produce multiple relationships, we consider all correct answers for the ground-truth. For sentences that do not describe any relation, we consider NONE as the ground-truth. In cases where a relation exists in the sentence but is not captured by any of the generated answers, OTHER is considered as ground-truth. As with the case of crowdsourced HITs, we consider **conservative** and **liberal** settings defined in 5.3.1.1 for our game-based evaluation.

We use the precision/recall measures defined in 5.4. The results for conservative and liberal settings are shown in Table 6.1. We see that the game achieved high precision, and even in the majority mode, high recall.

	Conservative	**Liberal**
Precision	0.815	0.829
Recall-any	0.893	0.897
Recall-majority	0.854	0.882

TABLE 6.1: Precision & Recall for MOVIEWIZARD Game

Regarding the cost of crowdsourcing, we looked at the inter-judge agreement as a function of the number of contributors for a question. We broke this down by the number of players.

Players	**Questions**	**Fleiss Kappa**
2	209	0.7176
3	199	0.7339
4	172	0.6956
5	146	0.6520
6	118	0.6979

TABLE 6.2: Inter-Judge Agreement for MOVIEWIZARD game

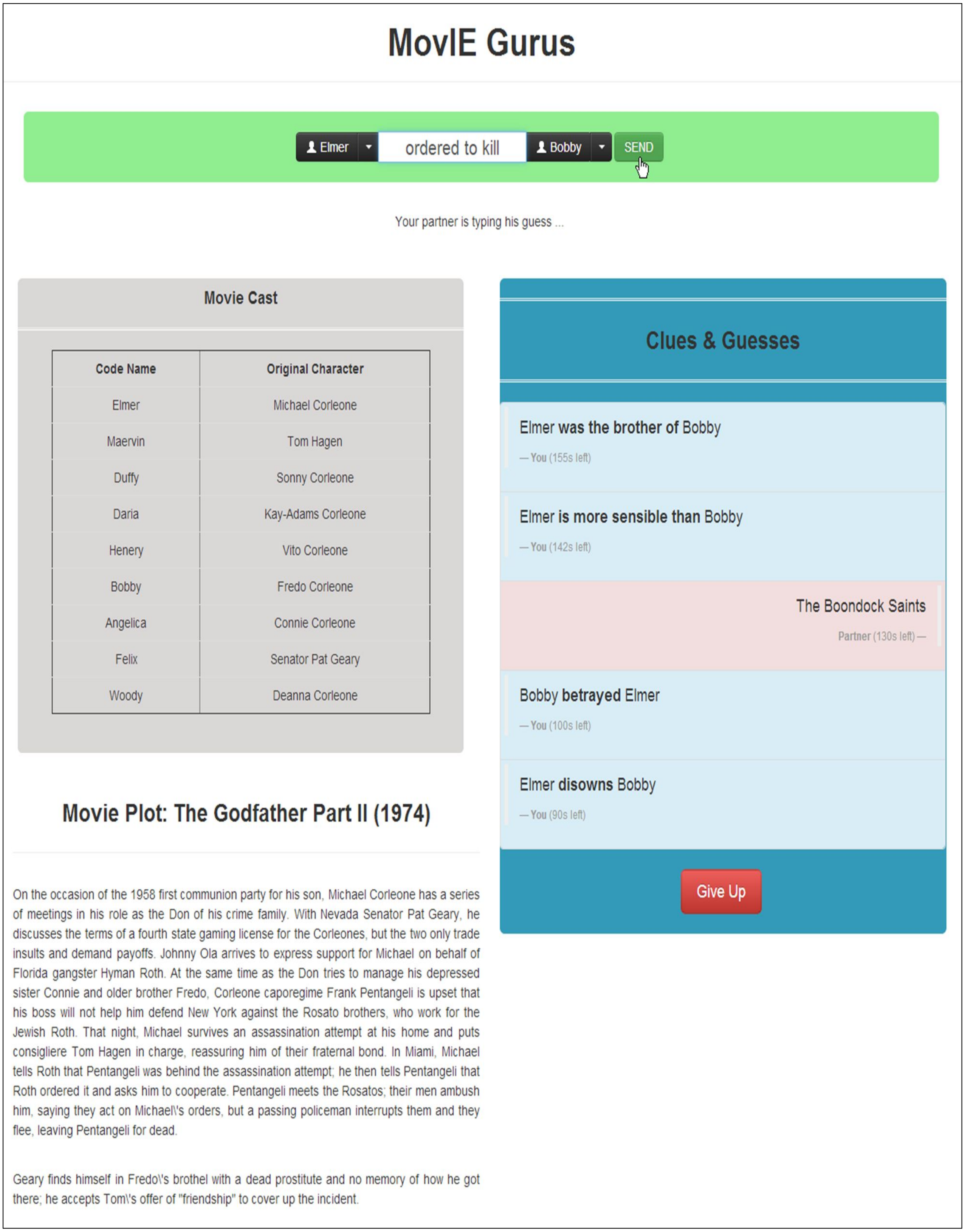
The results are shown in Table 6.2. We see that the agreement increases with more users per questions, but quickly saturates. So for high-quality HC contributions, we do not need a large number of players per question from the system perspective.

6.6 THE MOVIEGURUS GAME

MOVIEGURUS is a two-player interactive fact-generation game. It is an inversion-problem game (see [39]) wherein the system does not provide any candidates, and the relational facts are generated by the interactions between randomly paired players.

Game Play. The game is played in multiple rounds. In each round, one player assumes the player of "helper", and is assigned to a partner, randomly picked from all players of the game. The partner then assumes the role of a “guesser”. Upon pairing, the helper is presented with a movie title and its storyline. The helper is also provided with the list of characters in the movie and codenames corresponding to each of them. These purpose of the codenames is to hide the identity of the character roles in the movie and they are randomly chosen by the system from a pool.

The helper chooses any pair of characters from the list and types in a relational phrase. This phrase along with the corresponding codenames is shown on the screen of the guesser to help guess the movie title. The helper repeats the process until the guesser correctly identifies the movie title, in which case the round is won and both the players earn points. The round is lost when either of the players choose to give up or upon stipulated time out. From a successful round, the system benefits by harnessing the fact triples that are constructed by the player-typed relational phrases and the respective pairs of movie characters. Figures 6.2 and 6.3 show screenshots of the interactions between the helper and guesser in a round (here system presented the movie “The Godfather Part II”).



Your partner is typing his guess ...

Movie Cast

Code Name	Original Character
Elmer	Michael Corleone
Maervin	Tom Hagen
Duffy	Sonny Corleone
Daria	Kay-Adams Corleone
Henery	Vito Corleone
Bobby	Fredo Corleone
Angelica	Connie Corleone
Felix	Senator Pat Geary
Woody	Deanna Corleone

Elmer **was the brother of** Bobby

— **You** (155s left)

Elmer **is more sensible than** Bobby

— **You** (142s left)

The Boondock Saints

Partner (130s left) —

Bobby **betrayed** Elmer

— **You** (100s left)

Elmer **disowns** Bobby

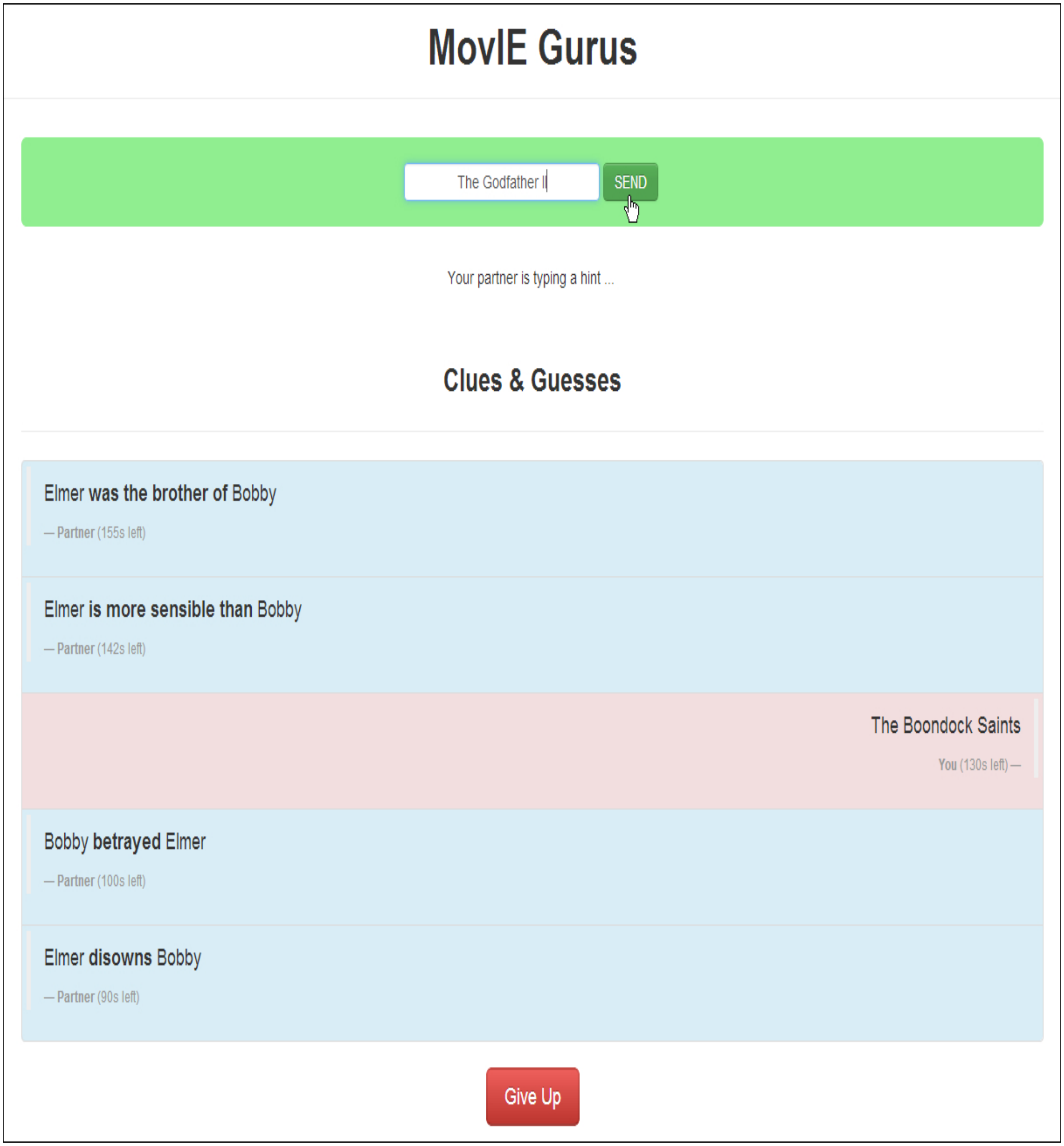
— **You** (90s left)

Movie Plot: The Godfather Part II (1974)

On the occasion of the 1958 first communion party for his son, Michael Corleone has a series of meetings in his role as the Don of his crime family. With Nevada Senator Pat Geary, he discusses the terms of a fourth state gaming license for the Corleones, but the two only trade insults and demand payoffs. Johnny Ola arrives to express support for Michael on behalf of Florida gangster Hyman Roth. At the same time as the Don tries to manage his depressed sister Connie and older brother Fredo, Corleone caporegime Frank Pentangeli is upset that his boss will not help him defend New York against the Rosato brothers, who work for the Jewish Roth. That night, Michael survives an assassination attempt at his home and puts consigliere Tom Hagen in charge, reassuring him of their fraternal bond. In Miami, Michael tells Roth that Pentangeli was behind the assassination attempt; he then tells Pentangeli that Roth ordered it and asks him to cooperate. Pentangeli meets the Rosatos; their men ambush him, saying they act on Michael\'s orders, but a passing policeman interrupts them and they flee, leaving Pentangeli for dead.

Geary finds himself in Fredo\'s brothel with a dead prostitute and no memory of how he got there; he accepts Tom\'s offer of "friendship" to cover up the incident.

FIGURE 6.2: MOVIEGURUS game – Helper screen



The Godfather II

Your partner is typing a hint ...

Clues & Guesses

Elmer was the brother of Bobby
— Partner (155s left)

Elmer is more sensible than Bobby
— Partner (142s left)

The Boondock Saints
You (130s left) —

Bobby betrayed Elmer
— Partner (100s left)

Elmer disowns Bobby
— Partner (90s left)

FIGURE 6.3: MOVIEGURUS game – Guesser screen

6.7 Discussion and Outlook

Unlike crowdsourced workers, game players are motivated by competition and their desire to be entertained. By designing interesting games that attract enthusiasts in specific domains such as movies, books, current affairs, sports, bio-medicine etc., factual knowledge can be acquired through game play. In this work, we used movie and book narratives, and presented interesting game scenarios in HIGGINS for different KA tasks. We demonstrated the MOVIEWIZARD, BOOKWIZARD and MOVIEGURUS games that help acquire relations between character roles in movie and book stories. Our in-house experiment of the MOVIEWIZARD game with a small set of players showed high precision and recall. Also, we found that answers from only a handful of players are sufficient to obtain good inter-player agreement. However, to fully leverage the potential of HC games for large-scale fact acquisition, targeting communities of interested players on the Web is essential. This requires deeper study on game-level factors, which is beyond the scope of this work and an interesting direction in itself.

7 CHAPTER

Conclusion

In this thesis we presented the design and implementation of a system that combines information extraction with human computing for advanced knowledge acquisition tasks. We recapitulate the contributions below and outline some directions for potential future work.

7.1 Thesis Contributions

We presented a novel system architecture, called HIGGINS, that combines an IE engine with an HC engine for effective knowledge acquisition. This HIGGINS strategy enables the use of automated IE methods to generate questions and possible answers that are cast into crowdsourced HITs. This way, the cost of the HC phase can be controlled and is substantially reduced compared to an HC-only approach. We successfully applied this combination to the difficult task of compiling relationships between characters in movie and book narratives.

The HIGGINS IE engine derives entity-relation triples from Web corpora aiming for high recall. The resulting triples, which contain noisy candidates, are ranked by specifically designed statistical language models. By harnessing semantic and statistical resources, the IE engine is able to create meaningful questions and answer candidates, fed into the HC engine for crowdsourcing. Our experiments demonstrate the benefits of our system in terms of both output quality and HC cost. Using the right combination of these resources alleviates the sparseness problem of relational phrases as demonstrated by its superior performance against its underlying statistical or semantic components. Our experiments on movie and book narrative summaries show that HIGGINS performs well in terms of precision and recall, and scores higher than state-of-the-art OpenIE systems or a pure crowdsourced approach.

Finally, we demonstrated how the fact candidates generated by the HIGGINS IE engine can be used to construct human computing games for fact acquisition. The MOVIEWIZARD and BOOKWIZARD games create multiple-choice questions with relation candidates for the players to choose from. Players are rewarded points based on output agreement and majority voting is used to determine correctness of facts. MOVIEGURUS is an interactive two-player game in which one of the two randomly-paired players helps the other player guess a context provided by the system, by constructing fact triples related to the context. By mapping the construction of correct fact triples to the success of the players, the game collects valid fact triples through their interactions.

7.2 OUTLOOK AND FUTURE WORK

The methods presented in this dissertation deviate significantly from existing approaches in the way humans participate in text-based knowledge acquisition processes. We utilize human inputs to assess and validate fact candidates generated by automated IE; by transforming them into HITs or game questions. In doing so we need to limit the number of candidates in order not to overwhelm the contributors. Producing redundant HITs is not desirable as costs present a scalability bottleneck. We achieve a fine balance by developing appropriate ranking models for fact candidates. There are several directions forward from here:

A. Active Learning

Although HIGGINS delivers decent precision, we expect its performance to improve if the candidates provided by the contributors are incorporated into the system.

- **Improving IE.** Feedback obtained from crowdsourced output can be used to improve the language model statistics at the IE phase to drive generation or pruning of certain candidates. This will be especially useful for boosting infrequent phrases at the IE stage.

- **Improving HC.** As HIGGINS produces quick consensus when good candidates are presented, HITs with perfect agreement require fewer overall judgements. This can be exploited to further reduce crowdsourcing costs. On the other hand, IE engine generates candidates even in cases where no relations exist between the entity pairs, and these can quickly be eliminated using the feedback.

B. Game Trails

In HIGGINS games, players interact with the system or with other players to generate fact triples. The sequence of interactions i.e., the clues presented and the corresponding player responses is what we term 'game trails'. Generated game trails can be mined to establish new facts, derive fact confidence, and strength of relation-phrase pairs. Other interesting associations such as important relations pertaining to an entity or context can also be obtained (in the MOVIEGURUS game, for example). Similar to the crowdsourcing case, we expect active learning on game trails to be beneficial for effective fact acquisition and inciting more player interest in the game play.

C. Taxonomy for Relations in Narratives

Although the relations we consider have subtle differences, some relations are abstract and subsume many different relational phrases (for example, i) "*wanted to kill*" subsumes "*shot at*", "*laid a trap for*" etc. ii) "X *dreamt of* Y" subsumes "X *dreamt of killing* Y", "X *dreamt of marrying* Y" etc.). By developing a taxonomy for relations (and their phrases) in narratives and stories, one can target specific relations while harvesting relational facts, further reducing crowdsourcing costs.

HIGGINS generates entity-relation triples for knowledge bases. As a by-product it also produces a relation-to-phrase mapping for long tail relations, that will be refined by incorporating feedback from HITs. We expect that this resource will find use in paraphrasing, summarization and semantic search applications.

Bibliography

[1] 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 ACM SIGMOD International Conference on Management of Data*, pages 1247–1250, Vancouver, Canada, 2008. ACM.

[2] Freebase. http://freebase.com.

[3] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, and Zachary Ives. Dbpedia: A nucleus for a web of open data. In *Proceedings of the International Semantic Web Conference (ISWC)*, pages 11–15, Busan, Korea, 2007. Springer.

[4] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: a core of semantic knowledge. In *Proceedings of the International World Wide Web Conference (WWW)*, pages 697–706, Banff, Alberta, Canada, 2007. ACM.

[5] Douglas B. Lenat and R. V. Guha. *Building Large Knowledge-Based Systems; Representation and Inference in the Cyc Project*. Addison-Wesley Longman Publishing, Boston, Massachusetts, USA, 1st edition, 1989.

[6] Wentao Wu, Hongsong Li, Haixun Wang, and Kenny Q. Zhu. Probase: a probabilistic taxonomy for text understanding. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 481–492, Scottsdale, Arizona, USA, 2012. ACM.

[7] Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R. Hruschka, and Tom M. Mitchell. Toward an architecture for never-ending language learning. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 1306–1313, Atlanta, Georgia, USA, 2010. AAAI Press.

[8] NELL. http://rtw.ml.cmu.edu.

[9] Marti A. Hearst. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the Conference on Computational Linguistics (COLING) - Volume 2*, pages 539–545, Nantes, France, 1992. ACL.

[10] Nicholas Kushmerick, Daniel S. Weld, and Robert B. Doorenbos. Wrapper induction for information extraction. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 729–737, Nagoya, Aichi, Japan, 1997. Morgan Kaufmann Publishers.

[11] Sergey Brin. Extracting patterns and relations from the world wide web. In *Proceedings of the International Workshop on The World Wide Web and Databases (WebDB)*, pages 172–183, Valencia, Spain, 1998. Springer-Verlag.

[12] Eugene Agichtein and Luis Gravano. Snowball: extracting relations from large plain-text collections. In *Proceedings of the ACM Conference on Digital Libraries*, pages 85–94, San Antonio, Texas, USA, 2000. ACM.

[13] Mary Elaine Califf and Raymond J. Mooney. Relational learning of pattern-match rules for information extraction. In *Proceedings of the National Conference on Artificial Intelligence and Conference on Innovative Applications of Artificial Intelligence (AAAI/IAAI)*, pages 328–334, Orlando, Florida, USA, 1999. AAAI Press.

[14] Robert Baumgartner, Sergio Flesca, and Georg Gottlob. Visual web information extraction with lixto. In *Proceedings of the International Conference on Very Large Data Bases (VLDB)*, pages 119–128, Roma, Italy, 2001. Morgan Kaufmann Publishers.

[15] Oren Etzioni, Michael J. Cafarella, Doug Downey, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S. Weld, and Alexander Yates. Methods for domain-independent information extraction from the web: An experimental comparison. In *Proceedings of the National Conference on Artifical Intelligence (AAAI)*, pages 391–398, Cancun, Mexico, 2004. AAAI Press.

[16] Michele Banko, Michael J Cafarella, Stephen Soderland, Matt Broadhead, and Oren Etzioni. Open information extraction from the web. In *Proceedings of the International Joint Conference on Artifical Intelligence (IJCAI)*, pages 2670–2676, Hyderabad, India, 2007. Morgan Kaufmann Publishers.

[17] Anthony Fader, Stephen Soderland, and Oren Etzioni. Identifying relations for open information extraction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1535–1545, Edinburgh, United Kingdom, 2011. ACL.

[18] Danushka Tarupathi Bollegala, Yutaka Matsuo, and Mitsuru Ishizuka. Relational duality: Unsupervised extraction of semantic relations between entities on the web. In *Proceedings of the International World Wide Web Conference (WWW)*, pages 151–160, Raleigh, North Carolina, USA, 2010. ACM.

[19] Rahul Gupta and Sunita Sarawagi. Joint training for open-domain extraction on the web: Exploiting overlap when supervision is limited. In *Proceedings of the ACM International Conference on Web Search and Data Mining (WSDM)*, pages 217–226, Hong Kong, China, 2011. ACM.

[20] Petros Venetis, Alon Halevy, Jayant Madhavan, Marius Paşca, Warren Shen, Fei Wu, Gengxin Miao, and Chung Wu. Recovering semantics of tables on the web. *Proceedings of the VLDB Endowment*, 4(9):528–538, June 2011.

[21] Anhai Doan, Raghu Ramakrishnan, and Alon Y. Halevy. Crowdsourcing systems on the world-wide web. *Communications of the ACM*, 54(4):86–96, April 2011.

[22] Edith Law and Luis von Ahn. *Human Computation*. Morgan & Claypool Publishers, 2011.

[23] Amazon mechanical turk. `http://www.mturk.com`.

[24] Crowdflower. `http://crowdflower.com`.

[25] Omar Alonso and Ricardo Baeza-Yates. Design and implementation of relevance assessments using crowdsourcing. In *Proceedings of the European Conference on Information Retrieval (ECIR)*, pages 153–164, Dublin, Ireland, 2011. Springer-Verlag.

[26] Panagiotis G. Ipeirotis and Praveen K. Paritosh. Managing crowdsourced human computation. In *Proceedings of the International World Wide Web Conference (WWW). Tutorial*, pages 287–288, Hyderabad, India, 2011. ACM.

[27] Michael J. Franklin, Donald Kossmann, Tim Kraska, Sukriti Ramesh, and Reynold Xin. Crowddb: Answering queries with crowdsourcing. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 61–72, Athens, Greece, 2011. ACM.

[28] Aditya G. Parameswaran and Neoklis Polyzotis. Answering queries using humans, algorithms and databases. In *Proceedings of the Conference on Innovative Data Systems Research (CIDR)*, pages 160–166, Asilomar, California, USA, 2011. CIDR.

[29] Aditya G. Parameswaran, Hyunjung Park, Hector Garcia-Molina, Neoklis Polyzotis, and Jennifer Widom. Deco: declarative crowdsourcing. In *Proceedings of the ACM International Conference on Information and Knowledge Management (CIKM)*, pages 1203–1212, Maui, Hawaii, USA, 2012. ACM.

[30] Pedro DeRose, Xiaoyong Chai, Byron J. Gao, Warren Shen, AnHai Doan, Philip Bohannon, and Xiaojin Zhu. Building community wikipedias: A machine-human partnership approach. In *Proceedings of the IEEE International Conference on Data Engineering (ICDE)*, pages 646–655, Cancun, Mexico, 2008. IEEE Computer Society.

[31] Jiannan Wang, Tim Kraska, Michael J. Franklin, and Jianhua Feng. CrowdER: Crowdsourcing entity resolution. *Proceedings of the VLDB Endowment*, 5(11): 1483–1494, July 2012.

[32] Cristina Sarasua, Elena Simperl, and Natalya Fridman Noy. Crowdmap: Crowdsourcing ontology alignment with microtasks. In *Proceedings of the International Conference on The Semantic Web (ISWC)*, pages 525–541, Boston, Massachusetts, USA, 2012. Springer-Verlag.

[33] Luis von Ahn and Laura Dabbish. Labeling images with a computer game. In *Proceedings of the Conference on Human Factors in Computing Systems (CHI)*, pages 319–326, Vienna, Austria, 2004. ACM.

[34] Tingxin Yan, Vikas Kumar, and Deepak Ganesan. Crowdsearch: exploiting crowds for accurate real-time image search on mobile phones. In *Proceedings of the International Conference on Mobile Systems, Applications, and Services (MobiSys)*, pages 77–90, San Francisco, California, USA, 2010. ACM.

[35] Jia Deng, Wei Dong, Richard Socher, Li jia Li, Kai Li, and Li Fei-fei. Imagenet: A large-scale hierarchical image database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 248–255, Miami, Florida, USA, 2009. IEEE Computer Society.

[36] Luis von Ahn, Mihir Kedia, and Manuel Blum. Verbosity: a game for collecting common-sense facts. In *Proceedings of the Conference on Human Factors in Computing Systems (CHI)*, pages 75–78, Montréal, Québec, Canada, 2006. ACM.

[37] Robert Speer, Catherine Havasi, and Harshit Surana. Using verbosity: Common sense data from games with a purpose. In *Proceedings of the Florida Artificial Intelligence Research Society Conference (FLAIRS Conference)*, Daytona Beach, Florida, USA, 2010. AAAI Press.

[38] Dimitris Karampinas and Peter Triantafillou. Crowdsourcing taxonomies. In *Proceedings of the Extended Semantic Web Conference (ESWC)*, pages 545–559, Heraklion, Crete, Greece, 2012. Springer-Verlag.

[39] Luis von Ahn and Laura Dabbish. Designing games with a purpose. *Communications of the ACM*, 51(8):58–67, August 2008.

[40] Kondreddi Sarath Kumar, Peter Triantafillou, and Gerhard Weikum. Higgins: knowledge acquisition meets the crowds. In *Proceedings of the International World Wide Web Conference (WWW). Companion Volume.*, pages 85–86, Rio de Janeiro, Brazil, 2013. ACM.

[41] Kondreddi Sarath Kumar, Peter Triantafillou, and Gerhard Weikum. Combining information extraction and human computing for crowdsourced knowledge acquisition. In *Proceedings of the IEEE International Conference on Data Engineering (ICDE)*, Chicago, IL, USA, 2014. IEEE.

[42] Kondreddi Sarath Kumar, Peter Triantafillou, and Gerhard Weikum. Human computing games for knowledge acquisition. In *Proceedings of the ACM International Conference on Information and Knowledge Management (CIKM)*, pages 2513–2516, San Francisco, CA, USA, 2013. ACM.

[43] Christiane Fellbaum, editor. *WordNet: An Electronic Lexical Database*. MIT Press, Cambridge, MA, May 1998.

[44] Ian Niles and Adam Pease. Towards a standard upper ontology. In *Proceedings of the International Conference on Formal Ontology in Information Systems (FOIS)*, pages 2–9, Ogunquit, Maine, USA, 2001. ACM.

[45] The Unified Medical Language System (UMLS): Integrating Biomedical Terminology. http://www.nlm.nih.gov/research/umls.

[46] Gene ontology. http://www.geneontology.org/.

[47] Oren Etzioni, Michael J. 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 International World Wide Web Conference (WWW)*, pages 100–110, New York, NY, USA, 2004. ACM.

[48] 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)*, pages 25–26, Rochester, New York, 2007. ACL.

[49] Mausam, Michael Schmitz, Robert Bart, Stephen Soderland, and Oren Etzioni. Open language learning for information extraction. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 523–534, Jeju Island, Korea, 2012. ACL.

[50] Gang Wang, Yong Yu, and Haiping Zhu. Pore: Positive-only relation extraction from wikipedia text. In *Proceedings of the International Semantic Web Conference and Asian Semantic Web Conference (ISWC/ASWC)*, pages 580–594, Busan, Korea, 2007. Springer-Verlag.

[51] Fei Wu and Daniel S. Weld. Autonomously semantifying wikipedia. In *Proceedings of the ACM International Conference on Information and Knowledge Management (CIKM)*, pages 41–50, Lisbon, Portugal, 2007. ACM.

[52] 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 International World Wide Web Conference (WWW). Companion Volume*, pages 229–232, Hyderabad, India, 2011. ACM.

[53] Philip Bohannon, Nilesh Dalvi, Yuval Filmus, Nori Jacoby, Sathiya Keerthi, and Alok Kirpal. Automatic web-scale information extraction. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 609–612, Scottsdale, Arizona, USA, 2012. ACM.

[54] Laura Chiticariu, Rajasekar Krishnamurthy, Yunyao Li, Sriram Raghavan, Frederick R. Reiss, and Shivakumar Vaithyanathan. Systemt: An algebraic approach to declarative information extraction. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 128–137, Uppsala, Sweden, 2010. ACL.

[55] Yuan Fang and Kevin Chen-Chuan Chang. Searching patterns for relation extraction over the web: Rediscovering the pattern-relation duality. In *Proceedings of the ACM International Conference on Web Search and Data Mining (WSDM)*, pages 825–834, Hong Kong, China, 2011. ACM.

[56] Tim Furche, Georg Gottlob, Giovanni Grasso, Omer Gunes, Xiaoanan Guo, Andrey Kravchenko, Giorgio Orsi, Christian Schallhart, Andrew Sellers, and Cheng Wang. Diadem: Domain-centric, intelligent, automated data extraction methodology. In *Proceedings of the International World Wide Web Conference (WWW)*, pages 267–270, Lyon, France, 2012. ACM.

[57] Zornitsa Kozareva and Eduard Hovy. Learning arguments and supertypes of semantic relations using recursive patterns. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1482–1491, Uppsala, Sweden, 2010. ACL.

[58] Roberto Navigli and Simone Paolo Ponzetto. Babelnet: building a very large multilingual semantic network. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 216–225, Uppsala, Sweden, 2010. ACL.

[59] Marius Paşca. Ranking class labels using query sessions. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (HLT)*, pages 1607–1615, Portland, Oregon, USA, 2011. ACL.

[60] Sebastian Riedel, Limin Yao, and Andrew McCallum. Modeling relations and their mentions without labeled text. In *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases (ECML PKDD)*, pages 148–163, Barcelona, Spain, 2010. Springer-Verlag.

[61] Jun Zhu, Zaiqing Nie, Xiaojiang Liu, Bo Zhang, and Ji-Rong Wen. Statsnowball: a statistical approach to extracting entity relationships. In *Proceedings of the International World Wide Web Conference (WWW)*, pages 101–110, Madrid, Spain, 2009. ACM.

[62] Feng Niu, Ce Zhang, Christopher Re, and Jude W. Shavlik. Deepdive: Web-scale knowledge-base construction using statistical learning and inference. In *Proceedings of the International Workshop on Searching and Integrating New Web Data Sources - Very Large Data Search (VLDS)*, pages 25–28, Bari, Italy, 2012. CEUR-WS.

[63] Fabian M. Suchanek, Mauro Sozio, and Gerhard Weikum. Sofie: a self-organizing framework for information extraction. In *Proceedings of the International World Wide Web Conference (WWW)*, pages 631–640, Madrid, Spain, 2009. ACM.

[64] Patrick Pantel and Marco Pennacchiotti. Espresso: Leveraging generic patterns for automatically harvesting semantic relations. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 113–120, Sydney, Australia, 2006. ACL.

[65] Ndapandula Nakashole, Martin Theobald, and Gerhard Weikum. Scalable knowledge harvesting with high precision and high recall. In *Proceedings of the ACM International Conference on Web Search and Data Mining (WSDM)*, pages 227–236, Hong Kong, China, 2011. ACM.

[66] Ndapandula Nakashole, Gerhard Weikum, and Fabian Suchanek. Patty: a taxonomy of relational patterns with semantic types. In *Proceedings of the Joint*

Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL), pages 1135–1145. ACL, 2012.

[67] Patty. http://www.mpi-inf.mpg.de/yago-naga/patty/.

[68] Karin Kipper Schuler. *VerbNet: A Broad-Coverage, Comprehensive Verb Lexicon*. PhD thesis, University of Pennsylvania, USA, 2006.

[69] Aashish Sheshadri and Matthew Lease. SQUARE: A Benchmark for Research on Computing Crowd Consensus. In *Proceedings of the AAAI Conference on Human Computation & Crowdsourcing (HCOMP)*, pages 156–164, Palm Springs, CA, USA, 2013. AAAI Press.

[70] Qiang Liu, Jian Peng, and Alexander T. Ihler. Variational inference for crowdsourcing. In *Proceedings of the Conference on Advances in Neural Information Processing Systems (NIPS)*, pages 701–709, Lake Tahoe, Nevada, USA, 2012. NIPS Foundation.

[71] Qiang Liu, Alexander T. Ihler, and Mark Steyvers. Scoring workers in crowdsourcing: How many control questions are enough? In *Proceedings of the Conference on Advances in Neural Information Processing Systems (NIPS)*, pages 1914–1922, Lake Tahoe, Nevada, USA, 2013. NIPS Foundation.

[72] Rion Snow, Brendan O'Connor, Daniel Jurafsky, and Andrew Y. Ng. Cheap and fast—but is it good?: Evaluating non-expert annotations for natural language tasks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 254–263, Honolulu, Hawaii, 2008. ACL.

[73] Bryan C. Russell, Antonio Torralba, Kevin P. Murphy, and William T. Freeman. Labelme: A database and web-based tool for image annotation. *International Journal of Computer Vision*, 77(1-3):157–173, May 2008.

[74] Carl Vondrick, Donald Patterson, and Deva Ramanan. Efficiently scaling up crowdsourced video annotation. *International Journal of Computer Vision*, 101 (1):184–204, January 2013.

[75] Thomas Steiner, Ruben Verborgh, Rik Van de Walle, Michael Hausenblas, and Joaquim Gabarró Vallés. Crowdsourcing event detection in youtube videos.

In *Proceedings of the Workshop on detection, representation, and exploitation of events in the semantic web*, pages 58–67, Berlin, Germany, 2011.

[76] Luis Von Ahn, Benjamin Maurer, Colin McMillen, David Abraham, and Manuel Blum. reCAPTCHA: Human-based character recognition via web security measures. *Science*, 321(5895):1465–1468, September 2008.

[77] Adam Marcus, Eugene Wu, Samuel Madden, and Robert C. Miller. Crowdsourced databases: Query processing with people. In *Proceedings of the Conference on Innovative Data Systems Research (CIDR)*, pages 211–214, Asilomar, California, USA, 2011. CIDR.

[78] Adam Marcus, Eugene Wu, David Karger, Samuel Madden, and Robert Miller. Human-powered sorts and joins. *Proceedings of the VLDB Endowment*, 5(1): 13–24, 2011.

[79] Sparknotes Editors. Sparknotes LLC. http://www.sparknotes.com/lit/, 2007.

[80] Edith L. M. Law, Luis von Ahn, Roger B. Dannenberg, and Mike Crawford. Tagatune: A game for music and sound annotation. In *Proceedings of the International Conference on Music Information Retrieval (ISMIR)*, pages 361–364, Vienna, Austria, 2007. Austrian Computer Society.

[81] 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 *Proceedings of the International Conference on Ontologies, Databases, and Applications of Semantics for Large Scale Information Systems*, pages 1223–1237, London, UK, 2002. Springer-Verlag.

[82] Robert Speer and Catherine Havasi. Representing general relational knowledge in conceptnet 5. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC)*, pages 3679–3686, Istanbul, Turkey, 2012. European Language Resources Association.

[83] Zhang Ce, Niu Feng, Re Christopher, and Shavlik Jude. Big data versus the crowd:looking for relationships in all the right places. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics: Volume 1 (ACL)*, pages 825–834, Jeju Island, Korea, 2012. ACL.

[84] WordNet. http://wordnet.princeton.edu, 2010.

[85] Yago. http://yago-knowledge.org.

[86] Heeyoung Lee, Yves Peirsman, Angel Chang, Nathanael Chambers, Mihai Surdeanu, and Dan Jurafsky. Stanford's multi-pass sieve coreference resolution system at the conll-2011 shared task. In *Proceedings of the Computational Natural Language Learning (CONLL): Shared Task*, pages 28–34, Portland, Oregon, 2011. ACL.

[87] Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. Generating typed dependency parses from phrase structure trees. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC)*, pages 449–454, 2006.

[88] Stanford dependency parser. http://nlp.stanford.edu/software/lex-parser.shtml.

[89] Reverb. http://reverb.cs.washington.edu.

[90] Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology (NAACL)*, pages 173–180, Edmonton, Canada, 2003. ACL.

[91] Christopher D. Manning and Hinrich Schütze. *Foundations of statistical natural language processing*. MIT Press, 2001.

[92] Conceptnet 5. http://conceptnet5.media.mit.edu, 2012.

[93] Adam Berger and John Lafferty. Information retrieval as statistical translation. In *Proceedings of the ACM SIGIR Conference on Research & Development on Information Retrieval*, pages 222–229, Berkeley, California, USA, 1999. ACM.

[94] Razvan C. Bunescu and Raymond J. Mooney. A shortest path dependency kernel for relation extraction. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing (HLT)*, pages 724–731, Vancouver, Canada, 2005. ACL.

[95] ChengXiang Zhai. *Statistical Language Models for Information Retrieval*. Morgan & Claypool Publishers, 2008.

[96] Peter F. Brown, Vincent J. Della Pietra, Stephen A. Della Pietra, and Robert L. Mercer. The mathematics of statistical machine translation: Parameter estimation. *Computational linguistics - Special issue on using large corpora*, 19: 263–311, 1993.

A APPENDIX

Entities and Entity Phrases

A.1 Extracting Entities

Following is the regular expression that HIGGINS uses to extract canonical entity names from cast sections of Wikipedia movie articles.

```
(?:\*)?(?:[\s]+)?(?:\[\[)?(.*?)(?:\]\]\s+-|\]\]\s+as|\]\]\s+stars
    \sas|\]\]\s+plays\s|\]\]\s+portrays\s|\]\]\s{0,15}\.{2,}|\]\]\
    s+|\s+-|\s+as|\s+stars\sas|\s+plays\s|\s+portrays\s|\s
    {0,15}\.{2,})((?:Mr\.|Ms\.|Mrs\.|Jr\.|Sr\.|Col\.|Cpl\.|Maj\.|
    Gen\.|Pvt\.|Sgt\.|Capt\.|Brig\.|Dr\.|[A-Z]\.|Lt\.|, Mr\.|, Ms
    \.|, Mrs\.|, Jr\.|, Sr\.|,Col\.|,Cpl\.|,Maj\.|,Gen\.|,Pvt\.|,
    Sgt\.|,Capt\.|,Brig\.|,\s[A-Z]\.|, Lt\.|[^?:\r\n\.,<])*)
```

A.2 Noun Phrase Detection

Following is the list of regex rules on part-of-speech tags (PennTreeBank tags) for extracting noun phrases from descriptions of characters in the cast sections.

1. Potentially a proper noun or common noun in either singular or plural form (e.g., “Gamekeeper”, “Death-eaters”, “Transfigurations teacher” ,“news reader’ etc.).

```
nounPhrase0 = "(NNP\\s?){1,}(NNPS\\s?){0,}(NNS\\s?){0,}(NN\\s?)
    {0,}(NNP\\s?){0,}(NNPS\\s?){0,}(NNS\\s?){0,}";
```

2. Potentially a title with a determiner (or Wh-determiner) and a proper noun in the beginning (e.g., “ex-Defense Against the Dark Arts teacher”).

```
nounPhrase1 = "((?=[^A–Z])DT\\s){0,}(JJS?\\s){0,}(NN\\s){0,}(NNP
    ?\\s?){1,}(POS\\s?)?(IN\\s?)?(\\bDT\\s?)?(JJS?\\s?)?(NNP?S?\\s
    ?){1,}(NNS?\\s?){0,}";
```

3. Potentially a title without a proper noun in the beginning (e.g., “sarcastic chairman of the East India Company”).

```
nounPhrase2 = "(JJS?\\s){0,}(NN\\s?){1,}(POS\\s)?(IN\\s)?(\\bDT\\
    s)?(JJS?\\s)?(NNS\\s?){1,}";
```

4. Potentially a title with plural common noun in the beginning (e.g., “motherly figure of Harry”).

```
nounPhrase3 = "(JJS?\\s){0,}(NNS\\s?){1,}(POS\\s)?(IN\\s)?(\\bDT
    \\s)?(JJS?\\s)?(NN\\s?){0,}";
```

5. Potentially a possession (e.g., “Harry’s Muggle Aunt”, “the Dead Man’s Chest”).

```
nounPhrase4 = "((?=[^A–Z])DT\\s){0,}(JJS?\\s){0,}(NN\\s){0,}(NNP?
    S?\\s?){1,}(CD\\s?){0,}(POS\\s){0,}(JJS?\\s){0,}(NNP?\\s?)
    {0,}(NNPS?\\s?){0,}(NNS?\\s?){0,}";
```

6. Potentially a official post (e.g., “Walter of Chatillon”).

```
nounPhraseIN = "(NNP\\s){1,}(IN\\s){1}(NNP\\s?){1,}";
```

7. Potentially an adjectival noun phrase (e.g., “scary Death-eaters”).

```
nounPhraseJJ = "(JJ\\s){1}(NNPS\\s?){1,}";
```

9. Potentially a well-known proper noun (e.g., “The Tigress”).

```
nounPhraseDT = "((?=[^A–Z])DT\\s){1}(NNP\\s){1,}";
```

To obtain longest meaningful description phrases for the character role, the text from the character description must be evaluated for matches against the above regular expressions strictly in the following order:

```
allNounPhrasesRegexList = {nounPhrase0, nounPhrase1, nounPhrase2,
     nounPhrase3, nounPhrase4, nounPhraseIN, nounPhraseJJ,
    nounPhraseDT};
```

The matched noun phrases are checked against the following types in the YAGO ontology [85].

{“PERSON”, “LIVING THING”, “IMAGINARY BEING”, “FICTIONAL”, “SOCIAL GROUP”, “FILM CHARACTERS”}.

B APPENDIX

Relations and Relational Phrases

B.1 Manually Compiled Relations for Higgins Dictionary

Tables B.1, B.2, B.3, B.4, B.5, B.6, B.7 show the list of hand-crafted relations that hold between a pair of character roles in movie or book narratives. To these relation strings, two transformations are applied to their lexical representation prior to performing a corpus search for these relations. They are

1. lemmatize the relation string (e.g. "X *have endorse* Y" for "X *had endorsed* Y").
2. replace modal verbs (e.g., could, will, would etc.), pronouns (e.g., her, his etc.), determiners (e.g., a, an, the) and adjectives with wild-card place holders that allow any members of their respective class.

accomplice of	accused	acknowledged
act on	admire	admired
admires	adopted	advisor of
agree to	agree with	allowed
also appointed	also credited	also expressed reservations about
also faces	also faces resistance from	also mocked

Table B.1: Hand-crafted Relations I

also speaks regularly with	alter ego of	ancestor of
apologized for	appointed	appreciated
approached	approach by	approved the sale of
argued with	arranged for	asked
assailed	auctioned	auditioned for
aunt of	avoid	awaited
baby of	is responsible for	was aware of
was an outspoken critic of	was an aide to	was the author of
is a daughter of	was the father of	was the head of
is the heroine of	is the image of	was the inspiration for
is a stooge of	is the wife of	was accused by
was addicted to	was adored by	was aligned with
will accept	would actually provide more light than	could be facing
would be rewarded for	should be separated from	would defeat
will face	would force	would give up
would have preferred	may have to take	would help honour
would inherit	would miss	would never begin a program with
will never forget	would not be taped by	will not block
could not comment on	would not prevent	would now assent to
will oppose	would pay	will perform along with
would probably continue along	would probably miss	would pummel
would reconsider	would replace	would sign
would soon be meeting with	may well be a pawn of	accompanied
is always overshadowed by	was appearing before	were asked to inform
is backed by	is being held captive by	is beside
was chosen by	is close to	is coming under

TABLE B.2: Hand-crafted Relations II

be concern about	was cooperating with	is currently working with
was endorsing	was endorsed by	is expected to take care of
was forced to endure	was going with	was haunted by
be host of	is interested in	is investigating
was involved with	was killed as	is like
was made aware of	was made to watch	was monitoring
was named	was not aware of	is not encumbered by
is not too fond of	is now working for	be oust by
is paying for	is press secretary to	was probably
was raising money for	was reassigning	was released yesterday by
was replaced by	was running toward	was seeking a stay of
was slated to become	is still below	was striving for
is struggling to hang on to	is survived by	was taking his cue from
was telling	was tired of	is to be followed by
was to denounce	is to marry	was wooed by
beat	became	became chairman after
befriended	befriends	began writing about
binds	blocked	boss of
boy of	boyfriend of	broke
broke with	bride of	bridegroom of
briefed	brought	brother of
brother-in-law of	called	called for
called on	campaigned for	cares more about
care-taker of	carried	challenged
chased	cheat	cheated
child of	chose	colleague of
came a day after	come across	came as
comes to	comes to finding	come with
controlled	converts into	cousin of
created	dance with	dance along with
daughter of	daughter-in-law of	declined to comment on

TABLE B.3: Hand-crafted Relations III

defended	deftly finds	delivered a new plan for
descendant of	destroyed	developed
devoted his energies to	dislike	dislike
dismissed	did not appear to understand	does not comment on
did not deny	did not fully understand	did n't have anything to do with
did not know	did not require	did not respond to
do not try to replicate	doctor of	drove
dropped	elder brother of	elder sister of
eliminated	emulate	encountered
enemy of	engaged	ex-boyfriend of
ex-girlfriend of	ex-spouse of	expected
expects to meet with	expects to retain	fails to deliver
fell behind	family member of	family of
fatally shot	father of	father-in-law of
feels terribly let down by	fielded complaints from	fights with
fought with	filed her first report about	found
finds out about	first met	followed
followed even	followed with	fooled
fools	foster-brother of	foster-child of
foster-father of	foster-mother of	foster-sister of
frees	friend of	generally endorsed
got to	girl of	girlfriend of
gave out	went to	grabbed
grandchild of	granddaughter of	grandfather of
grandmother of	grandson of	great-granddaughter of
great-grandfather of	great-grandmother of	great-grandson of
greeted	grilled	grows up to become
handled	hated	hates
has little chance of defending	has direct control over	had a high regard for

TABLE B.4: Hand-crafted Relations IV

had an argument with	had no comment on	have a connection to
had no memory of	has accepted	has accompanied
has accused	has acquired	had agreed to join
has also retained	had answered	had appointed
had appointed to oversee	has attacked	has attacked the credibility of
has baffled	has been buffeted by	has been dragging
had been driving	had been falsely accused by	has been linked to
has been pursuing	had been talking with	had been touting
has been under	has changed	has endorsed
had found no evidence of	had got to know	had given the book to
have idea about	had meetings with	had not expected
have not record the name of	has not spoken to	had not talked to
had nothing to do with	has pressed	had reached an agreement with
had recommended	has revived	had stayed with
had strongly criticized	had taken	had taught
had testified for	has threatened to topple	had to battle
had to deal with	had troubled	had truly addressed the needs of
have underestimate	had used	has used his knowledge of
has wasted little time getting to	has yet to tame	helps
helped create	helped found	hints at
hired	husband of	immediately accused
immediately telephoned	included	infant of
influence	informs	initially refused to support
inquired about	interviewed	invented
join	killed	knew
lad of	lady of	later dropped
lauded	laugh at	lawyer of

TABLE B.5: Hand-crafted Relations V

leads	left	left behind
like	liked	likes to encourage
liquidate	listened to	lobbied for
look for	lose touch with	loved
lover of	maid of	made his name as
marry	master of	matched
match maker of	met as	met with
menial of	missed	mistress of
mother of	mother-in-law of	moved
needed	needs to get	neighbour of
nephew of	never took	niece of
nominate	offend	offend
offered	offer to release	often heard
once was followed home by	once shared the limelight with	oppose
ordered	originally represented	orphan of
outperform	oversees	overwhelmingly supported
parent of	partner of	performed better than
periodically consult with	personally opposes	pick up
pitched	planned to call	plans to nominate
plays with	point to	praise
praise	prepared for	promised
proposes	protects	purchased
pushed	quoted	raised
rallied around	rarely met with	reach before
reacted angrily to	read a statement on	recalls
received	recreates	recruited
refuses to take	refused to travel to	relative of
released	relied heavily on	replaced
reports to	requested	reshaped
respected	returned to	returned to tell
ridicule	rose less than	rob

TABLE B.6: Hand-crafted Relations VI

rob	sanctioned	saved
saves	seconded	seduced
seduces	sent	sent for
servant of	serve	shook
shot	showed	showed evidence of
sided with	side-swiped	sang with
sings with	sister of	sister-in-law of
slammed	somehow mitigates against	son of
son-in-law of	spoke on	spoke with
spit on	spotted	stabbed
stalked	stalks	step-brother of
step-daughter of	step-father of	step-mother of
step-sister of	stuck close to	stifle
struck down	suggest	supervises
supports	sweet heart of	sympathized with
took	took control from	take everything from
took issue with	took on	take power from
took time out from	teaches	teacher of
teammate of	testified about	testified for
then devised a scheme for	then ranted on	think about
thought of	threatened to sue	threw
torture	tortured	traces
traded	trails	trainer of
tried to calm	tried to persuade	twin of
uncle of	used	use to refer to
used to sleep with	voted for	wants
wants to do as	wants to finish	wanted to use
warmly shook the hands of	waved	whispers to
widow of	widower of	wife of
win over	wound	works closely with
work for	writes to	younger brother of
younger sister of		

TABLE B.7: Hand-crafted Relations VII

B.2 Collocation Measures for Phrase Relatedness

- **Pointwise Mutual Information (PMI)**

  PMI, a collocation metric defined over a pair of words, measures the reduction of uncertainty about the occurrence of one word when there is knowledge of occurrence of the other word. We adapted this metric to the co-occurrence of a dictionary relation r with a noun-phrase pair n, and co-occurrence of a phrase p with n. Let N is the set of all noun-phrase pairs. For a dictionary relation r and a mined phrase p, PMI is evaluated as $PMI(p,r) = \log \frac{\Pr(p,r)}{\Pr(p)*\Pr(r)}$. If $N(r)$ is the set of noun-phrase pairs co-occurring with r and $N(p)$ is the set of noun-phrase pairs co-occurring with p, then $PMI(p,r)$ is estimated as,

  $$PMI(p,r) = \log \frac{|N(p) \cap N(r)| * |N(p) \cup N(r)|}{|N(p)| * |N(r)|}.$$

- **Normalized PMI**

  PMI has the undesirable property that its value for a pair of perfectly correlated events is *higher* when their combination is *less* frequent. As noisy phrases which often have low frequencies are boosted in such cases, it is desirable to establish a fixed upper bound for PMI. Therefore to reduce this effect of high PMI values for lower frequencies, it can be normalized as

  $$nPMI(p,r) = \frac{\log(\Pr(p,r)/\Pr(p) * \Pr(r))}{-\log \Pr(p,r)}.$$

- **Cosine of PMI**

  In PMI and nPMI we considered co-occurrence of r and p with the set of noun-phrase pairs in the corpus. However this approach ignores the co-occurrence frequencies of the individual noun-phrase pairs in N with r or p. Therefore we experimented with comparing PMI values of r with the noun-phrase pairs in N and the PMI values of p with the noun-phrase pairs in N using cosine similarity as,

$$cosinePMI(p,r) = \frac{\sum\limits_{e \in E} PMI(r,e) * PMI(p,e)}{\sqrt{\sum\limits_{e \in E} (PMI(r,e))^2} * \sqrt{\sum\limits_{einE} (PMI(p,e))^2}}.$$

Although cosinePMI performed better than PMI and nPMI, its scores are higher for relations that produce large number of phrases through co-occurring noun-phrase pairs. As the number of phrases for each relation is highly dependent on the nature of the corpus, cosinePMI did not yield us the desired robustness.

- **Mutual Information (MI)**

  MI is a information theoretic measure defined over two random variables (unlike PMI which holds for individual values of random variables). For two random variables X and Y, it is given as

$$MI(X,Y) = \sum_{x \in X, y \in Y} \Pr(x,y) * \log \frac{\Pr(x,y)}{\Pr(x) * \Pr(y)}.$$

  In our setting, we define X_{ce} and X_{re}, two binary random variables indicating co-occurrence of e with pattern c and relation r with a noun-phrase pair $e \in N$. $MI(c,r)$ can be calculated as,

$$MI(c,r) = \sum_{e \in E} \sum_{X_{ce}=0,1} \sum_{X_{re}=0,1} p(X_{ce}, X_{re}) \log \frac{p(X_{ce}, X_{re})}{p(X_{ce}) p(X_{re})}$$

  where the probabilities $p(X_{ce})$ and $p(X_{re})$ are estimated as follows:

$$\begin{aligned} p(X_{ce} = i) &= \frac{1}{|E|} \sum_{e \in E} (X_{ce} = i) \\ p(X_{re} = i) &= \frac{1}{|E|} \sum_{e \in E} (X_{re} = i) \\ p(X_{ce} = i, X_{re} = i) &= \frac{1}{|E|} \sum_{e \in E} (X_{ce} = i, X_{re} = i) \end{aligned}$$

  where i can take values in $\{0, 1\}$.

- **Pearson's Chi-square Test**

  Pearson's chi-square test is a statistical test between two variables that compares observed frequencies with the frequencies expected under their independence. For large differences in the observed and expected frequencies, the null hypothesis of independence of the two variables can be rejected. We employed this test to determine if higher number of common co-occurring noun-phrase pairs between a relation and a phrase imply dependence among them. For a (r, p) pair, we create the following 2×2 table, consisting of co-occurrence frequencies of noun-phrase pairs,

	r	$!r$
p	$O_{11} = \|N_r \cap N_p\|$	$O_{12} = \|N_p - N_r\|$
$!p$	$O_{21} = \|N_r - N_p\|$	$O_{22} = \|N_p \cup N_r\|$

where N_r and N_p are noun-phrase pairs occurring with r and p respectively.

$$\chi^2 = \frac{(O_{11}O_{22} - O_{12}O_{21})^2}{(O_{11} + O_{12})(O_{11} + O_{21})(O_{12} + O_{22})(O_{21} + O_{22})}$$

We tested the χ^2 values against probability level of $\alpha = 0.05$ in the χ^2 distribution. While χ^2 test works for large values of co-occurrence frequencies, for small cell values, the test is inaccurate (which is often the case with a large majority of our corpus phrases).

List of Figures

List of Tables