UNIVERSITÄT DES SAARLANDES

DOCTORAL THESIS

Methods and Tools for Summarization of Entities and Facts in Knowledge Bases

Tomasz Tylenda

Dissertation
zur Erlangung des Grades
des Doktors der Ingenieurwissenschaften
der Naturwissenschaftlich-Technischen Fakultäten
der Universität des Saarlandes

Saarbrücken, August 2015

Promotionskolloquium

Datum	28. September 2015
Ort	Saarbrücken
Dekan der Naturwissenschaftlich - Technischen Fakultät I	Prof. Dr. Markus Bläser

Prüfungskommission

Vorsitzender	Prof. Dr. Gert Smolka
Gutachter	Prof. Dr.-Ing. Gerhard Weikum
Gutachter	Prof. Dr.-Ing. Klaus Berberich
Beisitzer	Dr. Ida Mele

Abstract

Knowledge bases have become key assets for search and analytics over large document corpora. They are used in applications ranging from highly specialized tasks in bioinformatics to general purpose search engines. The large amount of structured knowledge they contain calls for effective summarization and ranking methods.

The goal of this dissertation is to develop methods for automatic summarization of entities in knowledge bases, which also involves augmenting them with information about the importance of particular facts on entities of interest. We make two main contributions.

First, we develop a method to generate a summary of information about an entity using the type information contained in a knowledge base. We call such a summary a semantic snippet. Our method relies on having importance information about types, which is external to the knowledge base. We show that such information can be obtained using human computing methods, such as Amazon Mechanical Turk, or extracted from the edit history of encyclopedic articles in Wikipedia.

Our second contribution is linking facts to their occurrences in supplementary documents. Information retrieval on text uses the frequency of terms in a document to judge their importance. Such an approach, while natural, is difficult for facts extracted from text. This is because information extraction is only concerned with finding *any* occurrence of a fact. To overcome this limitation we propose linking known facts with *all their* occurrences in a process we call fact spotting. We develop two solutions to this problem and evaluate them on a real world corpus of biographical documents.

Kurzfassung

Wissensbasen zählen zu den wichtigsten Bausteinen für die Suche und Analyse in großen Dokumentkorpora. Sie werden sowohl für hoch spezialisierte Aufgaben der Bioinformatik als auch in Suchmaschinen verwendet. Die große Menge an strukturiertem Wissen, die sie enthalten, fordert effektive Methoden des Zusammenfassens und Ordnens.

Das Ziel dieser Arbeit ist es, Methoden für die automatische Zusammenfassung von Entitäten in Wissensbasen zu entwickeln; dies beinhaltet auch die Bestimmung wichtiger Fakten einer Entität. Dazu leistet diese Arbeit zwei Beiträge.

Erstens entwickeln wir ein Verfahren zur Zusammenfassung der Informationen über eine Entität unter Verwendung der Typinformationen, die in Wissensbasen zur Verfügung stehen. Wir nennen eine solche Zusammenfassung ein *Semantic Snippet*. Unser Verfahren benötigt hierfür zusätzliche externe Informationen über die Wichtigkeit von Typen. Wir zeigen, dass solche Informationen durch Methoden des Human Computing, zum Beispiel mit Hilfe von Amazon Mechanical Turk, oder aus der Evolution enzyklopädischer Artikel in Wikipedia gewonnen werden können.

Der zweite Beitrag der Arbeit ist eine Methode zur Verknüpfung von Fakten mit ihren Vorkommen in ergänzenden Dokumenten. Bei der Informationsgewinnung aus Texten wird die Häufigkeit der Wörter in einem Dokument verwendet, um ihre Wichtigkeit zu beurteilen. Ein solcher Ansatz erscheint natürlich, ist aber nicht ohne weiteres möglich für den Fall von aus Text extrahierten Fakten. Dies liegt daran, dass die Informationsextraktion auf die Suche nach *einem* Vorkommen eines Fakts fokussiert ist. Um dieser Einschränkung entgegenzuwirken, schlagen wir einen Prozess vor, der bekannte Fakten mit *all ihren* Vorkommen verknüpft. Diesen Prozess nennen wir *fact spotting*. Zwei Methoden für diesen Ansatz werden in der Arbeit entwickelt und auf einem Korpus von biographischen Dokumenten evaluiert.

Summary

Knowledge bases have become key assets in information retrieval, natural language processing, question answering, and many other areas that require processing information at the level of facts, entities, and concepts. Recent progress in information extraction led to very large knowledge bases that often make it possible to satisfy a user's information need without going to any source documents. This poses a challenge if the knowledge base contains both interesting and important information as well as trivial and insignificant, which often is the case. What is needed are methods to decide which information is crucial and which is secondary if not irrelevant. This dissertation contributes to effective summarization and ranking in knowledge bases for two aspects:

- **Semantic Snippets.** The first problem that we study is summarizing the available information about a given entity. Knowledge bases contain many facts about entities. Although a person can quickly browse a set of facts about a single entity, this becomes impossible if we are given more of them. This problem arises often in the context of semantic search engines: given a query they tend to either provide just identifiers of the entities, which may not be enough for the user, or overwhelm him or her with all facts they know about each entity. What is needed are snippets that are in the middle ground, that is, provide crucial information, but avoid overloading the user with all known facts.

  Particularly useful information about entities is contained in the type system of the knowledge base. For example, YAGO knows 38 types of Albert Einstein; in case of Freebase every fact, such as David_Beckham –/SPORTS/PRO_ATHLETE/SPORTS_PLAYED_PROFESSIONALLY→ Football, implies that the entity has a certain type, such as Athlete. Moreover any fact, e.g. (...) –WON→ Nobel_Prize, can be can be viewed as a type, e.g. Nobel_Laureate. We therefore base our snippets on types.

  We propose a set of features that a good, type-based summary should have. The summary should be concise, so that it can be read quickly or displayed in limited space; types that are included should denote important properties of the entity, and not trivial or exotic information; the selected types should have the right granularity, that is, cannot be

too general or very specific to a small set of entities; finally, the summary should consist of a diverse set of types.

We develop two methods for obtaining a type-based summary. The first views the types of an entity as a directed acyclic graph with the types as nodes which are connected by the subClassOf relation. The semantics of the subClassOf relation makes the graph transitively closed. Since any type implies all of its supertypes, the summary forms an independent set of nodes in such a graph; that is, no two nodes of the summary are connected by a directed edge. In order to select a summary we assign costs and weights to nodes and select an independent set of vertices with the highest weight subject to a cost constraint. Our second summarization methods views types as sets of entities that they contain. In order to make the summary diverse, we prefer families of types with small intersection. At the same time we also prefer large types (with many entities) to avoid exotic types that do not carry important meaning.

Our summarization methods also use information external to the knowledge base. We do this in order to avoid selecting types which are meaningless or exotic. We show different methods of assessing the usefulness of a type for summarization. Due to their small number, types high in the hierarchy of the subClassOf relation can be effectively judged by crowdsourcing the problem to a human computing platform such as Amazon Mechanical Turk. For the leaf types we can utilize the edit history of Wikipedia articles. We assume that the types which are most important with respect to an entity are added as Wikicategories of its articles before the less important ones.

Our summarization algorithms were implemented in our YAGO ontology browser. The work was published as a demonstration paper [130] in the World Wide Web Conference.

- **Fact Spotting.** The second problem studied in this thesis is finding facts known to a knowledge base in new documents. Given a set F of facts about an entity and a document d, we want to find a set of facts $F' \subseteq F$ that occur in d. We will refer to this problem, which can be thought of as the reverse of information extraction, as *fact spotting*.

  Fact spotting can be used to gather statistics about the frequency of fact occurrences in a corpus of documents. This finds application in problems where we need to know how important different facts are. For example, we can treat fact frequency as a proxy for importance, similarly to how frequency statistics is used in information retrieval. Using fact spotting to mine importance of facts links this problem to our semantic snippets, which required as an input similar information on the importance of types.

  Fact spotting is useful for more applications than statistics gathering. Provenance information in knowledge bases is limited to documents from which the facts can be extracted. If we are able to augment knowledge bases with fact occurrences in supplementary documents, we can improve applications that rely on fact provenance, including automatic truthfulness assessment and semantic search engines that operate on facts.

  We develop two methods of fact spotting. The first uses a high quality relational paraphrase dictionary to find fact occurrences with high preci-

sion. To improve recall we then speculatively output facts for which we have partial evidence of presence and whose related facts were found in the document. Our second fact spotting method turns this idea around. First, we aggressively find a large set of candidate facts, and then prune them based on consistency constraints. Our constraints include natural rules on how named entities are disambiguated and a special constraint that exploits relations between knowledge base facts. For instance, if a document mentions the year of an event (e.g. actor winning an Academy Award), then it will also provide information about the event (e.g. movie, category). Such relations between facts can often be obtained from the knowledge base structure, for example, from compound facts in Freebase or the SPOTLX fact representation in YAGO 2.

Our evaluation on biographical documents that were manually annotated with the ground truth facts shows that fact spotting can achieve both high precision and high recall. The work on fact spotting was published in refereed workshop papers [131, 129].

Zusammenfassung

Wissensbasen zählen zu den wichtigsten Bausteinen für Information Retrieval, automatische Sprachverarbeitung, automatisches Beantworten von Fragen, sowie viele andere Bereichen, die die Verarbeitung von Informationen auf der Ebene der Fakten, Entitäten und Konzepte erfordern. Wissensbasen ermöglichen es, dem Informationsbedarf des Nutzers nachzukommen ohne auf Quelldokumente zurückgreifen zu müssen. Die dadurch entstehende Herausforderung ist, dass Wissensbasen oft sowohl interessante und wichtige Informationen als auch triviale und unbedeutende Fakten enthalten. Was wir brauchen, sind Methoden, um zu entscheiden, welche Informationen von entscheidender Bedeutung sind und welche weggelassen werden können. Diese Dissertation trägt zur effektiven Zusammenfassung und zum Ordnen in Wissensbasen neue Methoden zu zwei Aspekten bei:

- **Semantic Snippets.** Das erste Problem, mit dem sich diese Arbeit befasst, ist das Zusammenfassen verfügbarer Informationen über eine Entität. Wissensbasen enthalten viele Fakten über Entitäten. Ein Nutzer kann schnell durch einen Satz von Fakten über eine Entität navigieren, jedoch wird diese Aufgabe unmöglich, wenn mehrere Entitäten auf einmal betrachtet werden. Dieses Problem tritt häufig in Zusammenhang mit semantischen Suchmaschinen auf. Diese neigen dazu für eine Anfrage entweder nur die Kennungen der Entitäten zu liefern, was oft nicht ausreichend für den Nutzer ist, oder überschwemmen den Nutzer mit allen Fakten, die sie über jede Entität kennen. Was man braucht, sind Snippets, die einen Mittelweg darstellen. Sie liefern wichtige Informationen, ohne dabei den Benutzer mit bekannten Fakten zu überlasten.

  Das Typsystem von Wissensbasen erhält eine besonders nützliche Art von Informationen über Entitäten. YAGO kennt zum Beispiel 38 Typen von Albert Einstein; im Fall von Freebase beinhaltet jeder Fakt, wie beispielsweise David_Beckham –/SPORTS/PRO_ATHLETE/SPORTS_PLAYED_PROFESSIONALLY→ Football, einen bestimmten Typ, wie Athlete. Jeder Fakt, z. B. (...) –WON→ Nobel_Prize, kann zudem als Typ betrachtet werden, z.B. Nobel_Laureate. Deshalb bilden Typen die Basis unserer Snippets.

  Wir schlagen Eigenschaften vor, die eine gute typbasierte Zusammenfassung haben soll. Die Zusammenfassung sollte kurz sein, so dass sie schnell

gelesen werden kann oder in begrenztem Raum angezeigt werden kann; Typen in der Zusammenfassung sollen wichtige Eigenschaften der Entität beschreiben, nicht aber triviale oder exotische Informationen; Typen sollten die richtige Granularität haben, also nicht zu allgemein oder zu spezifisch sein; zudem soll eine Zusammenfassung aus einer vielfältigen Menge von Typen bestehen.

Wir entwickeln zwei Verfahren zum Erzeugen einer typbasierten Zusammenfassung. Beim ersten werden die Typen einer Entität als gerichteter azyklischer Graph betrachtet. Die Typen sind hierbei die Knoten, die durch die subClassOf Beziehung verbunden sind. Die Semantik der subClassOf Beziehung macht den Graph transitiv geschlossen. Da jeder Typ alle seine Supertypen impliziert, bildet die Zusammenfassung eine unabhängige Menge von Knoten in einem solchen Graph. Das heißt, keine zwei Knoten der Zusammenfassung sind durch eine gerichtete Kante verbunden. Um eine Zusammenfassung auszusuchen, weisen wir den Knoten Kosten und Gewichte zu und wählen eine Knotenmenge mit höchstem Gewicht vorbehaltlich der Kosteneinschränkung. Unser zweites Zusammenfassungverfahren betrachtet Typen als Mengen von Entitäten, die sie enthalten. Um die Vielfalt der Zusammenfassung sicherzustellen, bevorzugen wir die Mengen der Typen mit kleiner Schnittmenge. Gleichzeitig ziehen wir auch große Mengen mit vielen Entitäten vor, um exotische Typen, die keine wichtige Bedeutung tragen, zu vermeiden.

Unsere Zusammenfassungsverfahren nutzen auch Informationen, die außerhalb der Wissensbasis liegen. Wir tun dies, um die Ausgabe von bedeutungslosen oder exotischen Typen zu vermeiden. Wir präsentieren verschiedene Methoden zur Beurteilung der Wichtigkeit eines Typs für die Zusammenfassung. Die Typen weit oben in der Hierarchie der subClassOf-Beziehung können, aufgrund ihrer geringen Anzahl, durch Crowdsourcing des Problems mittels einer Human Computing Platform, wie zum Beispiel Amazon Mechanical Turk, beurteilt werden. Für Typen unten in der Hierarchie, die Blätter des Graphen, können wir die Editierhistorie von Wikipedia nutzen. Wir gehen davon aus, dass die wichtigsten Typen einer Entität als Wikipedia-Kategorien des jeweiligen Artikels der Entität vor den weniger wichtigen aufgenommen werden.

Unsere Zusammenfassungverfahren wurden im YAGO Ontologie Browser implementiert. Die Arbeit wurde als Demonstration [130] in der World Wide Web Conference veröffentlicht.

- **Fact Spotting.** Das zweite in dieser Dissertation untersuchte Problem ist die Suche nach der Wissensbasis bekannten Fakten in neuen Dokumenten. Gegeben eine Menge F von Fakten über eine Entität und ein Dokument d, wollen wir eine Menge $F' \subseteq F$ von Fakten, die in d auftreten, finden. Wir nennen dieses Problem, das als Umkehrung der Informationsgewinnung betrachtet werden kann, Fact Spotting (Faktenerkennung).

  Fact Spotting kann für das Sammeln von Statistiken über die Häufigkeit von Fakten in einem Korpus von Dokumenten verwendet werden. Dies findet Anwendung bei Problemen, die Informationen über die Wichtigkeit verschiedener Fakten verlangen. Beispielsweise können wir die Häufigkeit der Fakten als Maß für die Wichtigkeit betrachten, ähnlich wie

Häufigkeitsstatistiken im Information Retrieval verwendet werden. Der Gebrauch des Fact Spotting zur Entdeckung der Wichtigkeit von Fakten verbindet dieses Problem mit unseren Semantic Snippets, die als Eingabe solche Informationen über die Wichtigkeit von Typen benötigen.

Fact Spotting hat weitere nützliche Anwendungen. Herkunftsinformationen in Wissensbasen sind beschränkt auf Dokumente, aus denen die Fakten ursprünglich extrahiert wurden. Sind wir in der Lage, Wissensbasen mit den Faktenvorkommen in zusätzlichen Dokumenten zu ergänzen, können wir Anwendungen, die sich auf die Herkunft der Fakten verlassen, verbessern, insbesondere automatische Wahrhaftigkeitsbewertung sowie semantische Suchmaschinen, die auf Fakten arbeiten.

Wir entwickeln zwei Methoden zum Fact Spotting. Die erste verwendet ein hochwertiges Wörterbuch von relationalen Umschreibungen, um Faktvorkommen mit hoher Präzision zu finden. Um die Trefferquote zu verbessern, werden vermutliche Fakten ausgegeben, wenn unvollständige Hinweise auf deren Vorkommen existieren und verwandte Fakten im Dokument gefunden wurden. Unsere zweite Fact Spotting-Methode kehrt diese Idee um: zuerst werden viele mögliche Fakten gefunden und dann, basierend auf Konsistenzbedingungen, reduziert. Unsere Konsistenzbedingungen umfassen Regeln für die Disambiguierung von Entitäten und für Abhängigkeiten zwischen Wissensbasisfakten. Wenn beispielsweise ein Dokument das Jahr eines Ereignisses erwähnt (z.B. Schauspieler gewinnt einen Oscar), dann wird es auch Informationen über das Ereignis selbst (z.B. Film, Kategorie) erwähnen. Eine solche Beziehung zwischen Fakten kann oft von der Wissensbasis extrahiert werden, beispielsweise aus der Verbindung zwischen Fakten in Freebase oder der SPOTLX Struktur in YAGO 2.

Unsere Auswertung auf einem Korpus von biographischen Dokumenten mit manuell annotierten Fakten zeigt, dass Fact Spotting sowohl eine hohe Genauigkeit als auch eine hohe Trefferquote erreichen kann. Die Arbeit über Fact Spotting wurde in begutachteten Workshop-Artikeln [131, 129] veröffentlicht.

Acknowledgments

First and foremost, I would like to express my deepest gratitude to my advisor, Prof. Gerhard Weikum, for giving me the opportunity to pursue research under his guidance. This thesis would not be possible without his support and scientific advice.

I am thankful to all members of the Databases and Information Systems Department for creating a great working environment. They supported me both as scientists and as friends. I am thankful to people with whom I had the pleasure to collaborate, in particular Mauro Sozio, Arturas Mazeika, and Sarath Kumar Kondreddi. I also thank Saskia Metzler for helping me translate the abstract and summary of this thesis into German.

Last but not least, I would like to thank my parents, Jacek and Janina Tylenda, who always supported me.

Contents

CHAPTER 1

Introduction

1.1 Motivation

We live in the age of abundance of data. The size of the World Wide Web is immense: already in July 2008 Google reported that its search engine found more than 1 trillion unique URLs [5]. In addition to that, we are also gaining access to huge repositories of scientific publications, newspaper archives, and databases covering topics from protein interactions to movies. This impressive amount of data can be useful only if we are able to find knowledge that we are looking for. The first step in this direction was made by search engines operating on text. While highly useful, they only operate on surface representation and cannot genuinely understand the documents. In order to achieve further progress we need to equip computers with knowledge, so that they can link pieces of information scattered over multiple sources.

Initial efforts in building knowledge bases (also called knowledge graphs or ontologies) were manual. The field of artificial intelligence developed Cyc [78] by manually encoding facts and rules about the world. Another important example of a manually created knowledge base is WordNet [48], which contains linguistic information about English words and their senses. While such manually created knowledge bases can be large and highly useful, progress in information extraction (IE) allowed automatic creation of even larger knowledge bases, such as YAGO [118], DBpedia [8], or Freebase [19]. Knowledge bases are increasingly used by major search engines: Google's Knowledge Graph and Bing's Satori allow the search engines to answer many information needs directly, without having to read source documents.

With the wealth of data comes the old problem of navigating it. Therefore, our first aim in this thesis is to develop a method for *summarizing* structured information from knowledge bases. A big challenge in summarization is deciding which information is important. The connections in a knowledge base alone cannot tell us what is noticeable and salient to an entity and what is not. We therefore require external sources of informativeness of types and facts. For types such a resource could be compiled manually with Amazon Mechanical Turk or obtained automatically from their occurrence in a text corpus or the edit history of Wikipedia articles.

To obtain the same information for facts, we can use statistics about their occurrences in a document corpus. This poses a challenge, however, because the only occurrences of facts that we know are those which were found by information extraction systems. This is limited to textual locations that are easy to extract, discarding all those occurrences that are redundant or too difficult for IE due to their complexity. To bridge this gap we propose the problem of *fact spotting* which is linking known facts to their occurrences in new documents. The task has many application beyond statistics gathering; they include KB curation, truthfulness assessment, and document analytics at the semantic level.

1.2 Approach

Summarization of information in knowledge bases is particularly important in semantic search engines, which provide users with a list of entities satisfying a query. Such a list should be accompanied by short descriptions of each entity. It is undesirable that the search engine only provides names of the entities or overwhelms the user with all known facts. What we need instead are short descriptions of entities, which we call *semantic snippets*. In order to generate a snippet we must select a set of important facts about an entity. The kind of information known to the knowledge base that lends itself particularly well to generating snippets are entity types. Our approach is based on selecting a small set of types of an entity. To this end we identify desirable properties of a snippet and the types it should consist of. We propose two methods of generating semantic snippets: one uses the SUBCLASSOF relation on types to avoid redundancy; the other views types as sets of entities they contain and uses an intersection based algorithm to find the best snippet. We also show how informativeness of types can be obtained from resources external to the knowledge base.

To solve the problem of fact spotting we propose two algorithms that identify occurrences of known facts. The input to our system is a document, a selected entity, and a set of facts about it. We use dictionaries of entity aliases and paraphrases of relations to aggressively identify occurrences of the facts, aiming for both high precision and high recall. Our methods make use of the dependencies between facts to speculatively find facts in the presence of incomplete evidence, and to prune incorrect matches if inconsistencies are found. We evaluate our approaches on a set of biographical documents of varying style and length. Our experiments include the application of fact spotting to knowledge base curation, comparison to an information extraction system OLLIE, and evaluating how well it performs in the task of named entity disambiguation, which is a subproblem of fact spotting.

1.3 Thesis Contribution

The main contributions of the thesis are:

- **Semantic Snippets.** The first contribution of the thesis is the method for summarizing information about entities using their types. We recognize the properties that a good summary should have and develop algorithms

for the generation of such summaries. Our summarization methods were incorporated in an Ontology Browser and published as a demo paper:

- Tomasz Tylenda, Mauro Sozio, Gerhard Weikum. *Einstein: physicist or vegetarian? summarizing semantic type graphs for knowledge discovery.* 20th International Conference on World Wide Web (WWW). 2011.

- **Fact Spotting.** The second contribution of the thesis is a method for linking facts in a knowledge base to textual documents — the problem which we call *fact spotting*. Knowledge bases can store information about provenance of facts, which describes how and from where the facts were extracted. Our method augments it with fact occurrences in supplementary documents, which were not considered in the process of building the knowledge base. Such links are useful for many applications which include statistics gathering, truthfulness analysis, or text analytics at fact level. Our work on fact spotting has been published in the following workshop papers:

  - Tomasz Tylenda, Yafang Wang, Gerhard Weikum. *Spotting Facts in the Wild.* Workshop on Automatic Creation and Curation of Knowledge Bases (WACCK). 2014.
  - Tomasz Tylenda, Sarath Kumar Kondreddi, Gerhard Weikum. *Spotting Knowledge Base Facts in Web Texts.* Workshop on Automatic Construction of Knowledge Bases (AKBC). 2014.

1.4 Thesis Outline

The remainder of this thesis is organized as follows: Chapter 2 describes the related work, including knowledge bases, information extraction systems, truth finding, semantic search, summarization, and named entity disambiguation. Chapter 3 describes the summarization problem for knowledge bases. It presents the summarization task, discusses desirable properties of semantic summaries, and introduces two summarization methods. The structure of the knowledge base alone cannot tell us which properties of entities are salient and should be included in the summary. Therefore in Chapter 4 we describe Mechanical Turk experiments on informativeness of types, and discuss how the Wikipedia edit history can be used to mine the importance of categories. Chapter 5 describes the work on fact spotting, which is the reverse of information extraction: given a document and a set of facts we determine whether the facts are mentioned in the document. Chapter 6 concludes the thesis and presents possible directions for future research.

CHAPTER 2

Related Work

In this chapter we review the works which form the background for this thesis or are related to our contributions. We start with a discussion of knowledge bases (KBs). Many of them are constructed automatically, which makes them inherently connected with information extraction systems (IE), which are also discussed here. Large scale knowledge bases need effective systems for summarization and exploration, which we discuss as well. The problem of sumarization has been studied in different contexts, including information retrieval, visualization, knowledge management, etc. We also present problems related to our fact spotting task: named entity disambiguation, which can be solved jointly with fact spotting; question answering, which can be solved using similar methodology; and truth finding and knowledge base curation which are downstream applications that can benefit from fact spotting.

2.1 Knowledge Bases

Knowledge bases appeared in the fields of artificial intelligence and computational linguistics. The seminal work in the field is Cyc [78]. Cyc is a system that tries to codify common sense knowledge to provide a critical mass of knowledge needed for natural language processing and artificial intelligence applications. It was created by manually encoding assertions in a logical language. One of the aims of the project was to allow automatic harvesting of additional knowledge. A system that automatically learns new facts was demonstrated in [89]. It generates search engine queries to find values of unknown attributes of entities, extracts them, and finally verifies whether they are true and consistent with already known facts.

WordNet [48, 97] is a widely used lexical database of the English language, which can be thought of as a machine readable dictionary. It connects words with their meanings called *synsets*. WordNet knows about synonymy (multiple words have the same meaning) and polysemy (one word has more than one meaning). It contains morphological data, for example, derivational and inflectional forms of words. Synsets are connected by links indicating hypernymy (super-concept, e.g. *maple* and *tree*), meronymy (part of), etc. The links between sub- and super- concepts form a taxonomy, which was used as the foundation of the type system in YAGO [118, 119].

Manually created knowledge bases appeared also in biomedical applications. For instance, the Gene Ontology [7] was created to unify the knowledge about genomes of eukaryotes, that has been rapidly growing in recent years. Unified Medical Language System (UMLS) [18] is a database of terminology created to enhance access to biomedical literature, for example, by listing all synonymous names of proteins.

YAGO. YAGO [118, 119] is an automatically created knowledge base. The logical model of YAGO is based on RDFS. Facts are subject-predicate-object triples associated with fact identifiers. Reification allows expressing n-ary facts and general properties of relations. YAGO extracts facts from semi-structured parts of Wikipedia: categories and infoboxes. Although Wikipedia categories form a directed acyclic graph, not all of them are proper types and not all links correspond to subclass relationship. Categories describe concepts (e.g. physicists), relations (e.g. people born in Ulm), topics (e.g. physics), and are also used for administrative purposes (e.g. articles needing cleanup). YAGO uses the first kind to identify which pages refer to entities; they are recognized using a heuristics that conceptual categories appear in plural. Categories which describe relations are used to extract facts by matching them to a set of predefined patterns (e.g. *Rivers in Germany* yields instances of LOCATEDIN). YAGO does not try to clean up the category hierarchy into a true taxonomy, instead it leverages the WordNet taxonomy by mapping categories to WordNet synsets. The mapping algorithm compares modifiers and the headword in the category name with words known to WordNet, and when the mapping is still ambiguous uses the most common synset as the target. TYPE relations connect entities in YAGO to types. Types are linked by SUBCLASSOF and form a directed acyclic graph. Attributes of common Wikipedia infoboxes are manually mapped to relations and provide an important source of facts. The mapping takes into account infobox types, e.g. length in the context of a car means size and length of a song means duration. Infobox types are also used to obtain types of entities. YAGO uses WordNet and Wikipedia redirects to mine surface forms of entities, which are available as the MEANS relation. Provenance of facts is stored in FOUNDIN and EXTRACTEDBY relations indicating the origin of facts and the method used to extract them. To ensure high quality YAGO employs type checking. Sampling based evaluation showed accuracy of about 95%, with many errors originating not from the extraction process, but from incorrect statements in the source documents.

YAGO 2 [64] extends the original system by incorporating three additional kinds of information. First, many facts are naturally associated with time; for example, heads of states change and YAGO 2 knows time period when they are in office. YAGO 2 assigns temporal dimension to facts and entities to capture such information. Second, many entities and facts, such as music festivals or battles, can be assigned a location. Facts are therefore given a geographic component with data coming from GeoNames database (`www.geonames.org`). Third, entities are associated with keywords collected from sources such as anchor text in Wikipedia links. Together these three kinds of data are used to build 6-tuple representation of data, coined SPOTLX (SPO + Time, Location, and ConteXt).

The most recent update to YAGO is YAGO 3 [86], which extends the original system into a multilingual knowledge base.

Freebase. Bollacker, et al. developed Freebase [19]. Knowledge bases are traditionally centrally managed. On the other hand, Wikipedia contains semistructured data, which is collaboratively edited. Freebase tries to merge both worlds by offering a collaboratively edited structured repository of knowledge. To make this possible, Freebase includes a scalable tuple store which supports versioning. The data can be queried with the Metaweb Query Language, MQL for short (Metaweb Technologies is the company which originally developed Freebase). The type system of Freebase is more lenient than those of other knowledge bases, which is a desirable feature in a collaboratively edited resource (e.g. conflicting types are allowed since they can arise from different views of users). More details about Freebase and how it models knowledge are provided in Section 5.2.

Following the acquisition of Metaweb Technologies by Google, Freebase became one of the data source in Google Knowledge Graph [116]. The Freebase team announced in December 2014 that the project will be closed and the existing data transferred to Wikidata project (`www.wikidata.org`).

DBpedia. DBpedia [8] is another example of a widely used knowledge base. Its aim is to support collaborative grass-roots effort in building the Semantic Web by providing a rich and diverse corpus of data. To this end DBpedia makes three contributions: an information extraction framework working on Wikipedia, a large RDF dataset, and a set of software tools to integrate the data in applications. While Wikipedia is edited as text using the MediaWiki markup language, some data is stored in relational databases to facilitate generation of HTML pages. DBpedia extracts this data and information included in infoboxes of important types. The entities are assigned identifiers based on their names in the English edition of Wikipedia. The dataset can be accessed using Linked Data conventions where URIs can be dereferenced over `http` protocol to retrieve the RDF description about an entity. DBpedia can also be queried in SPARQL, or downloaded in a serialized form. The data is interlinked to multiple other data sets, including US census, DBLP, MusicBrainz, WordNet, Cyc, etc. The software provided by the project facilitates embedding the data into Web pages, searching it, and using it in applications.

Taxonomizing Wikipedia. The data contained in Wikipedia is domain independent, up-to-date, and multilingual, which makes it a perfect candidate for extracting structured knowledge. Ponzetto and Strube [109] use it to derive a large-scale taxonomy. Their method uses multiple heuristics to identify the taxonomic part of the Wikipedia category network. They include methods based on the syntactic structure of the category labels and connectivity in the network.

Other Notable KBs. Menta [92] is a multilingual knowledge base, including both entities and classes, built from all language editions of Wikipedia. BabelNet [105] is a multilingual knowledge base constructed from Wikipedia and WordNet, and enriched by applying machine translation techniques to provide

lexicalizations for all languages. Never-Ending Language Learner (NELL) [100] is a system which builds a knowledge base in an iterative fashion by extracting new statements from Web pages and integrating them with an existing, previously learned facts. Probase [137] a large, general-purpose taxonomy focusing on isA relation, which takes the probabilistic approach to representing knowledge. Wikidata (`www.wikidata.org`) is a free (as in freedom), collaboratively edited knowledge base operated by Wikimedia Foundation.

2.2 Information Extraction

Information Extraction (IE) is the process of isolating machine readable facts from content that was originally created for human reading. For an overview of IE see tutorials [12, 121]. In the following section we present an overview of IE systems and problems related to IE, such as relations discovery and extraction of relational paraphrases.

2.2.1 Fact Extraction

DIPRE. In one of the early works on Information Extraction Brin [20] presents DIPRE – Dual Iterative Pattern Relation Expansion. DIPRE is an iterative algorithm which is applied to extracting the book-author relation from Web pages. It starts with a small set of seed instances of the relation, extracts new patterns, and then uses them to collect new instances. While generating patterns from occurrences is in general difficult, DIPRE uses a set of simple heuristics to generate good patterns, for example, pattern lengths are limited, and author names and book titles must match predefined regular expressions. Pattern specificity is measured in order to combat too general patterns. On the other hand too specific patterns are not a problem, since they just lead to no extraction at all. In the experiments a set of 5 books (from which only two produced any patterns) led to over 15000 extractions.

Snowball. Agichtein and Gravano's Snowball [1] improves over DIPRE, by introducing a novel way of generating pattern. Confidence of patterns and extracted relation instances are evaluated and low-confidence data is dropped. Additionally a named entity tagger is employed in pattern matching. Experiments, in which company-location pairs are extracted from a corpus of Web pages, show that Snowball achieves better recall than DIPRE at similar precision level.

TextRunner. Banko et al. [9] propose Open Information Extraction paradigm (Open IE). The aim is to allow building systems that allow processing data at Web scale, extracting all possible relational triples, and doing this without any user involvement. This contrast with previous approaches that required hand crafted patterns and obtained triples for limited number of relations. Open IE was implemented in TextRunner system and later improved in follow up works [10, 11].

OLLIE. OLLIE [90] is an Open Information Extraction system that addresses two weaknesses of earlier works: it is able to extract relations expressed not

only by verbs, but also by other parts of speech, and it considers context of extraction, so that beliefs and conditions do not contribute to erroneous extractions. OLLIE learns a set of patterns for extracting relations and their arguments and applies them to dependency parsed text. The process starts with the seed set of 110000 ReVerb [44] extractions (to ensure quality they must be extracted at least twice, and the arguments must be proper nouns). Seed tuples are used to retrieve sentences from the ClueWeb corpus. Ideally, the retrieved sentences would express the same relations as seed tuples, but this in not always the case. To reduce errors, content words from seed tuples must be connected by short dependency paths in retrieved sentences. It is important that not only arguments, but also relations from seed tuples are matched to retrieved sentences, because multiple relations can hold between two entities. In the next step OLLIE learns patterns for extracting relations by comparing relations in seed tuples with dependency paths in retrieved sentences. Learned patterns can be purely syntactic or contain semantic and lexical constraints. Care is taken to avoid patterns that would match paths containing words which modify their meaning, e.g. *X considered Y as a coach* does not imply (Y, is coach of, X), because the word *considered* changes the meaning of the phrase. OLLIE's extractions can be augmented with *clausal modifiers*, when they depend on a condition, e.g. *If he wins five key states, Romney will be elected President.* Similarly, extractions in the context of words such as *believe* are tagged with *Attributed To* field. Experimental evaluation shows that OLLIE improves recall at similar precision level to earlier ReVerb and WOE$^{\text{parse}}$ systems.

Canonicalizing Open KBs. Open information extraction yields triples in which the arguments and relations are not linked to semantic identifiers, whereas "closed IE" techniques output triples with arguments resolved to canonical names. Galárraga et al. [53] study bridging the gap between the two by canonicalizing entities and relations in the output of an open IE system. The first step is to cluster noun phrases. It is assumed that a single name within a document (e.g. *Obama*) always refers to the same entity (either *Barack* or *Michelle*). Each mention consists of a noun phrase, url of the document where it was found, and a set of relation-value pairs associated with the phrase referred to as attributes (e.g. $\langle$*was born in*, *Honolulu*$\rangle$). Multiple similarity functions are defined for such mentions; they include Jaccard similarity on tokens in mentions, IDF-weighted overlap, Jaro-Winkler string similarity, URL- and attribute-based functions, as well as combinations of them. The similarity functions are used in a hierarchical agglomeration clustering modified for improved efficiency on large data sets. Selection of the canonical name for a cluster of noun phrases is done simply by highest frequency. The idea for clustering verbal phrases is to first learn rules for equivalence of the phrases and then to enforce transitivity. Two verbal phrases are considered equivalent if between them there is subsumption ($\sqsubset$) in both direction, e.g. *was born in* $\sqsubset$ *'s birthplace is* and *'s birthplace is* $\sqsubset$ *was born in*. Subsumption rules are in turn learned by mining them from the data. Finally, relation clusters are mapped to Freebase relations in a similar process of finding equivalence between verbal phrases and Freebase relations.

iPopulator. iPopulator [76] is an information extraction system for automatic population of missing Wikipedia infobox values. It automatically analyzes the structure of infobox attributes, e.g. the number of employees in a company infobox can be a number followed by an optional number, such as *5538* or *92000 (2008)*. iPopulator automatically creates training data by labeling Wikipedia pages with known attribute values. Subsequently, conditional random field (CRF) models are trained for extraction of attributes. The models are automatically evaluated and the good ones, as measured by precision, are kept. Data is extracted only from the beginnings of articles. The experiments compare iPopulator against Kylin and K2 systems [134, 133].

Kok and Domingos [72] extract general, high-level concepts and relations from triples extracted by TextRunner [9]. Their system uses Markov logic to perform simultaneous clustering of objects and relations. StatSnowball [145] is an information extraction system used in an entity search engine EntityCube [43]. Mooney and Bunescu [101] apply machine learning techniques (Relational Markov Network) to information extraction. Experimentation includes various domains: the system extracts facts about protein interactions, books, job offers, and resumes. Minz et al. [99] use vast amount of data available in KBs to automatically label training data for information extraction, which is the paradigm known as distant supervision. First, entity pairs from Freebase are marked in a document corpus, then a multiclass logistic regression classifier is trained to do extractions. Features include words, POS tags, and labels assigned by a dependency parser. Negative examples are generated from random entity pairs. Distant supervision suffers from occurrence of incorrect relation patterns. Multiple methods have been proposed to reduce this problem, for a survey see [113]. Carlson et al. [27] use semi-supervised approach (some labeled points and lots of unlabeled data) to simultaneously learn categories and relations. Judiciously chosen constraints on extracted data ensure that an iterative information extraction algorithm does not exhibit topic drift. Omnivore [23] is a system that performs model independent extraction. Most extractors, while they are not tied to a specific topic, are not model independent, e.g. TextRunner extracts triples from natural language text. At the same time some data is more often represented in tabular form (e.g. GDP) and other in natural language (biographies). To overcome this problem, Omnivore runs multiple, domain independent extractors in parallel and aggregates their results. Koch et al. [71] showed that relation extraction can be improved by using both named entity linking and co-reference resolution for argument identification and making the system type aware.

2.2.2 Beyond Triple Extraction

Temporal Ordering of Relations. Talukdar et al. [124] explore the problem of finding a typical order of relations. While the problem of temporal scoping aims to assign a time interval to facts, e.g. *Bill Clinton presidentOf US (1993–2001)*, the goal of temporal ordering is to mine typical constraints between relations, e.g. a person must be born before he or she becomes a president, or a director must make a movie before he or she is awarded a prize for that movie.

The proposed approach is based on a conjecture that the narrative order of

facts (i.e. order of occurrence in a document) is correlated enough with typical temporal order to serve as a signal for learning the latter. The input to the system is a set of relations to be ordered, a set of facts, and a corpus of documents. YAGO2 and Wikipedia were used in the experiments. In the first phase the system learns which verbs can be used to express given relations. Next, the relative order of verbs is collected. The verbs are considered to express simultaneous events if they occur in the same sentence, or one is considered to express earlier event than the other if they occur in different sentences.

A graph with verbs and relations is built, where verbs are connected with relations they express and other verbs based on the co-occurrence in documents. Finally a label propagation algorithm is used to infer the *before* and *simultaneous* constrains on relations, based on *before* and *simultaneous* relations on verbs and their association with the relations they express.

The experiment show that averaging over large set of documents is beneficial, and that the system is useful for the related task of temporal scoping of facts.

Relation Discovery. Hasegawa et al. [60] propose a method for discovering arbitrary relations among named entities in a large text corpus. Applications of the problem include summarization and question answering. The relations are defined broadly and can mean affiliation, role, location, etc. The main advantage of the described approach is that it does not need initial seeds, and hence is not limited to a predefined list of relations. The paper describes a system based on context clustering. It works in the following steps:*1*) named entity tagging, *2*) extracting pairs with context, *3*) calculating similarity, *4*) clustering, and *5*) labeling. The named entity tagger must annotate the output with classes. Fine grained annotations are beneficial, e.g. breaking ORGANIZATION into COMPANY, MILITARY and GOVERNMENT allows detecting more relation types. Context, which is extracted, are the words between two entities. They are normalized by stemming. Contexts are extracted only if the entity pair was found in the corpus more than some threshold number of times. All contexts (all intervening words) from all occurrences of a pair of entities are merged into a bag-of-words. This representation is used to calculate cosine similarities between pairs of entities. The clustering step employs these similarities in complete linkage hierarchical clustering. In the labeling step most common words occurring in the contexts are assigned as labels to clusters. The system was evaluated on one year of the New York Times corpus. The relations from PERSON-GPE (geo-political entity) and COMPANY-COMPANY domains were manually labeled. The authors report that relations in COMPANY-COMPANY domain were more difficult due to presence of multiple similar relations. The context length was restricted to 5 words only, which helps avoid noise. The authors hypothesize that the outer context, that is parts of the sentence before the first entity or after the second one, may contain useful information for discovering relations, but it would also amplify problems with noise. The labels for relations which were evaluated contain both verbs (e.g. *acquire*) and nouns (e.g. *president*).

PATTY. Nakashole, et al. present PATTY [104] – a large database of textual patterns expressing relations connected by subsumption relation. The pattern

extraction process starts by dependency parsing a corpus of text. Subsequently, named entities are detected and disambiguated. Textual patterns are extracted when a sentence contains two entities. The patterns consist of words on the shortest path connecting two entities in the dependency graph obtained by parsing. SOL patterns in PATTY consist of syntactic (S), ontological (O) and lexical (L) features. They contain:*1*) words, *2*) POS-tags, which are placeholders for words of the right part-of-speech, *3*) * sign, denoting any sequence of words and *4*) <type>, which is a placeholder for an entity of type <type>. For example, a pattern `<person>'s [adj] voice * <song>` matches "Amy Winehouse's soft voice in «Rehab»". The set of entity pairs which support the pattern contains the pair (Amy Winehouse, Rehab).

Patterns can be generalized in many ways, but it is not clear which of them yield meaningful results. PATTY generates all possible generalizations of a pattern and keeps only those which do not change the set of entity pairs supporting the pattern (soft constraint is used for robustness). Patterns can subsume other patterns, for example, the pattern `<politician> was governor of <state>` subsumes `<politician> politician from <state>`. Patty mines such subsumptions and constructs a taxonomy of patterns (non-transitive DAG).

Experimental evaluation was performed on New York Times and Wikipedia text corpora. Two type systems were used, one from YAGO2 and the other from Freebase. It was observed, that more fine-grained and ontologically clean types from YAGO2 yield better patterns than the types from Freebase. Experimental evaluation shows both high precision of patterns and pattern subsumptions, as well as high recall of relations.

Other works studying the problem of mining relational paraphrases include [102, 58].

WebTables. WebTables [25] is a very large corpus of tables extracted from the documents crawled by Google. The tables are filtered so that only those which contain data (as opposed to, for example, table used for layout) remain. Special indexing and ranking techniques are developed to allow retrieval of data from the table corpus.

2.3 Summarization, Exploration, and Search in Knowledge Bases

2.3.1 Summarization for Knowledge Management

The works by Zhang, Penin and Cheng [143, 108, 31] consider the problem of generating a small domain summary for the purpose of ontology retrieval. If a user has a specific task which requires formalizing knowledge, for example describing wine, an already existing ontology can be reused. The systems tackle the task of finding a good existing ontology, which includes snippet generation. We describe these works in details below.

Ontology summarization. Zhang et al. [143] proposed a system for summarizing ontologies. When engineers look for an existing ontology which can be reused, a specialized search engine provides them with many results which

have to be quickly evaluated. Judging usefulness of existing ontologies requires generating appropriate summaries.

The authors identify two desirable properties of a summary of an ontology: first, it should be concise and indicative, which translates to constraints on its size and relevance of included statements; second, it should be coherent and extensive, which means lack of redundancy and good coverage of summarized statements.

The summarization algorithm operates on RDF sentences, which are sets of RDF triples connected by blank nodes. Blank nodes provide context which is necessary to understand RDF statements and therefore RDF sentences cannot be meaningfully broken down any further. RDF sentences can be connected by sequential and coordinate links, if they share subjects, predicates or objects. The kind of link depends on the parts of the RDF sentences which are shared.

The sentences in the RDF sentence graph are ranked with various vertex centrality measures, such as in- and out-degree, shortest path centrality, PageRank, HITS, and their variants. The final summary of an ontology is generated by re-ranking the sentences in a fashion similar to the Maximal Marginal Relevance introduced in [26]. This final step was introduced to ensure coherent and not-redundant summary. The measures of both properties use the notion of sequential and coordinate links introduced earlier.

The work was extended in the follow-up [108], which studies the same problem of generating snippets of an ontology. It also operates on RDF sentences created by linking triples through blank nodes. One of the contributions of the work is the measure of similarity on RDF sentences, which is used to group related sentences into RDF topics. The system presents the snippets as text using a natural language generation component.

In a follow-up on [143] Cheng et al. describe [31] an ontology summarization system. The system uses a novel method of computing salience of RDF sentences, which is combined with query relevance and summary cohesion metrics. The final summary optimizes a linear combination of goodness measures:

$$\begin{aligned}\text{Goodness}(o, S, Q) := (1 - \alpha - \beta)\ &\text{Salience}(o, S) \\ &+ \alpha\ \text{Relevance}(S, Q) \\ &+ \beta\ \text{Cohesion}(S),\end{aligned}$$

where o is the ontology, S the summary and Q the query. In order to calculate salience of RDF sentences, a sentence-term bipartite graph is built, where each RDF sentence is connected to a term it describes, then a variant of PageRank is run on the graph. Salience of a summary is the sum of PageRanks of its constituents: $\text{Salience}(o, S) := \sum_{s \in S} PR(s)$. Relevance of the summary with respect to a query is measured by keyword overlap between sentences s and the query. The score for the whole summary is again the sum of scores of its elements:

$$\text{Relevance}(S, Q) := \sum_{s \in S} \text{TextSim}(s, Q) = \sum_{s \in S} \frac{|\text{KWSet}(s) \cap Q|}{|\text{KWSet}(s)|}.$$

Finally the cohesion is measured by overlap of terms described by RDF sentences. Let Desc(s) be the set of terms described by the sentence s, then

$$\text{Cohesion}(S) := \sum_{s_i, s_j \in S \text{ s.t. } s_i \neq s_j} |\text{Desc}(s_i) \cap \text{Desc}(s_j)|.$$

The system was integrated into Falcons Ontology Search [30].

2.3.2 Entity Summarization

Diversity in Graphical Entity Summarisation. Sydow et al. [122] study the role of diversity in graphical entity summarisation (GES) in entity-relationship (ER) graphs. In the running example, nodes in the graph represent actors, movies, and directors, and the edges relations between them. Multiple edges are allowed, for example a person may direct and star in the same movie. Edges can have weights which reflect their importance or other qualities, such as novelty or credibility.

ER graphs can be explored by means of structured queries in SPARQL, but for users unfamiliar with the schema, domain, or SPARQL language keywords queries can be better. The most basic unstructured query on ER graph is: *what are the most important facts about a given entity?* Summarization can hence be viewed as query answering or information retrieval. Summarization on the graph is also related to text summarization. The advent of modern mobile devices with small screens calls for more work on presentation of structured information when space is at premium (we are given some presentation budget).

Yet another view on summarization in ER graphs is by analogy to text IR – entity corresponds to a query, ER graph to a document corpus and summary to the result set. Diversity in the result set is analogous to high coverage of different edge labels.

The focus of the work is on diversification in GES. Diversification helps when the user intent is unknown, e.g. summary of Tom Cruise should contain both facts from the private life (birth date, spouse) and professional life (films he acted in).

Formally, the graphical entity summarization is stated as follows, given an ER graph D, node q, and budget k, find a connected subgraph S of D containing q and at most k edges. Two algorithms are presented: diversity-oblivious PRECIS and diversity-aware DIVERSUM. Weights on edges are treated as an external input. Facts in ER graphs constitute edges. The most important facts about an entity are topologically closest, but there may be many of them. PRECIS simply selects k edges that are closest to q (the distance is defined as the inverse of the weight). In the experiments on a data set extracted from IMDB (movie domain) and the query *Tom Cruise* the algorithm selects only edges labeled `actedIn`, which is clearly not optimal. DIVERSUM restricts repetitions of labels in the selected subgraph. The graph is divided into zones, such that ith zone is i hops away from the query node q. Starting from the zone nearest to q the algorithm iterates over label in order of decreasing frequency and adds edges with minimum distance to the query node. A label can occur at most once within each zone.

Experiments were performed on a dataset extracted from IMDB (`www.imdb.com`) and YAGO. Witness counts from [42] were used as edge weights. Some relations and incorrect facts were pruned. Both algorithms were run on 20 selected actors with budget $k = 7$ and $k = 12$. Gold-standard summaries were prepared using Wikipedia infoboxes. Evaluation takes into account semantics of relations and dependencies among them, e.g. if a summary contains a fact with `actedIn` relation then it is counted as "hit" towards `isA actor`. The two presented algorithms were compared against the gold-standard summaries as well as against each other with help from human judges. Experiments show that the diversity-aware approach outperforms the diversity-oblivious algorithm.

AGNES. Sobczak et al. [117] developed AGNES – a visualization algorithm for entity summaries. The input to the algorithm is a small summary graph, for instance, output from DIVERSUM [122]. The algorithm automatically computes a layout which fulfills several desirable properties: *1*) summarized node is in the center of the layout, *2*) other nodes are placed evenly around it, *3*) nodes close to the central node topologically, are also placed close to it, *4*) multiple edges and long textual labels are supported, *5*) overlap of graph elements and labels is avoided. The work was motivated by deficiencies of other general layout algorithms, which in particular did not allow selection of a central node and were unable to handle long node and edge labels.

AGNES starts by calculating a spanning tree of the graph rooted in the central node. The tree is traversed in-order and natural numbers are assigned to non-central nodes. The numbers determine the angle of the node position in radial coordinates. The radius is calculated based on the distance of a node from the root and the size of a subtree rooted at it. Thus the layout of the spanning tree resembles a spider's net. Finally, the remaining edges of the graph are added to the tree.

Experiments performed on a crowdsourcing platform show that humans favor AGNES over a general graph layout tool Gephi (`gephi.org`).

Templates for Summarization. Li et al. [81] studied the problem of automatic generation of templates for entity summaries. Their method was applied to mine sentence templates from introductions of Wikipedia articles, which usually contain a short summary of the whole article. The method exploits the fact that sentences used in articles are similar within categories, for example, articles about physicists contain sentences about their alma mater, awards, contributions, etc. Sample templates for physicist include *ENT made contributions to ?*, *ENT is noted for ?*, *ENT won the ? award*. Such human readable sentence patterns contain placeholders for the entity and one or more arguments. Templates are grouped by topics, for instance, the first two sample templates express the same fact.

The templates are generated in two steps. Aspects (entity types, e.g. university that a person graduated from) are assigned to words in summaries, then frequent patterns are mined to obtain the templates. The key insight in aspect identification is that the words in summaries belong to four categories:*1*) stop words, *2*) words specific to a category of entities, *3*) words associated with an aspect and *4*) document (entity) specific. An LDA-based method is used to

assign each word to one of the categories. Subsequently, sentences in summaries are parsed, frequent subtrees are mined from the parse trees, and converted to patterns. Additional steps ensure that the patterns contain placeholders for entities and arguments and do not contain document specific words (i.e. words specific to a particular entity).

2.3.3 Search and Exploration

Falcons. Cheng et al. [30] developed Falcons – a search engine and browser for Semantic Web. The system crawled 7 millions documents containing more than 250 millions RDF statements. Its functionality includes keyword search for concepts and objects (entities), searching for ontologies, and ontology recommendation. Search results for entities are not limited to a list of names, but also include a short summary consisting of statements where it occurs.

The functionality of Falcons is build on the foundation of *virtual documents*. A virtual document is created for each entity, it contains its names as well as the names of neighboring nodes. The virtual documents allow leveraging existing information retrieval techniques to provide search and ranking on RDF data.

Tabulator. Tabulator developed by Berners-Lee et al. [15] is a generic (not tied to any particular domain) browser for linked RDF data on the Web. The user can navigate data along predicates in a web of concepts. Tabulator recognizes two modes of interaction with data: exploration mode resembles how we use WWW, the user follows links to discover new nodes; analysis mode resembles money management applications, which focus on sorting and visualizing data whose structure is known. While the RDF data is a graph, Tabulator presents it as a tree. This contrasts with other systems that use point and arrow representation, which, while intuitive, is not space efficient. Additional views emphasize specific RDF properties, e.g. map view presents geographical locations on a map, calendar shows data in a calendar, etc. Tabulator automatically dereferences URIs to recursively collect additional data. Some inference is performed, for example, on owl:sameAs predicate, but this is limited by the necessity to keep the performance acceptable (interactive), since the system runs in a web browser on the client side. Tabulator was extended to also support write access to the Semantic Web [16].

CATE. Tuan et al. developed CATE [127] – a system for context aware presentation of entities. CATE shows entities on a time line augmented with other related entities and events. For example Carl Friedrich Gauss is presented with other mathematicians (e.g. Legendre, Riemann) and historical events, which happened during his life (e.g. French Revolution). The notion of context in CATE is defined as an object with three attributes: time, location and topic. Contexts of an entity are obtained from its Wikipedia categories which match certain patterns, e.g. *18th_century_mathematicians* matches time 18th century and topic mathematics. The data used in the systems is divided into three parts: YAGO knowledge base, text of Wikipedia pages about entities, and a set of images retrieved for the entities. The central part of CATE is a set of language model based methods devised to rank entities with respect to contexts, and contexts with respect to entities.

BANKS. BANKS (Browsing ANd Keyword Searching) [17] was designed to let users unfamiliar with query languages or schema use keyword queries to search information in relational databases. A database is modeled as a graph with tuples as nodes and foreign-primary key relations as edges. Query terms are matched with attribute names and values . The answer to a query is a rooted tree with a directed path from the root to each node selected by a keyword. BANKS ranks the answers based on the connection between the tuples and their importance measured by a method similar to PageRank.

MING. Kasneci et al. [67] consider the problem of mining informative entity relationship subgraphs. Given a large graph, a set of k (≥ 2) query nodes, and a budget b, the task is to find a subgraph with at most b nodes, which captures the connection between query nodes. For example, given the knowledge base YAGO [119] and two physicist, Einstein and Planck, find the connection between them. The paper proposes a method to measure informativeness of nodes and edges in the graph and an algorithm to extract a subgraph connecting the query nodes. It is argued that the link structure alone is not sufficient to capture the informativeness in the graph, since it only represents a fraction of the world. Therefore the authors resort to statistics obtained from a text corpus to measure informativeness. Informativeness is obtained for entities as well as for the edges. The edges represent endorsements, which propagate importance from one entity to the other. In general it is not symmetric and therefore edges are assigned different weights in both directions. MING runs a PageRank-style algorithm on the entity relationship graph to propagate informativeness. The informativeness of entities, obtained from the corpus, is used as the probability of randomly jumping to the entity. After assigning the informativeness, a subgraph connecting the query nodes is extracted. First a recall-oriented pruning is performed to reduce the size of the graph. Then the STAR algorithm [68] is run to obtain a tree connecting the query nodes. Nodes of the tree are labeled + and nodes of degree 1 (which do not connect anything) are labeled -. Labels are propagated based on the connection to already labeled nodes. The final connecting subgraph is extracted from the nodes labeled +. Experimental evaluation on a data set extracted from YAGO shows that MING is superior to baseline CEPS [126].

Ranking queries on RDF graphs. Elbassuoni et al. [42] consider the problem of ranking results for queries on RDF/ER graphs. Such graphs can be explored using SPARQL queries, which may require result ranking, relaxation of queries if there are too few results (for instance if the graph does not exactly match the query), and support for keywords in the queries.

The work exploits the fact that RDF graphs are often extracted from semi- and unstructured websites. RDF triples are augmented with witness counts, which are the number of times a triple was seen in the corpus from which the data originated. Additionally, the witness count is multiplied by the confidence of extraction. Triples are also associated with keywords, which also have witness count.

Queries are sets of triple pattern containing variables; they can optionally contain keywords, e.g. `Woody Allen produced ?x {murder lover}. Woody`

`Allen directed ?x` is a query for movies directed and produced by Woody Allen where murder and lover are the theme.

Relaxation is performed by replacing constants with variables and dropping keywords. Queries are ranked using language model developed for RDF graphs. In contrast to traditional IR, there are no documents, every subgraph with the right structure is a potential answer; another difference is the *vocabulary gap*: query contains variables whereas answers contain constants.

ROXXI. The problem of finding SPO facts in documents has been studied in the demonstration proposal by Elbassuoni, et al. [41]. The motivation is to allow users who browse knowledge bases, through a graphical browser, keyword search, or with SPARQL, to see the original documents which provided the facts to the knowledge base. This lets users judge the correctness of the facts and learn additional information which was not extracted.

Interaction with ROXXI start when the user submits an entity or SPARQL query to retrieve facts. The facts, which may or may not form a connected graph, are presented in a graphical form together with other neighboring facts from the knowledge base and documents which support them. Documents supporting the facts are ranked using a novel method based on statistical language models. They are presented with query dependent snippets that the system generates. When the user selects a document, occurrences of facts are highlighted. The system takes advantage of the metadata in the knowledge base. It is required that the extractor saves not only extracted facts, but also patterns used, documents where the facts occurred, and their positions within documents.

Ranking heterogeneous entities. Zaragoza [142] et al. propose the problem of ranking many entities of different types with respect to an ad-hoc query. For example, a search engine can present to a user a heterogeneous set of results consisting of documents as well as people, countries, dates, etc. The study uses an annotated Wikipedia corpus as a source of entities. Given a query their system retrieves a set of 500 passages from Wikipedia. The entities contained in these passages are to be ranked. The real relevance of entities is provided by human judges for 50 queries. The baseline for ranking of entities is the maximum score of the passages that the entity was extracted from. The study considers two more complex methods of ranking. First the authors describe methods which use bipartite graph of passages and entities extracted from them. The simple degree of an entity in such a graph performs better than the baseline. The results can be further improved if the relative frequencies of entity types are considered. For example country names are popular entities and therefore they have high degrees in entity-passage graph. This degree can be discounted by overall frequency of the entity type (country) in the corpus. The second class of entity ranking methods considers correlations between Web search results for an entity and the original query.

Facetedpedia. Li et al. propose [79] Facetedpedia – a faceted retrieval interface for Wikipedia. The system takes a query, e.g. “us action film”, obtains a ranked list of Wikipedia pages from an external search engine, and builds

facets for browsing them, e.g. by actors or companies. The interface of Facetedpedia is query dependent and generated dynamically from Wikipedia categories. This contrasts with prior works which focused on relational tuples and objects with available scheme, where dimensions of retrieved objects are known upfront and faceted interface could be precomputed.

The articles retrieved by the query are called *target articles*, and the articles they link to are called *attribute articles*, for examples, an article about a movie can link to the actors who played in it. The relation between a target article p and an attribute article p' is denoted by $p' \leftarrow p$. A *facet* is a rooted and connected subgraph of the Wikipedia category hierarchy. In a *safe reaching facet* each category leads (through an attribute article) to a target article (no dangling categories without any pages). A *faceted interface* is a set of safe reaching facets. Navigational path is a path in a facet that ends on a target article $c_1 \rightarrow \ldots \rightarrow c_k \Rightarrow p' \leftarrow p$. Faceted interface discovery problem is to find the "best" interface for a set of pages returned by the query. The challenges include defining a good function for the quality of a faceted interface and finding it efficiently (it is query dependent, so it has to be interactive).

To navigate a faceted interface the user selects categories in facets, usually in top-down fashion starting at the root and then selecting one of its children. The user can also alternate between different facets. Selected categories form a conjunctive query which narrows down the original set of retrieved documents. The cost of a path $c_1 \rightarrow \ldots \rightarrow c_k \Rightarrow p' \leftarrow p$ is $\log |\{p : p' \leftarrow p\}| + \sum_k \log fanout(c_k)$ where $fanout$ of a category is the number of its direct subcategories and attribute articles. The cost of a facet is defined as the average cost of reaching a target article through a path from the root plus a penalty for unreachable articles. The best k-facet interface is usually not the cheapest k facets. Users can jump between facets during navigation, and enumerating all possible choices is unfeasible. In general, facets should not overlap much, this is captured with average pairwise similarity of a k-facet interface. The similarity is defined as a variant of Jaccard coefficient.

Facetedpedia searches the space of possible facets in three stages: *1*) A subgraph of Wikipedia categories which lead to target articles is extracted. They are called *relevant category hierarchy* (RCH). *2*) Top-n facets in RCH with lowest navigational cost are found. The recursive algorithm avoids enumerating all facets. *3*) A k-facet interface is found from the set of top-n facets from the previous step.

S3K. S3K developed by Metzger et al. [94] is a system for semantic-aware retrieval of documents. The input to the system is a query in textual form or a set of RDF triples. S3K system works in three stages. First, a document corpus is pre-processed with an information extraction tool which finds facts in documents (witnesses). When the system receives the query it translates it to a set of RDF statements. Finally, witness documents are retrieved and ranked. Two possible user intentions are recognized: *1*) verifying whether the statement is true, and *2*) learning more details about the statement. Documents which satisfy the first intention are called *persuasive*, and those which satisfy the second are called *on topic*. The ranking model also takes into account the reputation of the source document, as calculated by PageRank and clarity of

statements in text, for example, a phrase *was born in* expresses unambiguously a fact, whereas *'s home country* can have other meanings (spent childhood there, etc.).

S3K uses two dictionaries, one with a set of possible names of entities, and another one with phrases indicating relations. Since the information extraction was not the main contribution it is not discussed in detail and treated as a black box. Document ranking is based on statistical language models and operates on the statement level, i.e. documents are ranked with respect to RDF statements, not words. Smoothing is used to avoid overfitting and account for missing query statements. The probability of generating a statement from a document takes into account, that that a relation can be expressed by many patterns with different confidences. The system was evaluated on a part of ClueWeb09 corpus. The baseline system was Apache Lucene with statement queries treated as standard keyword queries.

Le Monde. Huet et al. [65] link the knowledge base YAGO with an archive of the French newspaper Le Monde spanning years 1944 – 1986. Such connection allows understanding trends in the data in a way that would not be possible with word frequencies alone (for study of a large corpus with word frequencies see [95]). For example, semantic data about entities allows calculating what fraction of politicians mentioned are female.

Madhavan et al. [85] consider the problem of integrating structured data into a Web search engine. Challenges include integrating data coming from multiple sources into a large repository and harvesting data from *deep web*, for example, from web forms that allow querying databases of local services, businesses, schedules, etc.

The extraction graph created by TextRunner is an approximation of the real entity-relationship graph (names are not canonical). Cafarella et al. [24] study how it can be used to answer relational queries.

Anyanwu et al. [6] propose SemRank, a method for ranking complex relationship search results.

2.4 Results Diversification in Information Retrieval

Results diversification in information retrieval is a solution to two common problems. First, some queries are ambiguous, for example the query *jaguar* can mean a car, an animal, or a model of a guitar. Ideally, when the interpretation is not known, search results for such queries should contain a mix of documents for all interpretations (the other solution could be to personalize results based on the most likely interpretation). Second, large text corpora can contain duplicates and near duplicates, which may all be relevant to a query, but the result of a search should still contain each document at most once.

Maximal Marginal Relevance. In one of the seminal works on diversity in search Carbonell and Goldstein [26] present a method of combining query relevance with novelty in text retrieval and summarization. While retrieval by relevance alone is often appropriate, it is not the case in presence of multiple

relevant documents which are redundant. The proposed method utilizes a linear combination of relevance and novelty, coined maximal marginal relevance (MMR). MMR is defined as

$$MMR := \arg \max_{D_i \in R \setminus S} \left[\lambda Sim_1(D_i, Q) - (1 - \lambda) \max_{D_j \in S} Sim_2(D_i, D_j) \right]$$

where Q is the query, R the set of retrieved documents and S the set of already selected documents from R. λ is selectable by the user; small values are useful for exploration of the results, larger values can be used to drill down a particular topic (with possibly reformulated query). A study with 5 users demonstrated that MMR is superior to text retrieval based on relevance alone.

MMR can also be applied to text summarization, where the problem of redundancy is even more important, especially in multi-document summarization. In the experiments with MMR summarizer documents are segmented into sentences. Sentences which are most relevant to the query are retrieved, re-ranked with MMR and top-k of them presented in the original document order.

Diversity for ambiguous queries. Agrawal et al. [2] study the problem of diversifying search results when the query is ambiguous, e.g. "Flash" can be technology used in the Web, an adventure hero, or a village. A query classifier is used to obtain a probability distribution of intended categories of a query. The objective is to minimize the probability that among top-k results the user will not find any relevant. The crucial property of such objective function is that once we have a document which is good for category c_i it is better to add documents relevant for other categories, than keep adding documents relevant for c_i.

Probabilistic Model for Fewer Relevant Documents. Chen and Karger [33] show that diversity in information retrieval is a natural consequence of optimizing a particular model of user information need. The probability ranking principle (PRP), that is retrieving documents in descending order of relevance probability, is optimal when the goal is to return as many relevant documents as possible. Alternative goals are possible, for example the system may be optimized for returning any relevant documents. Some queries, for instance "trojan horse", have many possible interpretations, under the PRP the most common interpretation is considered and others skipped. The work proposes a new success metric – k-call at n. It is one if at least k of top-n retrieval results are relevant and zero otherwise. Since optimizing the metric directly leads to an NP-hard problem, a greedy algorithm is proposed. In case of $k = 1$ it leads to an iterative algorithm, where ith document is chosen to maximize the probability of relevance (like in PRP), but under the assumption that all document retrieved so far (1 to $i-1$) are irrelevant. This can be seen as negative relevance feedback. The side effect is increased diversity of the results, for instance, all meanings of "trojan horse" are considered.

Ambiguity, redundancy, and evaluation metrics. Modeling relevance in presence of ambiguous queries and redundant documents is the subject of the work by Clarke et al. [35]. Evaluation metrics, such as NDCG, do not consider ambiguity and redundancy. For example, retrieving a set of identical,

relevant documents would yield perfect NDCG, but would not be preferred by a user. The work models user's information need as a set of information nuggets, which may be, for example, questions that a retrieved document answers. Document also contain nuggets and are deemed relevant if they contain at least one nugget from the user's information need. The article introduces an evaluation measure based on NDCG, coined αNDCG. The new measure captures *novelty* (avoiding redundancy) and *diversity* (resolving ambiguity) in search results. In one of the experiments αNDCG shows that pseudo-relevance feedback decreases the diversity of search results, which had been suggested by earlier works. Such effect could not be measured with regular NDCG and other diversity oblivious measures.

2.5 Truth Finding

Truth finding systems automatically decide whether a given fact is true or not. While the problem is often ambitiously motivated with challenging tasks, such as deciding whether the statement *Obama is a Muslim* is true or false, the applications of truth finding systems are more utilitarian and common. For example, they can be used to decide whether an information extraction system made a mistake. Design of truth finding systems depends on assumptions about the origin and nature of true and false facts. For example, one may assume that in case of attributes, true values are similar to each other, but may be slightly different due to abbreviations, this holds for instance in case of lists of authors of books. Examples of works about truth finding include [84, 55, 140, 106, 39, 82, 83]. Below we describe some of the systems in more details.

Linguistic Features for Truth Finding. Nakashole and Mitchel [103] present a truth finding system which exploits linguistic features to assess reliability of a fact source. The underlying assumption, verified by Mechanical Turk experiments, is that true facts are expressed in neutral language, whereas subjective language often indicates that the statements are doubtful. The approach uses subjectivity and bias lexicons to asses reliability of fact sources.

T-verifier. The system developed by Li et al. [84] takes as an input a statement and using Web search decides whether it is true. In case it is false, a truthful version of the statement is provided. The focus are factual statements, which can be verified, as opposed to opinions. It is assumed that true versions of statements exist in the Web and are more prevalent than their false counterparts. A user submits to the system a doubtful statement, such as "Barack Obama is a Muslim", consisting of a topic unit (Barack Obama), and a marked doubt unit (Muslim). In the first stage, T-verifier submits the *topic unit* as a query to a search engine and considers all terms from the result pages as candidates for alternative units, which could substitute the doubt unit in the statement (e.g. Christian). The alternatives are subsequently ranked and the top-5 results are passed to the next part of the system. The goal of ranking is to find alternatives, which have the same topic as the doubt unit, but a different value, and are close in terms of type/sense. The last requirement ensures that "Christian" is considered an alternative to "Muslim", but "president" is not. In order to rank alternatives the authors propose several features based on occurrence of terms in search

results, which capture how relevant they are with respect to the statement topic and their similarity to the doubt unit. The final score is a linear combination of features with coefficients obtained by running a genetic algorithm. The authors showed that the top-5 results from the first stage are highly likely to contain the correct alternative to the doubt unit (or the doubt unit itself, if it is correct). The second stage of T-verifier re-ranks the output of the first stage. For each of the-5 results a query is submitted to a search engine. The results are used to build features for re-ranking. Some features from the first step are reused, but they are calculated for the new query containing an alternative statement. Other features include number of relevant pages retrieved and domains where the statements occur, for instance, edu and gov are considered trustworthy. In the second stage rankings by different features are combined with standard Borda and Condorcet algorithms. Since rankers have different and measurable accuracy, weighted versions of Borda and Condorcet are proposed as well. The evaluation uses the data from TREC-8 and TREC-9 Question Answering track.

Corroborating Information. Galland, et al. [55] investigated reconciling conflicting information from disagreeing sources. A real-world example of this problem is retrieving professional contact information of a person, when outdated information is available alongside the current data. Their model assumes a set $\mathcal{F} = \{f_1, \ldots, f_n\}$ of facts and a set $\mathcal{V} = \{V_1, \ldots, V_m\}$ of views over facts. Views are partial functions from $\mathcal{F}$ to $\{T, F\}$. Since normally sources state only positive facts, the negative facts are introduced through functional dependence. For instance, if a source states that "Paris is the capital of France", then it implicitly says that Lyon is not. Facts are assumed to have some logical value in the real world W. With each view the model associates its reliability, that is how much it agrees with real world, and its ignorance which captures how many facts do not have a value in the model. Facts are assigned a parameter which describes how difficult they are to get right. The goal of the work is to estimate the logical values of facts in the real world, the reliability of views, and optionally the difficulty of facts. To this end, the authors developed three algorithms that iteratively estimate the reliability of views and truthfulness of facts. The quality of predictions made by the system was evaluated on synthetic and real world data sets. Two real world data sets are extensively discussed. The first comes from an online betting system Hubdub. It contains 357 question in the domain of sports. The total of 3051 statements were made by 437 users. The other data set is a general knowledge quiz of 17 questions with 4 – 14 possible answers for each question. The quiz was tried by 601 people, who were allowed to skip some questions. The results of experiments show that proposed algorithms are able to beat the baseline algorithm based on voting or counting facts. The baseline has nevertheless performed reasonably well.

TruthFinder. Yin, et al. [140] developed TruthFinder – an algorithm for discovering true facts from multiple sources. The authors argue, that trustworthiness of information, which has been shown to be an important problem for users, is not captured by ranking algorithms such as HITS or PageRank. The relation between truthfulness of pages and facts is recursive: correct facts come from trustworthy pages, and trustworthiness of websites can judged from the facts they contain. Additionally, facts support other similar facts. For instance a facts

J. Widom wrote book B supports a similar facts *Jennifer Widom wrote book B*. The computational model proposed by the article rest on the following assumptions: *(i)* there is usually one true value of a fact, *(ii)* the true fact has similar representations on different pages, *(iii)* false facts are not similar, *(iv)* if a website provides true facts for many objects, it will also provide true facts for other objects. The TruthFinder algorithm iteratively estimates confidences of facts and websites until convergence. The confidence of a website has a very simple model – it is just the average of confidences of the facts it contains. The corresponding formula for facts is more involved. It is developed in the following steps: *(i)* a simple model of fact probabilities when pages are independent and there is no dependency among facts is proposed, *(ii)* a correction for influence between facts is added, *(iii)* another correction handles dependence among websites (due to fact copying), *(iv)* another correction is needed to improve the behavior of algorithm, when a fact is in conflict with another fact from a trustworthy website. The experiments were performed on a dataset of computer science books and authors extracted from online bookshops. It contains 1265 books with an average of 5.4 different author sets per book. 100 books were randomly sampled and their real author sets were verified by a human. The baseline for TruthFinder is voting. The accuracy of algorithms was measured with methods designed for the problem. They cover cases when a name is abbreviated or a middle name is missing and assign them lower score than for a full name. The results of the experiments show that TruthFinder is better than voting, and than data from the largest bookshop – Barnes&Noble. Additionally, it is shown that ranking of bookshops with TruthFinder captures accuracy of their catalogues better than their ranking in Google (with the query *bookstore*).

2.6 Named Entity Disambiguation

Named entity disambiguation, also known as named entity linking, is the problem of linking mentions in a document to canonical named entities, for example, to their Wikipedia IDs or knowledge base IDs. There is a broad range of work in this subject; for a recent survey see [115]. The systems by Bunescu and Pasca [22] and Cucerzan [36] disambiguate named entities by exploiting the contextual information extracted from Wikipedia. TAGME system by Ferragina et al. [49] focuses on disambiguating entities in short and poorly composed documents, such as tweets and news. A more general problem is Wikification [96, 98, 74, 111, 32], which is also linking mentions to their canonical identifiers, but the mentions are not limited to named entities, and can be common nouns such as “musician”.

Below we describe a state-of-the-art system, AIDA, which is of particular interest to us, since its core algorithm, based on finding a coherent set of entity-to-mention assignments, is related to our optimization step in fact spotting (see Section 5.5). We also compare to AIDA the performance of the named entity disambiguation subtask in fact spotting.

AIDA. AIDA [63, 141] is a system for named entity disambiguation. Input to AIDA is an arbitrary text, which can include HTML markup, tables, etc. The disambiguation methods relies on two key insights:

1. **Context similarity.** Context of a mention is related to the mentioned entity. If a name, such as *Kashmir*, is ambiguous, the context of a mention should be compared against canonical documents about possible matching entities (for instance, the Wikipedia pages about the region Kashmir and the song Kashmir) to disambiguate it. This method is known to work well for long documents.

2. **Coherence.** Entities mentioned in a document are related to each other. Joint disambiguation can exploit this coherence. This observation improves disambiguation in presence of many entities.

These key insights are combined with prior probability of a surface string meaning an entity, that is, its popularity. It is estimated using the number of times a name is used as a link to the entity page in Wikipedia. Entity mentions are detected either by Stanford NER tagger [51] or given as an input (for example, tagged manually). AIDA builds a graph of entities and mentions. Mentions are connected to entities from YAGO with edges, whose weights indicate the similarity of the mention context and the entity context. To this end, for each entity a set of keywords is collected, which are compared against mention context to yield similarity. The comparison can use weighted word overlap, KL divergence, or any other measure. Entities are also linked and the edge weight indicates the coherence, that is the likelihood that they appear together. It is measured by counting incoming links in Wikipedia shared among the entities. The graph is reduced in a greedy fashion by dropping entity nodes with the lowest weight until each mention is assigned one entity.

2.7 Question Answering

The problem of question answering has been tackled by many systems. One of the most prominent examples is IBM Watson [50]. The work which we consider closely related to our fact spotting (Chapter 5) is DEANNA by Yahya et al. [138]. It translates questions to SPARQL queries by mapping phrases in the question to semantic resources, such as entities, classes, and relations. We describe the system in more details below. KB-supported question answering was also studied in [14]. Fader et al. [46] developed a system for question answering over a noisy collection of tuples extracted from the Web. Open QA [45] answers queries using a combination of automatically extracted and curated knowledge bases. A particular subproblem of question answering, which is related to our fact spotting problem, is answer validation [112]. Answer validation is the process of deciding whether a text snippet supports an answer to a question, which can be used to select the best answer to a question from a set of candidates.

DEANNA. Yahya, et al. [138] developed DEANNA – a system for translating natural language questions to SPARQL queries. The system takes as the input a question, e.g. "Which female actor played in Casablanca and is married to a writer who was born in Rome?", maps the natural language to semantic resources and generates a query, which can then be answered by SPARQL engine operating on RDF data. The problem is motivated by difficulty of writing queries, which have to operate on complex datasets such as Linked Open Data,

where even advanced users may not know canonical names of entities and relations of interest.

While in general natural language and formal query languages allow complex operations such as aggregations, negations, and comparisons, DEANNA limits the scope of the problem to select-project-join queries. The main challenge is mapping ambiguous phrases of natural language to semantic resources: entities, classes and relations. The underlying knowledge base used in the experiments is YAGO [64] linked with IMDB data.

DEANNA translates questions to SPARQL as follows:

1. **Phrase detection and mapping** Token sequences from the inputs are linked to (possibly) many semantic items (entities, classes, and relations). Sequences may overlap, e.g. both `is married to` and its subsequence `married` will be linked with the same relation. Entity detection utilizes YAGO's `means` relation; relations are detected using a custom approach based on ReVerb [44]. A special *null phrase* captures relations expressed without any tokens, e.g. *Australian movie*.

2. **Q-unit generation.** Phrases are linked together into so-called q-units, based on the output from dependency parser. Q-units are raw representation of dependencies between phrases. They are essentially triples of phrases.

3. **Joint disambiguation.** In this step phrases linked through q-units are mapped to semantic items by means of integer linear programming (ILP). Use of ILP allows encoding complex constraints on the mapping. Typical run-times of this step are a few seconds.

4. **Query generation.** The graph of phrases and semantic items together with labels introduced in the disambiguation step is translated to SPARQL queries, e.g. semantic node `writer` becomes a statement `?x type writer`.

While DEANNA is a question answering system, the challenge it solves is mostly a complex disambiguation problem. The system therefore closely resembles named entity disambiguation (NED) frameworks such as AIDA. Similarity includes building a graph of phrases and semantic items (or mentions and entities in AIDA), employing coherence function on entities, and reducing the mapping graph in the ILP (or removing edges in AIDA).

CHAPTER 3

Entity Summarization

The Web and, in particular, knowledge-sharing communities such as Wikipedia contain a huge amount of information encompassing disparate and diverse fields. Knowledge bases such as DBpedia or YAGO represent the data in a concise and more structured way bearing the potential of bringing database tools to Web Search. The wealth of data, however, poses the challenge of how to retrieve important and valuable information, which is often intertwined with trivial and less important details. This calls for an efficient and automatic summarization method.

In this chapter, we consider the novel problem of summarizing the information related to a given entity, like a person or an organization. To this end, we utilize the rich type graph that knowledge bases provide for each entity, and define the problem of selecting the best cost-restricted subset of types as summary with good coverage of salient properties. We also implemented a browser, which allows exploring the knowledge base in a simple and intuitive manner, and includes a demonstration of our summarization methods.

3.1 Introduction

3.1.1 Motivation

Knowledge-sharing communities such as Wikipedia represent a huge and surprisingly reliable source of information in a wide variety of fields. Knowledge bases such as DBpedia [8], YAGO [118, 119, 64, 86], or Freebase [19] are a concise, formal representation of (specific pieces from) such encyclopedic sources. To open up ways of using structured query languages for knowledge discovery, effective techniques for querying have been developed. They include augmenting queries with keywords, query reformulation to improve recall, diversification, and efficient algorithms for top-k processing [40]. By extracting or tagging entities and their attributes in Web pages and linking them to corresponding facts in a background knowledge base, this can be further leveraged for semantic search on the Web.

Search engines for structured knowledge (in knowledge bases or gathered from the live Web), such as Entity Cube [43], Google Squared, Wolfram Alpha [132], sig.ma [128], NAGA [69], or Broccoli [13] tend to either give very brief answers,

Item Name	Image	Description	Date Of Birth	Nationality	Place Of Birth
Ursula Andress		**Ursula Andress** (born March 19, 1936) is a Swiss actress and a major sex symbol of the 1960s. She is best known for her roles as Bond girl Honey Rider in Dr. ...	Thursday March 19 1936	Swiss	Ostermundigen, Bern, Switzerland
Roger Federer		**Roger's** 16th grand slam title has secured him some space at the top of the ATP world ... Prince put in his place as **Federer** touches the sky (Sydney Morning ...	8 August 1981	Swiss	Basel, Switzerland
Martina Hingis		**Martina Hingis** (born 30 September 1980 in Košice, Slovakia, then Czechoslovakia) is a retired professional tennis player who spent a total of 209 weeks as ...	30 September 1980	Swiss	Kosice, Slovakia
Heidi		Fictive person in Johanna Spyri's story "**Heidi**"; on the web. World champion tennis player; Born September 30, 1980 in Košice (Slovakia), ...	June 01, 1973	German	Bergisch Gladbach, Germany
Leonhard Euler		**Leonhard** Paul **Euler** (15 April 1707 – 18 September 1783) was a pioneering Swiss mathematician and physicist who spent most of his life in Russia and Germany. ...	April 15, 1707	Swiss	Basel, Switzerland
Daniel Bernoulli		**Daniel Bernoulli** (Groningen, 8 February 1700 – Basel, 8 March 1782) was a Dutch-Swiss mathematician and was one of the many prominent mathematicians in the ...	January 29, 1700	Swiss	Gröningen, Netherlands
Johann Bernoulli		**Johann Bernoulli** (Basel, 27 July 1667 – 1 January 1748; also known as Jean or John) was a Swiss mathematician and was one of the many prominent ...	27 July 1667	Swiss	Basel
Auguste Piccard		**Auguste** Antoine **Piccard** (January 28, 1884 – March 24, 1962) was a Swiss physicist, inventor and explorer. **Piccard** and his twin brother Jean Felix were born ...	January 28, 1884	Swiss	Basel, Switzerland

Figure 3.1: Google Squared features the same and fairly generic attributes for all entities in the results set.

merely listing entity names, or overwhelm the user with the full set of facts that they find in their underlying repositories. For example, when you ask for "Swiss people", some of the above engines merely return a list of names. The user can click on each name to see more facts about the person, including Web pages that contain the entity, but this is a tedious way for knowledge discovery. Other engines show all – often hundreds of – facts about all Swiss people that DBpedia, Freebase, and other linked-data sources offer; this is a cognitive overload for most users. What we need instead is a kind of *semantic snippet* per result entity, highlighting the most salient facts about each but avoiding trivial or exotic information. For example, for the Swiss-people result Albert Einstein, we may want to see a compact summary saying that he was a scientist, won the Nobel Prize, was born in Germany (but grew up in Switzerland), graduated at the University of Zurich, and later was a professor at Humboldt University and even later at Princeton.

Google Squared (Fig. 3.1) does return attribute-structured records as answers to keyword queries – an adequate result granularity. However, the attributes are the same for each entity in the result set. For the Swiss-people query, Einstein is treated the same way as Roger Federer (a Tennis player): the presented attributes are fairly generic properties like birth date, birth place, death date, and death place. Another technique of tackling the problem of describing entities is to retrieve textual passages from the original source used to build the knowledge base [127, 41, 13], but these may not always be available.

3.1.2 Problem Statement

Explicit knowledge bases have very rich type systems, partly inferred from Wikipedia categories, the WordNet taxonomy, and other sources of this kind. For example, YAGO knows 38 semantic classes to which Einstein belongs.

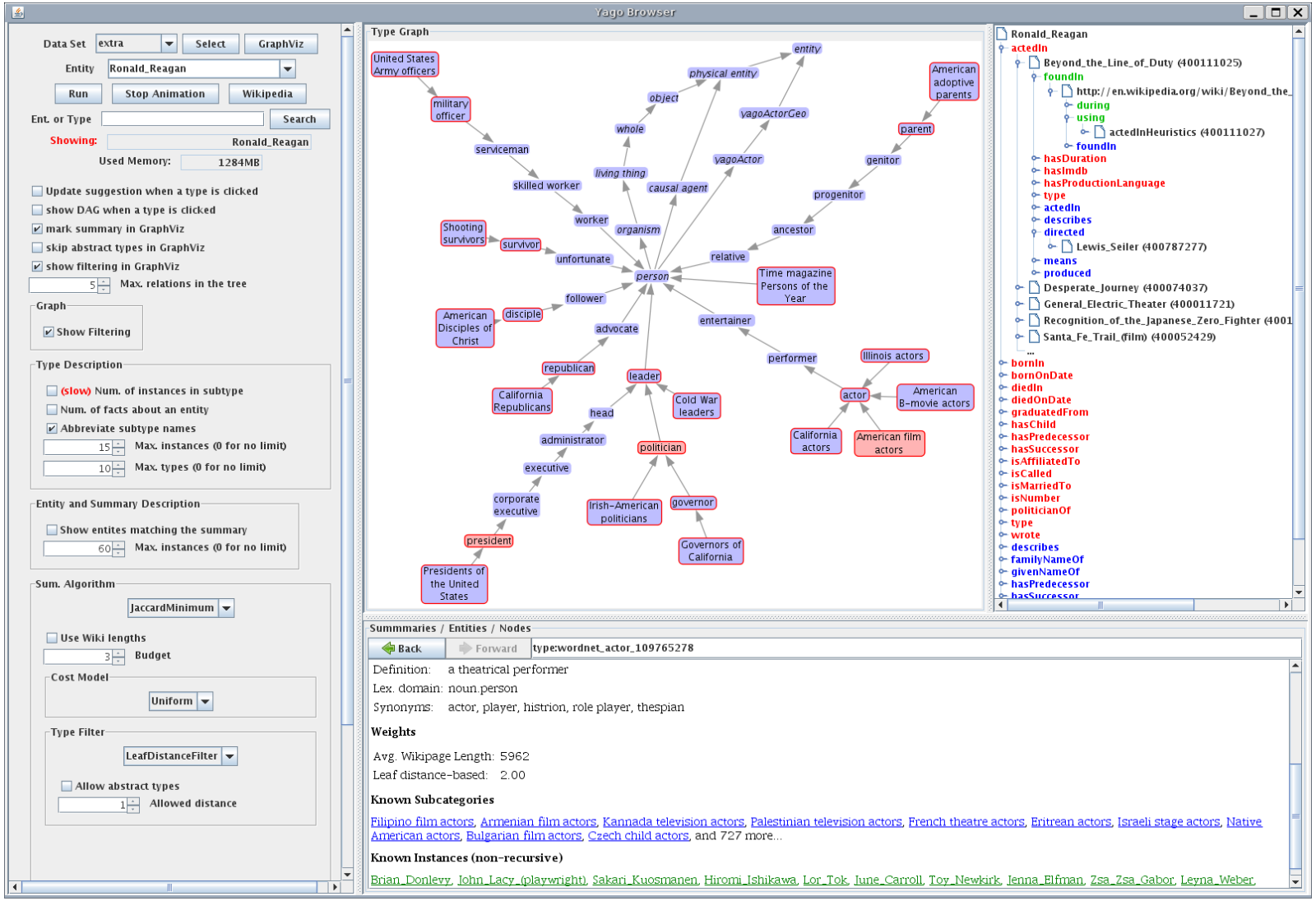


Figure 3.2: YAGO type graph for Ronald Reagan shown in our browser. *Left:* controls of the browser and summarization algorithms, *center:* the type DAG, *right:* relation view, *bottom:* additional information about types.

Additionally, the semantics of `type` relation augments the set of types of an entity with all the supertypes, e.g. a *physicist* has also the type *scientist* and *person* (see Fig. 3.3 for a detailed example). In case of the entity Albert Einstein the 38 types have more than 50 superclasses in the YAGO type system. Judiciously choosing a small subset of these could return more or less the ideal summary outlined above. Note that type/category names can often be viewed as encoding attribute values, such as class `NobelPrizeWinners` denoting the fact HASWON→ NobelPrize. Conversely, we can see groups of entities with the same value for some interesting attribute as a semantic type/class, such as BORNIN→ Germany defining the type `peopleBornInGermany`. Because of this duality, we restrict our approach for generating semantic snippets to selecting appropriate types from the rich repertoire available in the knowledge base.

In most search- and exploration-related situations, a good summary of an entity would contain between three and around ten properties. These could then be refined by explicit user interactions on specific subtype dimensions when needed. However, Yago and DBpedia have an order-of-magnitude larger number of types/classes to choose from. This is illustrated by the type graph for Ronald Reagan, as shown in a Yago browsing tool in Fig. 3.2. Note that the type graph of an entity is usually not a tree, but forms a directed acyclic graph (DAG), with subtype-supertype edges and a generic root type `entity`. For example, the type `mayor` has two supertypes: `politician` and `civil authority`, which converge in `person`.

The problem addressed in this chapter is the following: given the type DAG of an entity and the desired number of output types, select the most informative types. These types will then constitute a semantic snippet for semantic search results.

3.2 Design Considerations

In this section we first give a short description of how knowledge bases are organized and in particular how YAGO, which is the one we use for our purpose, represents facts. We then identify the properties of good type-based summaries.

3.2.1 The Knowledge Base

YAGO contains facts extracted from semi-structured parts of Wikipedia (infoboxes and categories) and WordNet [48]. A small excerpt is presented in Table 3.1. The named objects in YAGO can be classified into *entities* which represent persons, locations, organizations, etc., e.g. `Albert Einstein` and `University of Zurich` and *types* which form a categorization of entities, e.g. `physicist` and `Swiss physicists`.

In the task of summarization we are particularly interested in the relations `type` and `subClassOf`. The former is defined between entities and types, and the latter between types. Using the `type` and `subClassOf` relations we can build a direct acyclic graph representing all types of an entity. Fig. 3.4 presents such a DAG for Aimé Argand. The `subClassOf` relation is transitive ($a \rightarrow b \wedge b \rightarrow c \Rightarrow a \rightarrow c$), but for clarity we do not show transitive edges in type DAGs.

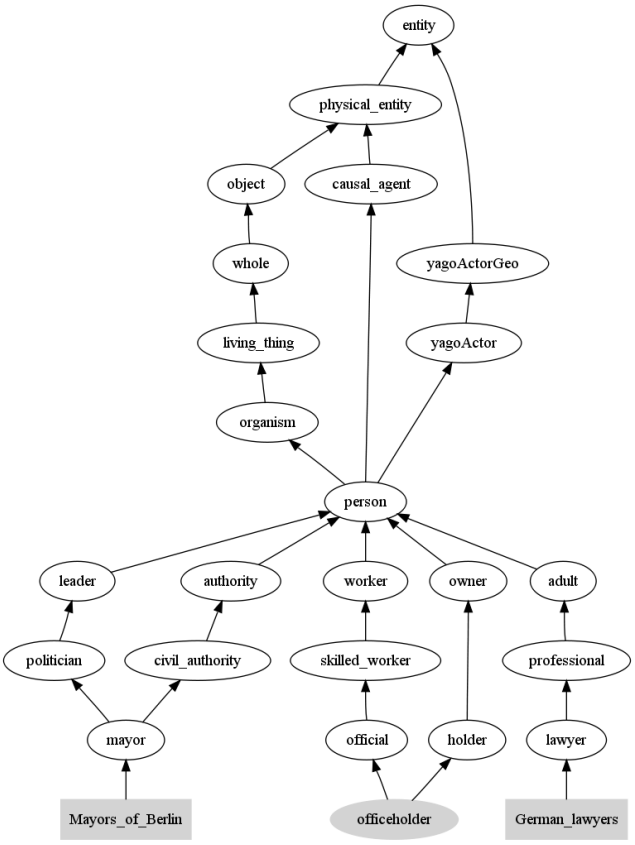


Figure 3.3: Direct acyclic graph (DAG) of types (entity: Max von Forckenbeck). Shaded nodes were extracted from Wikipedia, unshaded nodes are implied by *subClass* relation. Rectangular nodes come from Wikipedia, rounded from WordNet. Edges show the *subClass* relation.

Subject	**Predicate**	**Object**
Albert Einstein	bornIn	Ulm
Albert Einstein	bornOnDate	1879-03-14
Albert Einstein	graduatedFrom	University of Zurich
Albert Einstein	hasAcademicAdvisor	Alfred Kleiner
Albert Einstein	type	Swiss physicists
Swiss physicists	subClassOf	physicist
physicist	subClassOf	scientist
Wilhelm C. Röntgen	graduatedFrom	University of Zurich
...	...	...

Table 3.1: An excerpt from YAGO. Our summarization methods use `type` and `subClassOf` relations.

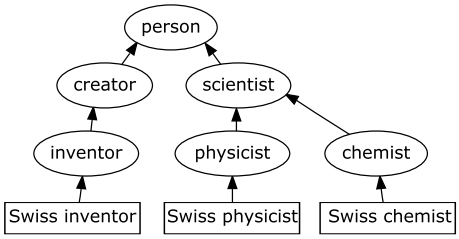


Figure 3.4: Type DAG for Aimé Argand. Rectangular nodes correspond to the Wikipedia categories and oval nodes correspond to WordNet synsets. Some nodes were removed for clarity.

3.2.2 Objectives of Summarization

We compiled the following list of desirable properties that any good summary should have.

Conciseness. A good summary should not be too long. Depending on how the summary is shown to the user, we consider two different measures of its size. If the types in the summary are presented in a list with each item occupying a single line, then we assign a unit cost to every type and enforce a simple cardinality constraint on the number of types that may be selected. If the types are presented as a list separated with commas and the total space occupied by such a list should be limited, then the cost of each type is defined to be equal to its length in characters.

Importance. There are types which carry valuable information about an entity, e.g. the type `physicist` is crucial when we talk about Albert Einstein. On the contrary, other types describe non-salient facts (`left-handed person`, `vegetarian`). Since our summaries are meant to provide users with the most significant facts, our method should favor the former kind of types. If instead we were generating trivia rather than summaries, it would make sense to prefer non-salient types.

Granularity. General types, which describe large categories of entities, e.g. `person` or `location` should not be included in the summary. Similarly, too specific types which describe a very small number of entities, e.g. `19th century physicist born in Zug` should not be included either.

Diversity. A summary should cover all aspects of an entity, e.g. {`member of the parliament`, `prime minister`} is worse than {`politician`, `physicist`}, because it focuses only on one aspect (political career) of the described entity.

A natural summarization method could serve the user with the top-ranked types according to some appropriate weighting function. Such a summary S_i for a given budget of i nodes would be a monotonic function in the sense of set inclusion, i.e. it would hold that $i \leq j \implies S_i \subseteq S_j$. However, we argue that this method would not deliver any good solution, as illustrated by the following example. Consider the task of summarizing the type graph in Fig. 3.4, while having only a limited budget on the number of types that can be selected.

Symbol	Meaning
T	all types of an entity
$\sigma \in T$	a type
$S \subseteq T$	summary
k	budget (types or characters)
$D = \langle T, E \rangle$	type DAG, $(a \to b \wedge b \to c \Rightarrow a \to c)$
$w : T \to \mathbb{R}_+$	weights of nodes (how good they are)
$c : T \to \mathbb{R}_+$	costs of nodes (1 or length)
$e : \sigma$	entity e has type σ

Table 3.2: Summary of notation.

Summaries that we find to be the best for cardinalities 1 to 3 are:

$$S_1 = \{scientist\},$$
$$S_2 = \{scientist, inventor\},$$
$$S_3 = \{inventor, physicist, chemist\}.$$

If the budget is only one type, we prefer to use the more informative `scientist` rather than too general `person`. When the budget is increased to two types, we expand the summary with the type `inventor`. However, when our budget consists of three types it is better to substitute `scientist` with `physicist` and `chemist`. In our example it is impossible to extend S_2 without introducing redundancy, since any type logically implies all its supertypes (e.g. if an entity is a `physicist` then it also has types `scientist` and `person`). In general, when the budget is increased we do not just add more types, but choose different types or split the ones which have already been selected (`scientist` is split into `physicist` and `chemist`). Hence, simply returning the top-ranked types to the user would not give satisfactory results.

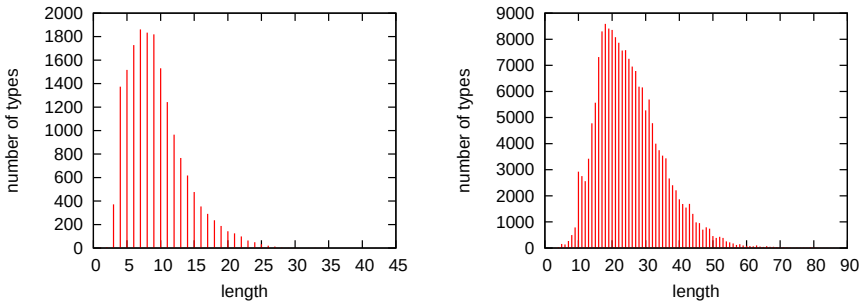
3.3 Summarization Methods

We propose two methods of generating type-based summaries. The first treats a type as a set of entities; a summary is built by finding a set of general types which are dissimilar. The second proposed method views types of an entity as a directed acyclic graph and finds the maximum weighted set of vertices subject to a redundancy avoidance constraint.

A crucial pre-processing step in the summarization is establishing the quality of available types. In order to allow tuning the specificity of types used in the summary we devised a PageRank-based method of weighting types. The second problem arising in our setting are types with missing qualifications. For example, YAGO knows the type `citizen`, which by itself is meaningless as opposed to a qualified version, such as `citizen of Germany`. We show how this problem can be alleviated by using simple frequency-based method. Quality and importance of types are also the subject of Chapter 4.

The requirement that a summary is small is captured by two possible constraints:

1. it has not more than k types (nodes),



(a) WordNet-derived types (b) Wikipedia-derived types

Figure 3.5: Distribution of type lengths (in characters) in YAGO. Modes of the distributions are 7 and 18.

2. the total length of all types in the summary is not more than k characters.

We will write $c(t)$ for the cost of including the type t in the summary. In our first model all types are given the same cost equal to 1, in the second case, the cost of a type is equal to its length in characters. This models the scenarios where types are displayed one per line or in a comma separated list, as discussed before. Fig. 3.5 presents distributions of type name lengths in YAGO. Summary of the notation is presented in Table 3.2.

3.3.1 Intersection-based Method

The first method for summarization simply treats a type as the set of all entities that it describes. For example, *physicist* will correspond to the set of all physicists known to the knowledge base. We write $e : \sigma$ if the entity e has type σ. The principle behind our summarization approach is the following problem: a user is given a summary and the task to guess the entity. We assume that the user knows all possible entities and their respective types, as well as the `subClassOf` relation. The probability that he guesses correctly is

$$Pr[correct|\sigma_1, \ldots, \sigma_k] = \frac{1}{|\{e|e : \sigma_1 \wedge \ldots \wedge e : \sigma_k\}|}.$$

That is, 1 divided by the number of entities that can have the given summary. For instance, if the summary is {*president of the US, member of the Republican Party*}, then the probability is 1 divided by the number of the presidents from the Republican Party. Stating our goal in such way leads to the following optimization problem:

$$\begin{aligned} \min_{S \subseteq T} \quad & |\bigcap S| \\ \text{s.t.} \quad & \sum_{\sigma \in S} c(\sigma) \leq k \end{aligned} \tag{3.1}$$

where the types are treated as sets of entities that they describe. We will refer to the summarization method based on the above problem as Min. Intersection.

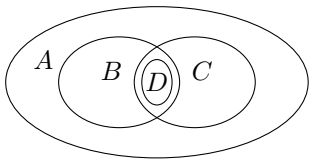


Figure 3.6: Summarization with types as sets: Jaccard coefficient is minimized by $\{A, D\}$, but we prefer $\{B, C\}$ (the budget is two sets).

Solutions to (3.1) often contain peculiar types with a small number of members. In order to overcome this issue, we require additionally that the summary should not only have a small intersection, but it should also consist of large sets. We could solve this problem directly, by imposing a constraint on the size of sets:

$$\begin{aligned} \min_{S \subseteq T} \quad & |\bigcap S| \\ \text{s.t.} \quad & \sum_{\sigma \in S} c(\sigma) \leq k \\ & |\sigma| \geq n \quad \forall \sigma \in S \end{aligned} \tag{3.2}$$

We can go further and filter sets not only by their size, but also based on the user's knowledge, that is we allow small sets if we know that the user knows them, such an approach would yield *personalized summaries*. Clearly, every algorithm that solves (3.1), can be applied to (3.2) after pruning the types, which violate the constraint on size.

To avoid choosing the parameter n for the cut-off we can incorporate type sizes in the objective function by minimizing the Jaccard coefficient instead of the intersection:

$$\begin{aligned} \min_{S \subseteq T} \quad & \frac{|\bigcap S|}{|\bigcup S|} \\ \text{s.t.} \quad & \sum_{\sigma \in S} c(\sigma) \leq k \end{aligned} \tag{3.3}$$

Minimization of the Jaccard coefficient can, however, lead to an imbalanced solution as shown in Fig 3.6. Including a small and a large set in the solution leads to small intersection, due to presence of the small set, and large union, due to the large set. We solve this problem by modifying the Jaccard coefficient in a way, which prefers balanced set sizes. Instead of using the union in the denominator, we choose the size of the smallest (that is, the most specific) type in the solution, thereby constraining all sets in the solution to be general. The final formulation of our problem is as follows:

$$\begin{aligned} \min_{S \subseteq T} \quad & \frac{|\bigcap S|}{\min_{\sigma \in S} |\sigma|} \\ \text{s.t.} \quad & \sum_{\sigma \in S} c(\sigma) \leq k \end{aligned} \tag{3.4}$$

In this way given an entity with the type set $\mathcal{T}$ we choose subset $\mathcal{S}$ of $\mathcal{T}$, such that: *i*) its intersection is small, and *ii*) the types are of medium cardinality.

The first property gives us the diversity of the summary, e.g. if a person is both a musician and scientist, we will choose types which describe each aspect of their career. The second one ensures that the types we choose are not too general (e.g. `person`) or too specific (e.g. `German quantum physicist`). We will refer to to the summarization method based on (3.4) as Mod. Jaccard.

We solve the minimum intersection problem (3.1) with the greedy algorithm presented in Alg. 1. We build the solution by accumulating sets which remove most elements per cost unit from the current intersection I. The optimal set at each step of the greedy algorithm is selected in line 9. We make sure that we do not add redundant sets to the solution by testing in line 12 if the set removes at least one element from the intersection. To solve the Mod. Jaccard problem we iterate over cardinalities of available types $s \in \{|T| : T \in U\}$ and solve the Min. Intersection on U restricted to sets of size at least s. We present the full algorithm in Alg. 2.

Algorithm 1: Greedy algorithm for the minimum intersection problem.

```
   Data: U – family of sets, b – budget, c(·) – cost function
 1 Proc MinIntersection(U,c,b)
 2 begin
 3     Solution := ∅
 4     I := ⋃U
 5     cost := 0
 6     U := {S ∈ U : cost + c(S) ≤ b}
 7     while U ≠ ∅ do
 9         T := arg max_{S∈U} |I\S| / c(S)
10         U := U \ {T}
12         if I \ T ≠ ∅ then
13             Solution := Solution ∪ {T}
14             cost := cost + c(T)
15             I := I ∩ T
16         end
17         U := {S ∈ U : cost + c(S) ≤ b}
18     end
19     return Solution
20 end
```

3.3.2 Graph-based Method

Our second formulation of the summarization task casts it as an optimization problem on a graph. We work with graphs such as Fig. 3.4 and Fig. 3.3. First, we assign to types positive weights $w : T \to \mathbb{R}_+$, which reflect how well they describe the summarized entity (good nodes have high values). Then we select a maximum set of vertices that does not exceed the budget allocated for the summary and whose elements are not redundant.

Algorithm 2: Algorithm for the Mod. Jaccard problem.

Data: U – family of sets, b – budget, $c(\cdot)$ – cost function
Proc `ModJaccard(`U,c,b`)`
begin
  $Solution := null$
  $objective := -\infty$
  for $s \in \{|T| : T \in U\}$ **do**
    $U_s := \{T \in U : |T| \geq s\}$
    $MI :=$ `MinIntersection(`U_s,c, b`)`
    if $|MI|/s < objective$ **then**
      $objective := |MI|/s$
      $Solution := MI$
    end
  end
  return $Solution$
end

Type Weights

It is reasonable to assume the following about types on a directed path in a graph: the weights first increase until they reach the maximum for the path and then decrease. This corresponds to going from very specific types to the best one and then to too general ones, e.g. `quantum physicist` $\rightarrow$ `physicist` $\rightarrow$ `scientist` $\rightarrow$ `person`.

In order to find appropriate weights for types, we exploit the fact that the entities in YAGO were extracted from Wikipedia and therefore we can easily obtain the lengths of their main articles (in bytes). We use the average article length as a proxy for importance of a type. This works well for types which are equivalent to categories in Wikipedia. YAGO contains also types that originate from WordNet (see Fig. 3.3). However, since almost all leaf types of entities are derived from Wikicategories, we can easily propagate the weights along the `subClassOf` edges to the internal WordNet types. We compiled the following list of properties, which describe how the weights should be propagated in the type DAG:

1. The types low in the hierarchy should have large weights as they are more precise. Abstract types should be assigned low weights.
2. If a type has multiple children (in a type DAG of a single entity) its weight should be amplified, e.g. in Fig. 3.4 the weight of *scientist* should be boosted, because it has two children, which shows that it is more important for the entity.
3. A single parameter should control the trade-off between choosing types low and high in the hierarchy,

To satisfy the above desiderata we developed a weight propagation method, which is based on a random walk on the type DAG. The walk starts at a leaf with

probability proportional to the average length of its Wikipedia articles (denoted by $awl(\cdot)$). At each node we *i*) either restart the walk (jump to a leaf) with probability α (or 1 in sink nodes), *ii*) or, with probability $(1-\alpha)$, we follow one of the outgoing links of the node. Since at the sink nodes we cannot follow outgoing links, the probability of jump is 1. The probability that we are in node v is

$$P(v) = P_{jump}(v)\left(\alpha \sum_{w \text{ not a sink}} P(w) + \sum_{w \text{ sink}} P(w)\right) + (1-\alpha) \sum_{w \in children(v)} \frac{P(w)}{outdeg(w)}$$

where

$$P_{jump}(v) = \begin{cases} awl(v)/\sum_w awl(w) & \text{if } v \text{ is a leaf node,} \\ 0 & \text{otherwise.} \end{cases}$$

In this way, weights are propagated along directed paths in the graph and are amplified where two or more paths converge.

Note that the weights of the Wikicategories are calculated independently of the entities they describe; `Nobel laureates in Literature` has equal weights for Ernest Hemingway and Winston Churchill, who is better known as a British Prime Minister, but also received a Nobel Prize for Literature. Our propagation method will however give more weight to internal nodes of the type DAG, if they have many descendants. For example, Albert Einstein will have a large subtree of types rooted in `scientist`, which contains types such as `theoretical physicist` and `cosmologist`. Therefore, the weights of the internal nodes in the graph are not independent of the entity, which is a desirable property.

In addition to finding too general, too specific, or unimportant types we also need to recognize types which lack an argument, e.g. `citizen` versus `citizen of Germany`, or `member` versus `member of the European Union`. The problem was already discussed in [75], but their results are applicable mostly to the Japanese language. In our system we use a simple frequency-based method to find types with missing arguments. We calculated the total number of occurrences of the noun, and the number of occurrences with an argument pattern like: *<noun> of, 's <noun>, (his|her|their|its) <noun>* in a 5-gram corpus generated from ClueWeb09 collection (provided by Ralf Schenkel). The ratio of the two values is used to decide whether an argument is needed. If a type requires an argument it is removed from the type DAG in the pre-processing step before running summarization on an entity.

Finally, YAGO provides a list of types which are considered very abstract. Such types are located close to the root of the type hierarchy and divide it into broad categories such as `living thing`, `object`, and so on. We simple remove them from the type DAG in pre-processing.

Optimization Problem

Our graph-based formulation of summarization problem is based on the maximum independent set problem. Given a graph $G = \langle V, E \rangle$ a set of vertices $U \subseteq V$ is independent (also called stable) if and only if no two vertices from U are connected by an edge, i.e. $\forall u, v \in U : (u, v) \notin E \wedge (v, u) \notin E$. The problem can be generalized to maximum weight independent set (MWIS) by

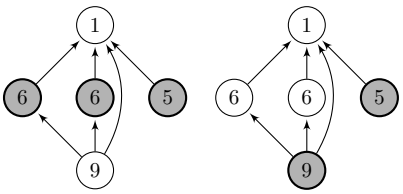


Figure 3.7: Type summarization with k-MWIS. Shaded nodes represent optimal summaries of size 3 and 2.

assigning weights $w(\cdot)$ to the vertices in the graph. The maximum independent set problem is NP-hard [56].

Relation `subClassOf` has transitive semantics, therefore we assume below that the type DAG is transitive ($a \to b \wedge b \to c \Rightarrow a \to c$). Since a type always implies all its supertypes, a summary should not contain a pair consisting of a type and its supertype, therefore it must be an independent set in the transitive type DAG. YAGO stores the type DAGs without the transitive closure. An independent pair of vertices in a transitive graph, corresponds to a pair of vertices which are not connected by a directed path in the transitively reduced graph.

An optimal summary will have a maximal weight among all summaries obeying the constraints on the size. An example is presented in Fig. 3.7. Formally, we solve the following optimization problem:

$$\begin{aligned} \max_{S \subseteq T} \quad & \sum_{\sigma \in S} w(\sigma) \\ \text{s.t.} \quad & \sum_{\sigma \in S} c(\sigma) \leq k \\ & S \text{ is independent} \end{aligned} \tag{3.5}$$

Nodes are assigned weights as described in the earlier paragraphs. The summarization algorithm chooses a subset of types of an entity, such that: *i*) the cost of the set is smaller than the budget, *ii*) the sum of weights is maximized, *iii*) no two types are connected by a directed path. The last requirement ensures that we do not choose redundant types, e.g. `quantum physicist` and `scientist`. The optimization problem itself does not have any tunable parameters, but the weighting method that we used has one parameter which governs the balance between general and specific types. Hence we are able to tune the method to our preferences.

We were neither able to prove NP-completeness of above problem, nor give a polynomial time algorithm to solve it. In general the maximum weighted independent set problem (MWIS) is NP-complete. Our graphs are however *transitive* and therefore MWIS can be solved in polynomial time [57]. The algorithm cannot however be applied to the problem with the cardinality constraint ($|S| \leq k$). On the other hand, if all weights and costs are equal, the problem stays in P (take k vertices from the solution to MWIS). Furthermore, if the costs are uniform and the constraint on cardinality is transformed into a penalty term and added to the objective function, we obtain an instance of MWIS problem, which is again solvable in polynomial time. Yet another possibility is to restrict

the class of graphs to $transitive \cap cographs$, which can be characterized by absence of induced P_4 [1]. The problem can be solved then in polynomial time, using modular decomposition of graphs. The restriction is however too strong – there can be type graphs which contain induced P_4.

Our implementation solves the problem exactly using JaCoP [73] constraint programming library. We discuss the software architecture in Section 3.5.

3.4 Ontology Browser

To demonstrate our approach to summarization and exploration of knowledge bases we developed a browser for YAGO. The browser is entity- and type-centric, that is, the user browses entities one at time and the types are in the center of the interaction. We present the entity relation graph in a standard user interface tree view, which offers intuitive navigation, easily expands and collapses parts of the graph, and uses little screen space. Finally, we implemented the summarization algorithms within the browser, which facilitates interactive evaluation of the summarization algorithms.

3.4.1 Browser Interface

The main window of the browser is shown in Fig. 3.2. In the beginning the user has to choose an entity by selecting it from a list, entering its unique identifier, or by using the built-in search box. The application shows the type DAG of the entity in the center of the screen, its relations in the tree on the left, and additional information in the bottom.

Relation View

Let us look at the relation view in the example (Fig. 3.2). The top node is the entity selected by the user. Its direct children are the relations and the next level contains their arguments, e.g. Ronald Reagan –ACTEDIN→ Beyond the Line of Duty. The structure is recursive, that is the children of Beyond the Line of Duty are the relations, which have it as an argument. All entities and values are show in black. The relations can be red, blue or green. Red relations should be read top-down, e.g. Ronald Reagan –ACTEDIN→ Beyond the Line of Duty, blue relations have the opposite direction, e.g. Lewis Seiler –DIRECTED→ Beyond the Line of Duty. Green relations describe metafacts (facts about facts). Each fact in YAGO has a unique identifier, e.g. 400111025: Ronald Reagan –ACTEDIN→ Beyond the Line of Duty. The metafacts use these identifiers to refer to facts they describe, e.g. 400111025 –FOUNDIN→ en.wikipedia.org/wiki/Ronald_Reagan.

The way in which we present the relations has several advantages over graph views, similar to the one used for type DAGs. First, it is easy to implement in various programming environments, since it uses only the standard tree widget. Moreover, the users find it easier to use familiar user interface elements also used in other applications than a custom graph view. The second advantage is that tree views efficiently use screen space and allow the user to easily expand nodes and collapse non-interesting parts of the graph.

[1] P_4 is a path on 4 vertices: $a \rightarrow b \leftarrow c \rightarrow d$ or $a \leftarrow b \rightarrow c \leftarrow d$

The relation view part of our browser is knowledge base agnostic, that is, it does not make any assumptions about the data. We on require that the representation has the subject-predicate-object form and the facts are assigned identifiers to allow meta-facts (facts about facts). Even the fact identifiers and meta-facts are optional. The interaction with relation view can be easily duplicated with a different knowledge base.

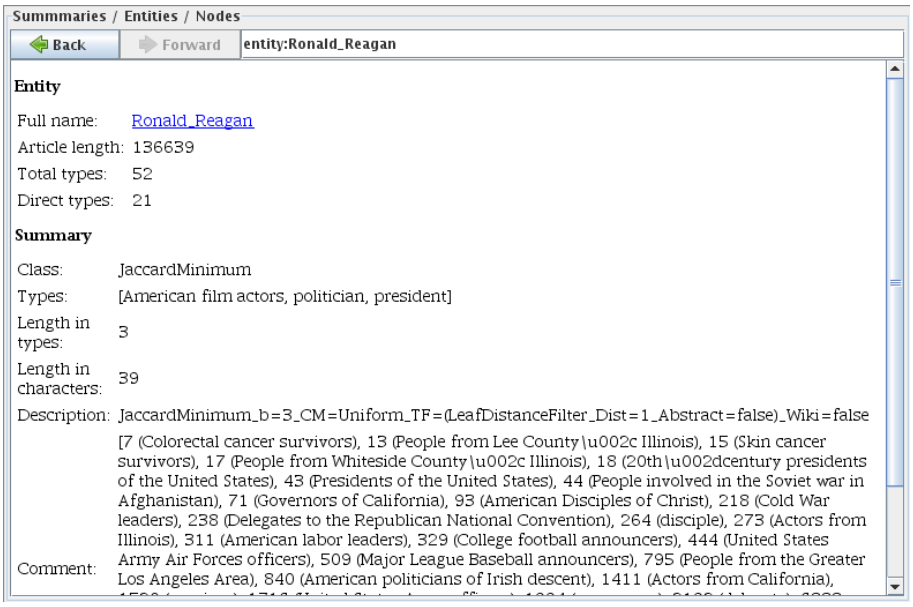
Contextual Information

In contrast to the relation view, the contextual information presented in the bottom box of the browser is mostly specific to YAGO. The information presented is basically a short report about an entity or a type. At the beginning of the interaction the box displays information about the current entity. This can be changed by clicking on a type in the center of the browser, or selecting a different entity, for example in the relation view. Reports about entities (Fig. 3.8(a)) contain statistics calculated from YAGO, such as the number of types of the entity, and from external sources, such as the article length in Wikipedia. We also include the summary of the entity obtained with currently selected summarization method and debugging information useful for improving summarization. Contextual information about types (Fig. 3.8(b)) contains statistics from YAGO, such as the number of subtypes, additional data for WordNet-derived types, e.g. their definitions, and the lists of sample subtypes and instances. The contextual information box functions much like a Web browser, all types and entities presented in blue, underlined font are clickable. For example, clicking on a sample entity of a type moves the user to the report about the entity. The chain of visited reports is stored and can be navigated using the *back* and *forward* buttons, exactly like in a Web browser. The architecture of the browser makes it easy to add new information to the reports.

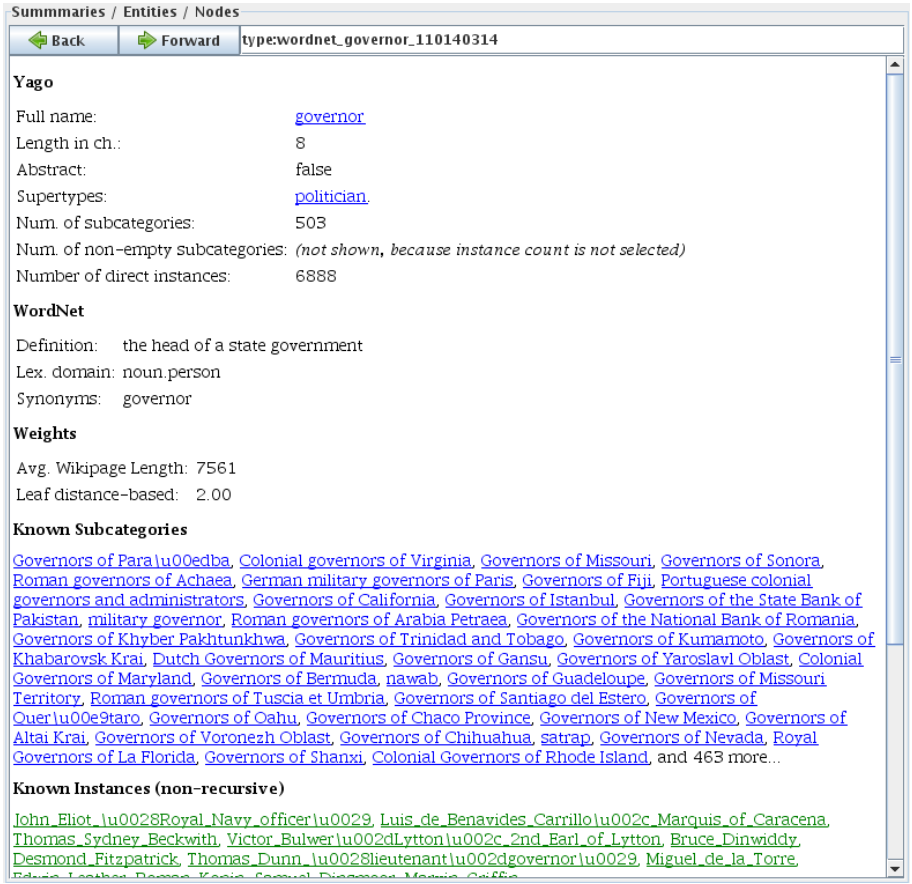
3.4.2 Summarization

The browser shows the results of summarization of an entity on the type DAG. The panel on the left controls which summarization algorithm is to be used. The user can switch between no summarization, the set-based algorithm, and the graph based algorithm. The interface allows to select a cost model based on the number of types or their lengths, as well as some additional parameters, e.g. the trade-off between specific and general types in the second algorithm. A sample summary is shown in Fig. 3.2. The nodes with the red outline were selected in the pre-processing phase and passed to the summarization algorithm. Subsequently, the summarization algorithm chose the summary, which consists of the red nodes.

In the example in the Fig.3.2 we requested a summary of the length at most 3 types. The filtering stage used in this example allows only the nodes which are leaves or parents of leaves. The summarization algorithm chose three types: `president`, `politician` and `American film actors`. Clearly, the summary includes the most important types – `president` and `politician`. It is diverse, because it mentions that Ronald Reagan was an actor, which is a valuable fact, as not many politicians are also actors. Additionally, we avoided selecting uninteresting types like `American adoptive parents` and too general ones like `leader`.



(a) Entity Context. The report contains basic information about the types of the entity, the length of its article in Wikipedia, and detailed information about the summary.



(b) Type Context. The report contains statistics related to YAGO, such as the number of subtypes, WordNet data, such as type definition, and lists of sample subtypes and entities.

Figure 3.8: Contextual Information in the Ontology Browser.

3.5 Software Architecture

Our ontology browser is implemented in Java programming language, which allows portability between Linux and Windows environments, as well as running the application conveniently within Web browser as an applet. The YAGO knowledge base is stored in PostgreSQL database on a server. The ontology browser retrieves data from the server only when it is needed. To decrease communication cost further we cache already retrieved data for future use.

To facilitate easy development of summarization algorithms our Ontology Browser defines a Java interface that classes providing summarization methods should implement. The tunable parameters are marked with Java annotations (form of metadata attached to the code). Our browser analyzes the annotations at run-time to generate user interface elements that manipulate the parameters of the summarization method, for example, Cost Model and Type Filter panels in Fig 3.2 are generated at run-time from the annotations in the code.

The graph-based summarization method is implemented using JaCoP [73] constraint programming library. Constraint programming frameworks such as JaCoP, Gecode [123], finite domain solver in GNU Prolog [38], and others offer a high level language for expressing complex combinatorial problems and solving them in an efficient way. An alternative is to use integer linear programming (ILP) and translate the constraints into numeric inequalities. See [21] for practical examples of encoding logical constraints in ILP. Since our instances of k-MWIS are small the performance was not an issue and we chose JaCoP for two reasons. First, constraint programming allows us to model problems using high level constructs such as graph, sets, and logical formulas, which leads to code both easy to understand and modify. The second reason is that virtually all Java solvers for integer linear programs, for example Gurobi [59], interface libraries written in C/C++ and compiled to native machine code (instead of portable Java bytecode). A notable exception is the Oj! Algorithms library (`ojalgo.org`). While such approach leads to better performance, it makes programs less portable and is a significant obstacle when developing applets for running within Web browser or servlets hosted on platforms such as Tomcat (`tomcat.apache.org`). The fact spotting problem described in Chapter 5 also resulted in an optimization problem. In order to develop pure Java system, we shortly tried ILP solver from the Oj! Algorithms library, and, since our problem could be formulated as a maximum satisfiability, also the Sat4j library (`www.sat4j.org`), which is used for library dependency management in Eclipse. We did not, however, follow this approach, since the results we obtained did not match Gurobi in terms of speed or quality of approximate solutions.

3.6 Evaluation

We evaluated our summarization methods on a set of 61 entities known to YAGO. The set contains entities from different domains such as politicians, artists, companies, etc. Our summarization methods support two cost models proposed earlier: uniform cost for all types and variable cost representing space occupied by a type in the summary, which in our case is just the length of the

type name in characters. We therefore evaluated our algorithms in two settings: the total size of the summary can be at most 3 types or 45 characters.

Measures. To measure the quality of the summaries we assessed the following three aspects, which reflect our desiderata from Section 3.2.2:

1. diversity,
2. importance,
3. granularity.

In order to measure diversity we count the number of non-redundant types. For example, in the summary *physicist, scientist, politician* only 2 out of 3 types are non-redundant, since *physicist* implies *scientist*. While redundancy in our algorithms means that one type is a subclass of the other, in the evaluation we treat also other closely related types as redundant. For the importance we report how many types in the summary carry valuable information about the entity. Finally, to evaluate granularity we count how many types are not too specific or too general. To make the values comparable across different entities, we divide them by the total number of types in the summary.

Baseline Methods. We compared our proposed summarization methods against two baseline approaches. The first natural baseline for summarization is to order the types by importance and serve the user with top-k most important types. In this case we use our type weighting based on average article length in Wikipedia to measure the importance of types. The second baseline approach is the method based on the minimum intersection (problem 3.1).

Results. The results of our experiments are presented in Table 3.3. We ran the two baselines described above (Top-k and Min. Intersection) and two proposed methods: k-MWIS which is the graph-based summarization solving the optimization problem (3.5) and modified Jaccard coefficient (Mod. Jaccard), which solves (3.4). We report the measures described above – diversity, importance, and granularity averaged over our test set of 61 entities. Since all of them are crucial for a good summary we also report their sum (D+I+G). Good granularity of types is to large extent obtained by careful selection of types which are fed to the summarization algorithm. Therefore we also report the sum of only the diversity and importance, which takes the pre-processing component out of the picture.

Our proposed methods compare favorably against the baselines in terms of the proposed measures. They achieve the two highest combined scores for both types of budget constraints. The worst performer is the natural Top-k baseline, which falls short especially in terms of diversity and importance of types included in the summary. On the other hand the Min. Intersection method, which is a simplistic version of Mod. Jaccard, achieved the best diversity on 3-type-long summaries. It is also the second best method when we combine diversity and importance on 45-character summaries.

Method	Div.	Imp.	Gran.	D+I	D+I+G	#Types
BUDGET = 3 TYPES						
k-MWIS	0.91	0.80	**0.98**	1.71	2.69	2.92
Mod. Jaccard	0.94	**0.91**	0.89	**1.86**	**2.75**	2.85
Min. Intersection	**0.96**	0.75	0.92	1.70	2.62	1.90
Top-k	0.74	0.79	0.97	1.53	2.49	2.98
BUDGET = 45 CHARACTERS						
k-MWIS	0.92	0.76	**0.93**	1.69	2.62	2.59
Mod. Jaccard	**0.96**	**0.88**	0.88	**1.84**	**2.72**	3.97
Min. Intersection	0.93	0.85	0.80	1.77	2.58	3.51
Top-k	0.81	0.78	**0.93**	1.59	2.52	2.45

Table 3.3: Evaluation of summarization algorithms. We report scores for diversity, importance, and granularity of summaries averaged over a set of entities.

We also report the number of types in summaries. In case of summaries with at most 3 types the averages are below 3 for two reasons: for some entities number of available types is fewer than 3 or the optimization method achieves the optimum with fewer than 3 types. The latter phenomenon shows particularly strongly for the Min. Intersection method, which, by choosing 2 exotic types, can obtain an intersection (3.1) with exactly one entity. The Min. Intersection and Mod. Jaccard methods use on an average more types when the budget is constrained in characters. The methods then prefer types derived from WordNet, which are typically shorter than the Wikicategories in leaves of the type DAGs (see Fig. 3.5).

Sample summaries obtained by our methods are presented in Table 3.4. They provide examples for the general results described above. We can see that the baseline Min. Intersection chooses two exotic types in the summary of Tina Turner. The other baseline method, Top-k, produces summaries lacking diversity: in case of Clint Eastwood it focuses on the awards he received and in case of Donald Knuth chooses a general type *laureate* as well as its special case. Our proposed Mod. Jaccard method yields diverse summaries containing informative types. For example, the summary of John McCain contains information that he was a prisoner of war, is a senator and a writer. Some of the types in the summaries may not be sufficiently informative or can even mislead the users. For example, *exile* appears in YAGO type graph as a supercategory of *expatriate*, when the entity has a category such as *Canadian expatriates in Romania*. This can be traced back to YAGO's extraction algorithms and can be treated as a issue related to the knowledge base itself, which is hard to fix in a summarization method.

Method	Entity	Summary
	BUDGET = 3 TYPES	
k-MWIS	White Witch	Fictional dictators, Fictional immortals, Fictional mass murderers
Mod. Jaccard	John McCain	prisoner, senator, writer
Min. Intersect.	Tina Turner	American people of Native American descent, People from Haywood County Tennessee
Top-k	Donald Knuth	Computer pioneers, Grace Murray Hopper Award laureates, laureate
	BUDGET = 45 CHARACTERS	
k-MWIS	Donald Knuth	Computer pioneers, Living people, Typographers
Mod. Jaccard	Lars Hirschfeld	The Football League players, exile, goalkeeper
Min. Intersect.	Barack Obama	lawyer, narrator, senator, writer
Top-k	Clint Eastwood	Fellini Gold Medalists, Kennedy Center honorees

Table 3.4: Sample summaries obtained with the baseline and proposed methods.

CHAPTER 4

Salience of Types

In Chapter 3 we developed methods for generating type-based summaries of entities. An important part of the task was assessing the usefulness of types that were considered for the summary. Possible problems included types being too specific, too general, missing qualifications (*member* instead of *member of Parliament*), or not being central to the summarized entity (*Albert Einstein* is *a vegetarian*). In this chapter we discuss in more details the problem of such uninformative types and show how it can be solved in practice. First, we show our experiment with human computing for assessing usefulness of types, and then demonstrate how the history of changes in Wikipedia can be utilizes for obtaining importance of types with respect to an entity.

4.1 Properties of Good Types

The summarization approaches presented in Chapter 3 treat types as nodes in a graph or sets of entities. They work under the assumption that the types are all reasonably good candidates for inclusion in the summary of an entity. In practice this is, however, not always the case. Therefore, we apply a pre-processing step to filter out types which:

- are too specific, e.g. `alumnus of Quarry Bank High School`,
- are too abstract, e.g. `person`,
- have a missing qualification (argument), e.g. `advocate` instead of `advocate of gun control`.

In addition to recognizing the above properties, which make types unconditionally unsuitable for summarization, it is also important to understand which types are most crucial for an entity, for example, that Albert Einstein is more a physicist than a vegetarian.

In the previous chapter we used statistics available in a large n-gram corpus to prune types with missing qualifications (arguments), and used the average article length to measure the importance of Wikipedia categories and their equivalent types in YAGO. Here we explore two other ways of assessing whether types should be included in a summary:

Entity	Good Types	Bad Types
Al Gore	senator, environmentalist	alumnus, communicator
Van Morrison	guitarist, songwriter	creator, person
John McCain	politician, aviator	ancestor, child
J. R. R. Tolkien	writer, military officer	peer, professional
Mike Gravel	legislator, soldier	articulator, holder

Table 4.1: Examples of informative and not informative types found in YAGO.

- outsourcing the problem to a human computing platform, such as Amazon Mechanical Turk, and
- mining the importance of types from the history of changes in Wikipedia.

The advantage of the first approach is that while the corpus statistics can tell us that there is no problem with a missing argument, the type may still be uninformative to a user for other reasons. In case of average article length, we noticed that some non-crucial types, such as `vegetarian` or `left-handed person` can have long articles. We hypothesize, that this occurs because famous people are added to all possible categories, whereas not-so-famous ones are only added to the most important ones.

The types in YAGO come from two sources: WordNet synsets and Wikipedia categories (see Fig. 3.3). YAGO uses only around 6000 WordNet synsets, which makes it feasible to evaluate their usefulness on a human computing platform. The situation is different in case of types derived from Wikipedia; there are hundreds of thousands of them, so an automatic evaluation is necessary. Additionally, the problems which we face with those types are different. WordNet synsets can have missing arguments or be too abstract, which is something where human judgments are useful, whereas the problem with types low in the hierarchy is that they may not be very relevant with respect to the summarized entity.

4.2 Human Computing Method

Table 4.1 presents a sample of 5 well-known entities and their selected WordNet-based types. For each entity we chose two types which describe them well and two which would not be be useful in a summary. While it is easy for a human to evaluate which types are informative and which are not, producing a list of criteria that could be used to evaluate them automatically is a challenging task. Our observation that YAGO uses only around 6000 types derived from WordNet, makes using a human computing a feasible approach to evaluating quality of such types.

Mechanical Turk

Human computing has become an important technique for addressing various tasks including evaluation in information retrieval [4], image tagging [3], etc. The popularity of the approach has drawn attention from industry, which offers

multiple platforms connecting requesters with people willing to solve tasks in exchange for monetary compensation. Among the best known are CrowdFlower and Amazon Mechanical Turk, which we used in our experiments.

Tasks on Amazon Mechanical Turk (MTurk) are small questionnaires consisting of a description and a set of questions, which can be single-choice, multiple-choice, or open (textual input). We present a questionnaire used in our experiments in Fig. 4.2. Amazon Mechanical Turk offers three ways of creating tasks. In the simplest case they can be built using a simple Web interface. For applications demanding more flexibility a set of more powerful command line tools is provided; they allow specifying constraints such as "exactly 2 out of 5 checkboxes must be marked." Finally, it is also possible to develop applications in a general purpose programming language such as Java or C# with the MTurk SDK (software development kit). We used the first option since it is easy to use and we did not need any additional features. In case of questionnaires prepared with the Web interface requesters prepare a template using special syntax to mark parts of the text to be substituted. Subsequently, they upload a comma separated values (CSV) file, which contains the substitutions. The results can be downloaded from the MTurk web page in a similar CSV file.

Informativeness of Types

Initially, we intended to provide turkers (workers on Amazon Mechanical Turk platform) with a list of properties which a good type should satisfy and ask them to assess them according to these criteria. This, however, proved to be quite difficult. First, it is not straightforward to provide exhaustive, accurate, and concise criteria for our task. Second, turkers often solve tasks simultaneously with other activities (e.g., watching television) and they still should be able to provide valuable output. Therefore, we decided to follow an indirect approach. Since our types will be basically used to build sentences like *<entity> is a <type>*, we asked directly whether such sentences make sense. For each type we needed a good example, preferable a famous entity known by everyone. The criterion we used was the length of the Wikipedia page, i.e. for each type we chose the entity with the longest Wikipedia page.

We performed several rounds of experiments and analyzed the output to find the best way of asking the question. The final version of our questionnaire is presented in Fig. 4.2. It includes a short explanation of the purpose of the experiment, examples of correct solution, the task to be solved, and a field for feedback. The values which were changed are the sample entity and the type. For comparison we also present in Fig. 4.1 an earlier version of the questionnaire, which features a more detailed set of questions.

While we are aiming at dividing types into two categories like in Table 4.1, we subdivided good types into two additional categories *very good* and *correct, but not salient*. This natural subdivision helps avoid a confusing situation when examples like *Winston Churchill was a painter* or *Albert Einstein was a vegetarian* are treated in exactly the same way as *Albert Enstein was a physicist*, which may seem odd to people solving our tasks. We additionally provide *not sure* answer and a field for feedback, which is a standard good practice in human computing. In order to detect dishonest users who click at random answers, we

Assess the Quality of a Description

We are developing a system which generates short descriptions of entities such as people, locations, etc., for instance: **"Albert Einstein is a physicist"**. You will be presented with such a description. Additionally, you will see the definition of the category used in the description, its synonyms and a link to the Wikipedia page about the entity. Your task is to evaluate the quality of the description.

- You are **NOT** asked to verify whether the information is correct, but if you know that it is wrong, please mark the appropriate box in the second question.
- The grammar is **NOT** important (e.g. past vs present tense, singular vs plural, a vs an).

Should you have any troubles understanding the task, please write a comment in the box at the bottom of the survey.

The Task

Description: Universal Serial Bus is a public transport.
Synonyms:
Definition: conveyance for passengers or mail or freight

1. Grade the quality of the description on the scale from 1 (poor) to 5 (perfect):

○ 1 ○ 2 ○ 3 ○ 4 ○ 5
(poor) (perfect)

2. Are you previously familiar with the entity (Universal_Serial_Bus)?

○ Yes ○ No

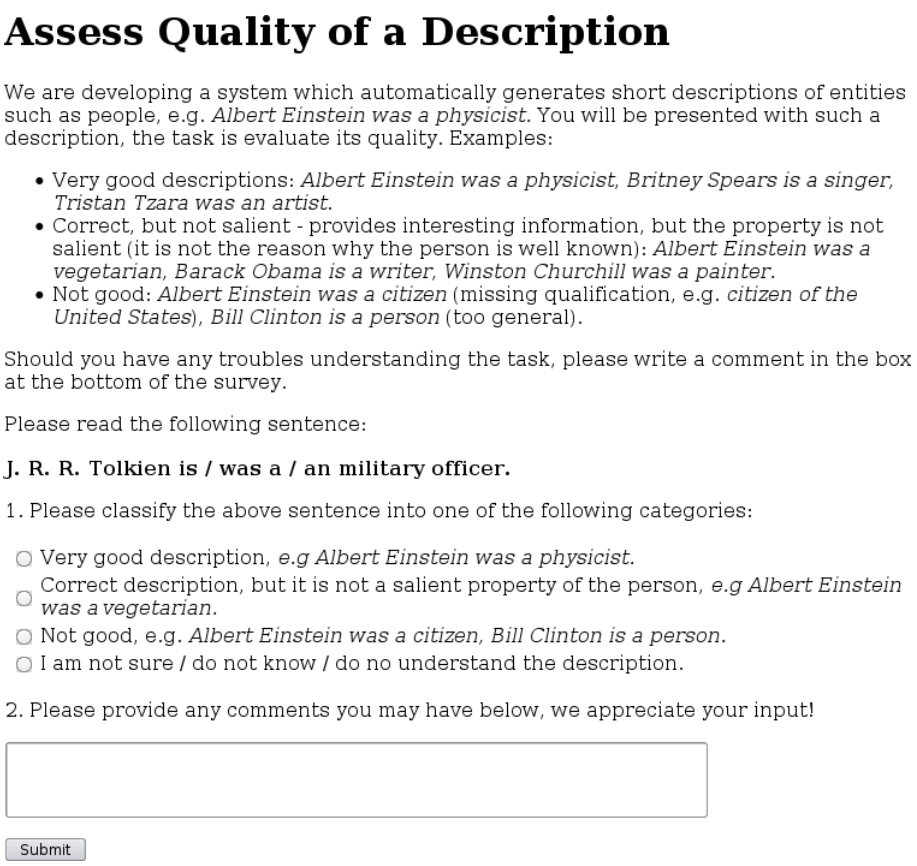
3. Please tell us more about the description:

☐ I know, that the description is **not correct**.
☐ Missing qualification, e.g. *"X is a citizen"* instead of *"X is a citizen of Canada"*.
☐ Too general / abstract, e.g. *"X is a person"*
☐ The description is technically true, but there is a more relavant description, e.g. *"A. Einstein is a violinist"* or *"a left-handed person"* instead of *"a physicist."*
☐ One of the synonyms is better.
☐ I do not understand the description.
☐ Not good for other reason (please elaborate in the comment filed below).

4. Please provide any comments you may have below, we appreciate your input!

Submit

Figure 4.1: Our initial attempt at developing a questionnaire for Amazon Mechanica Turk. The "Universal Serial Bus" is the full name of the USB – a common type of connector installed in PC computers (often used for memory sticks). The turker chose "1" in the first question, "no" in the second and "I know, that the description is not correct" in the last one.



Assess Quality of a Description

We are developing a system which automatically generates short descriptions of entities such as people, e.g. *Albert Einstein was a physicist.* You will be presented with such a description, the task is evaluate its quality. Examples:

- Very good descriptions: *Albert Einstein was a physicist, Britney Spears is a singer, Tristan Tzara was an artist.*
- Correct, but not salient - provides interesting information, but the property is not salient (it is not the reason why the person is well known): *Albert Einstein was a vegetarian, Barack Obama is a writer, Winston Churchill was a painter.*
- Not good: *Albert Einstein was a citizen* (missing qualification, e.g. *citizen of the United States*), *Bill Clinton is a person* (too general).

Should you have any troubles understanding the task, please write a comment in the box at the bottom of the survey.

Please read the following sentence:

J. R. R. Tolkien is / was a / an military officer.

1. Please classify the above sentence into one of the following categories:

- ○ Very good description, *e.g Albert Einstein was a physicist.*
- ○ Correct description, but it is not a salient property of the person, *e.g Albert Einstein was a vegetarian.*
- ○ Not good, e.g. *Albert Einstein was a citizen, Bill Clinton is a person.*
- ○ I am not sure / do not know / do no understand the description.

2. Please provide any comments you may have below, we appreciate your input!

Submit

Figure 4.2: Mechanical Turk questionnaire used for assessing the quality of types. The page shown to workers includes a description of the task, sample types with their evaluation, and the task itself.

assigned each entity-type pair to 5 different turkers. We also used 3 different sample entities for each type that we evaluate. Together this gives the total of 15 judgments per type. The results of our experiment are presented in Table 4.2. They are ordered by decreasing ratio of (good + correct-but-not-salient)/total judgments. The ratio is highest for salient types such as *scholar* or *spiritual leader* and low for *acquirer*, *head*, and so on. This shows that human annotators can successfully distinguish between types that are useful in summaries and those which are not.

4.3 Method Based on Wikipedia Edit History

The other kind of types whose usefulness for summarization needs to be assessed are the leaf types in YAGO (e.g. shaded rectangular nodes in Fig. 3.3). The vast majority of them was derived from Wikipedia categories. In contrast to a small number of types coming from WordNet, there are hundreds of thousand possible Wikicategories. Because they were added by editors as additional

Type	Good	NotSal.	NotGood	NotSure	Ratio
exile	8	7			1.00
reformer	7	7	1		0.93
composer	5	9		1	0.93
performer	10	3	1	1	0.87
scholar	10	3	2		0.87
spiritual leader	9	4	2		0.87
entertainer	8	5	1	1	0.87
leader	8	5	2		0.87
head of state	6	7	2		0.87
negotiator	5	8	1	1	0.87
serviceman	2	10	1	2	0.80
president	10	1	2	2	0.73
representative	3	8	3	1	0.73
educator	2	9	3	1	0.73
clergyman	8	2	1	4	0.67
associate	4	5	4	2	0.60
intellectual	2	7	3	3	0.60
lawgiver	2	7	3	3	0.60
honoree	1	8	5	1	0.60
traveler	1	8	4	2	0.60
head	1	7	6	1	0.53
militant	3	4	7	1	0.47
recipient	1	6	7	1	0.47
skilled worker	1	6	4	4	0.47
disputant	3	3	4	5	0.40
acquirer	1	4	8	2	0.33
unfortunate		5	9	1	0.33
absentee		3	9	3	0.20

Table 4.2: Type informativeness obtained from Mechanical Turk. Each type was assigned three sample entities, each such pair was evaluated by 5 turkers, which gives 15 evaluations per types. The questionnaire we used is presented in Fig. 4.2. Columns *Good, (GoodBut)NotSal(ient), NotGood, NotSure* count how many times each of the answers in question 1 was given. The *ratio* is (Good + NotSal.) / 15. The total cost of the experiment was 6.75 USD.

information about the article, they are never too abstract and we do not face the problem of missing arguments. While in general Wikicategories may not be true ontological types, for example, Albert Einstein could be in the category *physics* or a technical category *articles requiring clean-up*, those were already pruned by YAGO extraction algorithms. The problem that we face instead is to determine how important the leaf types are with respect to the entity. The following hypothesis is the motivation for using the history of changes in Wikipedia as a resource for assessing informativeness of types:

Hypothesis: Wikipedia articles initially contain the most important information, which is later extended with less important details.

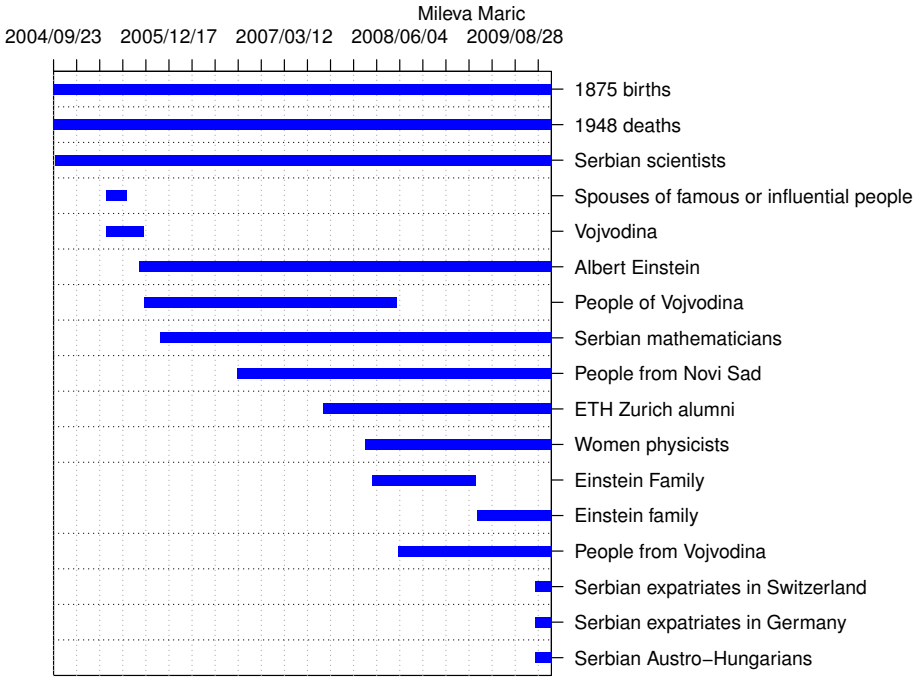


Figure 4.3: Evolution of the Wikipedia page about Mileva Marić. Some categories are equivalent and can me merged, for example, *Einstein family – Einstein Family* and *Vojvodina – People of Vojvodina – People from Vojvodina*.

4.3.1 Per Page Taxonomy Development

From the revision history of a page we build a Gantt chart, which shows the evolution of categories of an entity. The X axis in the chart shows time and the horizontal bars show when the page was in the category. An example is presented in Fig. 4.3. If categories are never removed from a page, we can use our hypothesis directly and rank the importance of categories with respect to the entity by the lengths of the time intervals in the Gantt chart. This assumption, however, rarely holds – in the example (Fig. 4.3) we can see that some categories could be merged. Sometimes it happens because a category was renamed (e.g. People of Vojvodina = People from Vojvodina), and sometimes users make the information more precise (e.g. American lawyers $\neq$ New York lawyers), renaming can also be result of mistakes (accidental changes) or vandalism. The information about renaming is not stored in Wikipedia, we only observe that some categories of a page disappear and others are added to a page.

In order to connect Wikipedia categories which express the same concept, we use the clustering approach outlined below. Given an entity with the set of categories $\{c_1, \ldots, c_n\}$, we build a graph. The nodes are categories which are connected by a link if one of the categories can be merged with the other. Finally, we find all connected components of the graph. We treat all categories within a component as equivalent. Below we present different methods to obtain links between pairs of categories.

Split-and-stem. Changes in categories are often very simple: editors change the capitalization, dashes to spaces, plural to singular, etc. Categories adjusted in such a way can be merged as follows:

1. Transform the name to lower case.
2. Split the name on non-alphanumeric characters.
3. Stem the words using Porter stemmer [110].
4. Merge two categories, if the results are the same. We compare the results as sets of term (i.e. order and repetitions do not matter).

Renaming-consistent edits. Although renaming categories can be a part of larger change, where multiple categories are renamed and the article text is improved as well, oftentimes the editors will only rename a single category in an article. This can be easily spotted when in two subsequent revisions exactly one category is removed from an article and exactly one is added. We look for such changes in our dataset and extract pairs of categories that were renamed. Such changes can be aggregated over all pages in Wikipedia. A sample of renamed categories mined from edit history is shown in Fig. 4.4.

Clustering allows us to use our hypothesis to rank importance of categories by the length of time intervals in the Gantt chart, even if they were renamed, made more specific, etc. We do this by taking for each cluster the union of time intervals of all its categories and ranking clusters by the length of such combined intervals. All categories within a cluster are therefore treated as equally important with respect to the entity.

4.4 Software Architecture

The English Wikipedia with article history can be easily downloaded as a single large XML file (in recent versions the archive was broken into a few files). It contains all versions of all pages, including special pages, such as categories. Revisions of pages are stored without any back references to previous versions – that is, if a typo is corrected, then a full copy of the page will be stored as a next revision. As the result, the file is much larger than necessary, but it is easy to process. For our purposes, we extracted the names of the categories of each regular (encyclopedic) page in the dump. Additionally, for each revision, we also have metadata, such as the author, time stamp, length, comment, and a few other fields. The extraction uses regular expressions on the wiki markup.

We store the revisions in a database table (we used PostgreSQL DBMS), which contains: id of the revision, id of the page, timestamp, a comment and a flag indicating whether the revision is *minor* (this field can be marked by the editor if the change is not substantial, e.g. fixes a typo), length of the page in bytes, and the identifier of the author of the revision. For each revision we store the list of its Wikipedia categories as well.

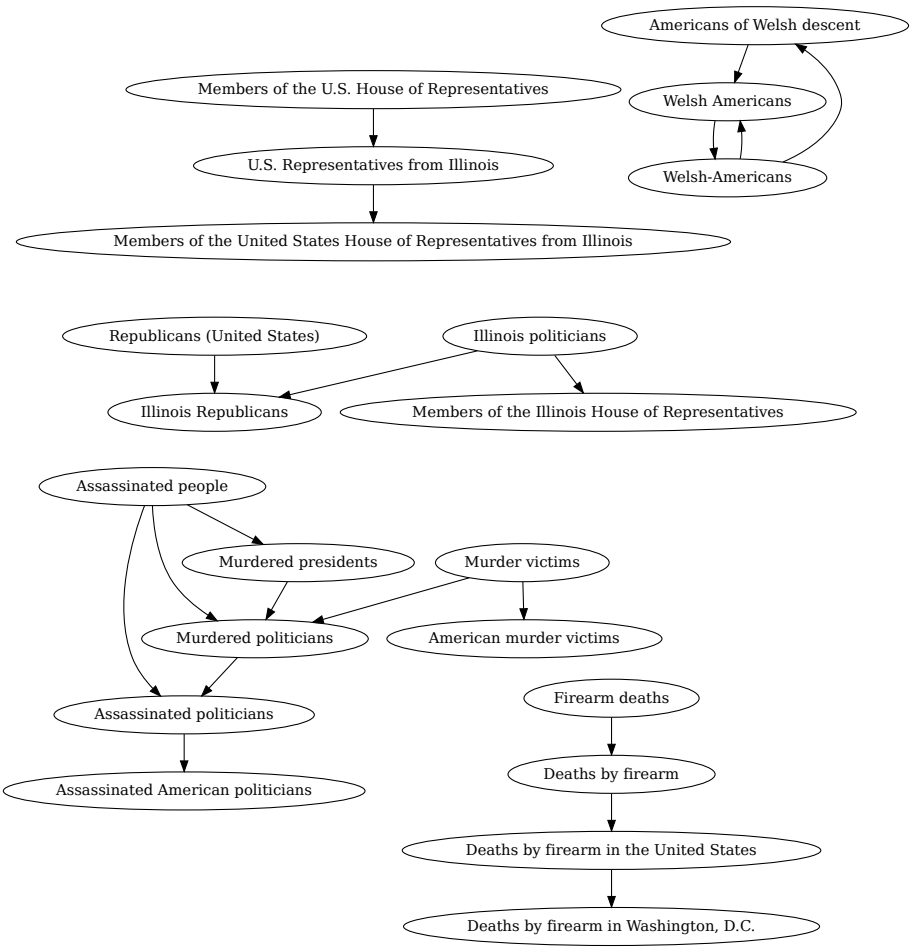


Figure 4.4: Category graph for *Abraham Lincoln*. Edges indicate renaming mined from the Wikipedia edit history. We show the direction of changes, but it is irrelevant for our clustering.

Data Cleaning. The data in Wikipedia is noisy for many reasons. The first step in cleaning process is to drop revisions which were considered *vandalism*. We assume that the latest revision was not a vandalism and proceed backwards:

- if a revision is unique, we output it and move the cursor one revision backwards;
- if it is a copy of another revision (which means that it reverts to it) output the first occurrence and move the cursor one revision prior to the first occurrence.

Additionally, we also filter out all revisions shorter than 32 bytes.

Distributed Processing. Since loading the data involves decompressing a single XML file, it is hard to parallize. On the other hand analyzing page renaming

in the large dataset can benefit from distributing the work on a cluster of machines. We therefore implemented our renaming event extraction on Apache Pig platform (`https://pig.apache.org/`). Pig offers a SQL-like environment for processing the data on Apache Hadoop (`https://hadoop.apache.org/`) platform, thereby providing a convenient alternative for designing multiple MapReduce jobs, which implement standard operations on dataset, such as filtering and joins. For an overview of MapReduce paradigm we recommend the book by Lin and Dyer [80].

4.5 Evaluation

Our objective is to obtain a ranking of categories by their importance with respect to entities, based on our hypothesis that important categories are added to a Wikipedia page first. To rectify the effect of category renaming we proposed methods of merging Wikipedia categories which express the same concept in different ways. Therefore we experimentally evaluate both the quality of clustering and the quality of the resulting ranking of the categories. To this end we manually created the ground truth for both clustering and ranking. For each entity in our test collection we marked which categories are equivalent and should belong to the same cluster and also ranked their importance. Our ground truth annotations fulfill a natural constraint that all categories within a cluster are assigned the same rank. The rankings allow draws between clusters, since often it is difficult to judge their relative importance.

Our test collection consists of two sets of entities: US presidents and Lithuanian gods. The first set contains prominent entities with long articles and many categories including both important ones as well less important details. The second set contains entities with much shorter articles and few categories, usually of equal or similar importance. In total our collection contains 18 entities and 356 entity category pairs.

4.5.1 Evaluation Measures

Clustering. Clustering can be evaluated using *internal* criteria, which consider only the clustering itself and judge the similarity of items within clusters and between them. Evaluation using *external* criteria compares the results against a known ground truth. In our settings we follow the latter approach. We use two measure of the quality of clustering: *purity* and the *Rand index*. Let N be the total number of items, $C = \{c_i\}_{i=1}^{I}$ be a set of clusters, and $G = \{g_j\}_{j=1}^{J}$ be the ground truth. We define purity as:

$$purity(C, G) = \frac{1}{N} \sum_i \max_j |c_i \cap g_j| .$$

Intuitively, each cluster is assigned to the most common category from the ground truth, then we calculate the fraction of elements which belong to this category and average this value over all clusters. Purity is simple to understand, but it does not penalize too fine grained clustering, in particular, assigning each item to a singleton cluster will always yield maximal purity equal one.

The Rand index (RI) is another measure of clustering quality, which avoids the aforementioned problem. We can see clustering as the problem of deciding

whether two given items belong together or not. The Rand index is then the fraction of $N(N-1)/2$ pairs which are classified correctly:

$$RI(C,G) = \frac{TP+TN}{N(N-1)/2},$$

where TP (true positives) is the number of pairs which are in the same cluster in C and the ground truth and, similarly, TN (true negatives) is the number of pairs which are in different clusters. Purity and the Rand index take values from 0 in the worst case to 1 in the best case. For more details about evaluation of clustering the reader should consult Chapter 16 in [87].

Ranking. To compare a ranking against the ground truth we use Kendall rank correlation coefficient

$$\tau = \frac{n_c - n_d}{N(N-1)/2}.$$

Where n_c is the number of pairs of elements which are in the same order in both rankings (concordant pairs) and n_d is the number of pairs which are in opposite order (discordant pairs). Kendall τ take values from -1 to 1, where 1 corresponds to two identical rankings and -1 to ranking items in opposite orders. Since our ground truth rankings allow draws, the bounds are tighter in our evaluation.

4.5.2 Discussion

The results of our experiments are presented in Table 4.3. On the set of important entities (US presidents) we can see that clustering categories based on renaming edits yields the best results in term of quality of the resulting ranking of categories. It outperforms both no clustering and split-and-stem approach, which actually merges very few categories. The effect is also visible in terms in the quality of clustering (Rand index). In case of the less prominent entities (Lithuanian gods) the results are worse in terms of both clustering (Rand index) and ranking. This is expected, as the categories all have similar importance.

4.6 Application in Entity Timelines

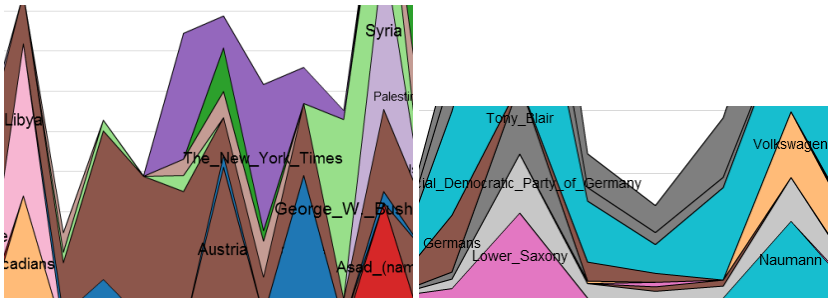
Entity Timelines [91] is the visualization system for entities and concepts in a large text corpus developed by Arturas Mazeika with me as a co-author.

Ranking the importance of types with respect to an entity has applications beyond summarization in knowledge bases. In this section we present the Entity Timeslines system [91] for efficient visualization of entities and concepts in a text corpus, which uses the technique based on Wikipedia edit history to rank types used for grouping entities.

The text visualization community proposed tools to visualize the evolution of concepts represented by keywords. Since words rather than named entities are considered, such systems are not naturally able to recognize when the same entity is referred to by different names, e.g. Michail Sergejevic Gorbachov and Mikhail Gorbachev. Similarly, related concepts cannot be easily merged, e.g. it is

		No Clustering				Split-and-Stem				Renaming Edits			
Entity	**# Cl.**	**# Cl.**	**Purity**	**RI**	**Kendall**	**# Cl.**	**Purity**	**Rand**	**Kendall**	**# Cl.**	**Purity**	**Rand**	**Kendall**
Ašvieniai	8	12	1.00	0.89	-0.08	11	0.92	0.88	0.06	10	1.00	0.92	-0.23
Bangpūtys	3	8	1.00	0.61	-0.18	8	1.00	0.61	-0.18	6	1.00	0.68	-0.04
Dievas	5	14	1.00	0.76	0.07	14	1.00	0.76	0.07	8	1.00	0.86	-0.22
Peckols	8	10	1.00	0.93	-0.04	10	1.00	0.93	-0.04	10	1.00	0.93	-0.04
Perkūnas	7	12	1.00	0.83	-0.11	12	1.00	0.83	-0.11	9	0.83	0.82	-0.14
Žaltys	6	10	1.00	0.78	0.07	10	1.00	0.78	0.07	9	1.00	0.80	0.11
Average	6.2	11.0	1.00	0.80	-0.05	10.8	0.99	0.80	-0.02	8.7	0.97	0.84	-0.09
Andrew Jackson	12	19	1.00	0.95	0.03	19	1.00	0.95	0.03	12	1.00	1.00	0.53
Andrew Johnson	21	33	1.00	0.97	0.04	33	1.00	0.97	0.04	21	0.88	0.97	0.40
Barack Obama	22	31	1.00	0.97	0.13	30	1.00	0.97	0.10	23	0.90	0.97	0.28
Benjamin Harrison	22	32	1.00	0.98	-0.05	31	1.00	0.98	-0.11	20	0.94	0.99	0.24
Bill Clinton	8	9	1.00	0.97	0.03	9	1.00	0.97	0.03	9	1.00	0.97	0.03
Calvin Coolidge	24	30	1.00	0.99	0.12	29	1.00	0.99	0.12	23	0.90	0.99	0.31
Chester A. Arthur	21	32	1.00	0.97	-0.10	31	1.00	0.98	-0.09	21	0.94	0.99	0.07
Dwight D. Eisenhower	25	31	1.00	0.98	0.26	31	1.00	0.98	0.26	27	1.00	1.00	0.42
George H. W. Bush	19	26	1.00	0.98	0.00	26	1.00	0.98	0.00	21	0.92	0.98	0.17
George Washington	13	16	1.00	0.97	0.23	16	1.00	0.97	0.23	14	1.00	0.98	0.32
Richard Nixon	21	22	1.00	1.00	0.33	22	1.00	1.00	0.33	21	1.00	1.00	0.36
Ronald Reagan	6	9	1.00	0.89	0.00	9	1.00	0.89	0.00	6	1.00	1.00	0.00
Average	17.8	24.2	1.00	0.97	0.09	23.8	1.00	0.97	0.08	18.2	0.96	0.99	0.26

Table 4.3: Evaluation of clustering and ranking of Wikipedia categories. We report the number of clusters in the ground truth (Cl.) and for each category clustering method: number of produced clusters, purity, Rand index (RI), and Kendall rank correlation coefficient.



(a) Chirac\Schröder (b) Schröder\Chirac

Figure 4.5: Contrasting J. Chirac vs. G. Schröder in Entity Timelines.

not easy to connect Walkman and iPod, which are both portable music players. In order to overcome these limitations, the proposed system operates at the entity level and uses the YAGO knowledge base to group similar entities.

Demonstration of the system operates on the collection of The New York Times articles published between 1987 and 2007. Pre-processing the corpus included identifying and disambiguating named entities in all articles. Evolution of entities is visualized as a timeline using the paradigm of stacked areas and river metaphor. Given a named entity as a query, other entities co-occurring in the same document are visualized as stacks (polygonal areas). The area of the stack depicts the frequency of the co-occurring entities over time. Canonicalization of named entities allows effective comparison and contrasting of two timelines. The system supports interactions with the visualization and multiple displays over the data, which include taking union, intersection, difference, etc (Fig. 4.5).

Entity Timelines can group similar items based on their YAGO types. YAGO offers multiple ways to roll up/drill down hierarchies for almost all entities, for example, the 19 hierarchies of `Mikhail_Gorbachev` include the following: Soviet politician → politician → leader → person → entity, Russian atheist → atheist → disbeliever → nonreligious person → person → entity, Moscow State University alumni → alumnus → scholar → intellectual → person → entity, etc. Since the timelines consists of multiple named entities, using all hierarchies of a named entity for grouping would result in a large lattice. Instead, the best hierarchy is found for each named entity and they are all incorporated into one (Fig. 4.6).

The best hierarchy is identified in a two-step process. First, we pick types that correspond to the earliest and most long-lasting categories in the history of the Wikipedia article about the entity. This step utilizes the information about importance of types that we extract from article edit history. The system, however, did not use clustering of the categories. In the second step we carefully applied an additional filtering using the term frequency scores of nouns and words of the abstract from the Wikipedia article (content up to the table of contents). The computed hierarchy is used in semantic drill down/roll up as well as for filtering.

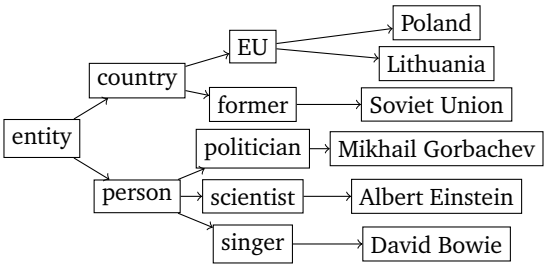


Figure 4.6: Union hierarchies for *Lithuania, Poland, Soviet Union, Mikhail Gorbachev, Albert Einstein,* and *David Bowie* (simplified for clarity).

CHAPTER 5

Fact Spotting

Information extraction techniques are used to automatically create large scale knowledge bases (KBs). To achieve high precision, they are typically applied to quality sources, such as Wikipedia, and aim to obtain only high-confidence occurrences of facts. However, retrieving many additional occurrences of facts in supplementary sources, such as news, product reviews, or social media, can benefit applications for assessing the truth of statements, gathering statistics, contextual enrichment, quality assurance of the KB, and advanced analytics that combine text sources with structured knowledge.

In this chapter we address the task of retrieving occurrences of known facts from previously unseen supplementary documents, the problem which we call *fact spotting*. Two solutions are presented. The first is based on finding high confidence facts using relation paraphrase dictionary and then speculatively guessing presence of facts with partial evidence. The second approach employs aggressive matching to find mentions of entities and relations in text and builds a large pool of fact candidates. We then perform joint matching of facts with consistency constraints across multiple, mutually dependent facts. Finally, we construct a graph from the fact occurrences and compute a random-walk-based ranking of spotted facts. We evaluate our approach by spotting Freebase facts in biographies and news articles from the Web.

5.1 Introduction

5.1.1 Motivation

Information extraction (IE) [28, 34, 52, 114, 90, 120] is the task of isolating machine-readable facts, usually in subject-predicate-object form, from different kinds of Internet or enterprise contents originally created for human reading. Such sources range from structured but schemaless (e.g. Wikipedia infoboxes) to fully unstructured (e.g. news articles, product reviews, posts in online communities). Owing to the difficulty of the task, IE systems tend to work well only on certain types of inputs, for example, by tailoring the extraction to Wikipedia infoboxes or Web sites with highly regular structural patterns. However, facts are also expressed in many other sources that are *not amenable to IE*. This is not a problem if we are only interested in the obtained facts themselves,

since – depending on the application domain – sources such as Wikipedia may already have sufficiently good coverage and additional sources are not needed. There are, however, several scenarios where finding further – ideally many – supplementary locations of fact occurrences is important.

Truth Finding and KB Curation. Although IE methods have made enormous advances, errors still arise due to both incorrect extractions and false statements in source documents. Especially social media may even include intentionally misleading statements (e.g., "Obama is a muslim"), and even news sources often have a political bias. Checking the truthfulness of statements is alleviated and strongly supported by providing it with more sources of evidence [83, 84, 103, 107, 139, 144]. When automatically constructing a KB using IE methods, this task becomes even more important. In this context, fact assessment is typically done by human curators or advanced forms of crowdsourcing [125]. The curators' work is greatly simplified if they see a variety of supplementary sources that support or contradict the fact in question. This has a big impact on the cost of KB quality assurance.

Evidence Corroboration for Better IE. Even for the task of IE itself, it is often beneficial to gather multiple sources of evidence or counter-evidence for observing an entity or a fact. Supplementary sources are of great value for corroboration [135, 136].

Statistics Gathering. Many applications need statistics about entities and relational facts; the bare facts alone are insufficient. These include querying a KB with search-result ranking (e.g., [69]), and KB exploration such as analyzing and explaining the relationships between entities of interest (e.g., [47]). The necessary statistics for these tasks are way more informative and thus lead to better outputs if they are not limited to the sources from which facts were originally extracted, but consider also other documents with fact-supporting evidence or co-occurrences of entities.

Text Analytics with Background Knowledge. Understanding text documents in terms of entities and relationships can improve applications over enterprise documents, news articles, and social media. Examples include multi-document summarization, monitoring collaboratively edited contents such as Wikipedia, news aggregation, opinion mining on review sites and online communities, classifying and processing customer requests (e.g., questions or complaints about services or products), and other kinds of business, market, and media analytics.

5.1.2 Contribution

We address the *fact spotting* problem defined as follows.

Definition. Given a document d about an entity s, and a set F of facts about s (from a KB), find the subset $F' \subseteq F$ of facts stated in the document.

Fact Spotting. The contributions of this work are fact spotting methods, which aim at achieving both high precision and high recall. We do not place any assumptions on the origin and style of the documents where the facts are spotted. Our methods are designed to handle difficult inputs where the facts

of interest are stated using complex natural language, and potentially spread across multiple sentences connected by pronouns or other co-references.

Key Insights. In order to deal with these difficulties, we exploit *dependencies between facts* that follow from their semantics. First, facts about dates and locations of events are complementary and are typically stated along with the event they describe. For example, the fact David Beckham –MARRIEDIN→ 1999 will usually accompany David Beckham –MARRIEDTO→ Victoria Beckham. Such mutually dependent co-facts are readily available in KB's like YAGO 2 [62] or Freebase [19]. YAGO 2 explicitly annotates facts with time and location, whereas Freebase groups related facts together; in such groupings we treat all facts about dates and numbers as dependent on other facts. Second, there is a benefit in considering named entity linking and fact spotting together. For example, the mention "Dirty Harry" is ambiguous and can mean both Dirty Harry (movie) and Harry Callahan (role). The particular disambiguation that we choose will help us decide whether a sentence in Clint Eastwood's biography fits better the fact Clint Eastwood –PLAYEDIN→ Dirty Harry or Clint Eastwood –PLAYEDROLE→ Harry Callahan.

Our Approach. We present two solutions to the fact spotting problem. The first is based on speculative spotting of facts in presence of partial evidence. We start by finding high confidence facts and then add facts, whose presence we are not sure of, but their related facts were found. The second method has three stages. First, we match components of facts against the given text. This involves matching entities to their mentions using a dictionary of aliases, and matching relation to their surface representations, typically multi-word phrases occurring in the document. Our goal at this stage is to compile a large set of candidate matches that will later be pruned. Performing co-reference resolution (e.g., for personal pronouns) makes it possible to find facts whose components are spread across multiple sentences. As our goal is high recall first, we are willing to accept a relatively high risk of false positives at this stage. The second stage is jointly matching the components of a fact and also several mutually dependent facts in the text. Here we prune the candidate set from the previous step, eliminating many of the false positives. We judiciously model this stage as an integer linear program (ILP). The solution is a consistent set of facts found in the document and a disambiguation of mentions to entities. Finally, the third stage uses random walks with restarts on a specially constructed graph to rank facts. Our technique takes into account the dependencies between facts. The output is a list of facts sorted in descending order of confidence that a fact was spotted correctly.

In the remainder of the chapter we assume that both the subject entity and the document to be inspected are given. If we are only given the document, we can heuristically determine the entity by running a named entity disambiguation system [49, 63, 93, 111, 32] on the document text. Then we choose the most frequently mentioned entity, possibly in a weighted voting manner, e.g., by giving entities in titles, heading, or special font higher relevance.

Evaluation. We experimentally evaluated our methods on a set of biographical documents of varying styles, gathered from different Web sites. Our experiments confirm that considering dependencies between facts and performing fact spotting and entity linking jointly are beneficial for obtaining good precision

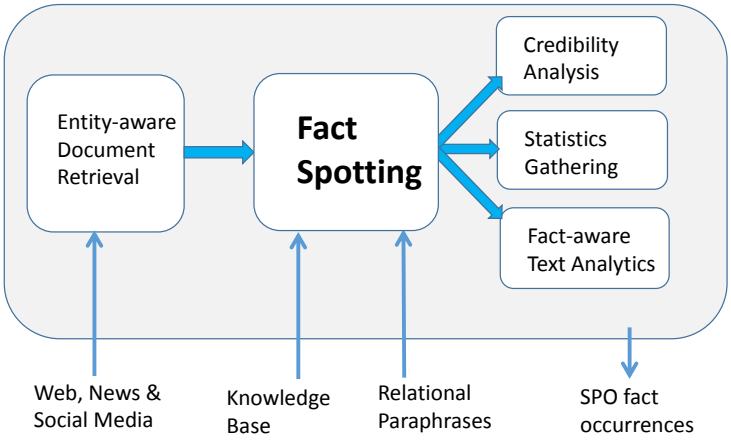


Figure 5.1: Fact spotting system environment.

and good recall. We also present use-cases that leverage fact spotting for named entity disambiguation alone and for the task of KB curation.

5.2 Preliminaries

The system environment we are working in is depicted in Fig. 5.1. The input to the fact spotting component is an entity (or a set of facts about it) and an arbitrary document. Our methods make use of a knowledge base and a dictionary of paraphrases to determine which facts occur in the document. The output, in the form of fact positions or aggregated statistics, can be processed in downstream applications, including truthfulness analysis, entity/fact-aware summarization, KB curation, etc.

5.2.1 Freebase Data

We use Freebase (`www.freebase.com`) as a knowledge base of facts. Freebase stores data as a graph. Nodes which correspond to entities and classes are known in Freebase parlance as topics. They represent people (e.g., Bob Dylan), artistic creations (e.g., Hurricane (song)), classes (e.g., music awards), abstract concepts (e.g., love), artistic movements (e.g., Impressionism), etc. Topics can have types assigned to them, e.g., song writer. A type can be thought of as a container for properties. Types are grouped into domains, e.g., music, business, medicine, and given identifiers which consist of domain name and type name, e.g., `/business/company`. Property names contain the name of their types, e.g., `/business/company/industry`. Not all identifiers follow the `/domain/type/property` template, for example, `/en` is the namespace of the most common topics in English, and `/m` contains machine readable identifiers. It is possible that a topic has more than one identifier.

Freebase data can be represented in the following JSON format. A topic consist of two fields: an `id`, for example `/m/02mjmr` for Barack Obama, and `property` field. The latter is a mapping from a property name (e.g. `/celebrities/celebrity/net_worth`, `/people/person/spouse_s`) to a tuple containing `val-`

uetype, value, and count (number of different values – Freebase properties are by default multi-valued). The following property types exist: float, int, bool, datetime, uri, string, key, object and compound. The first six are literals. Key is an identifier, e.g. Barack Obama can be referred by the machine id /m/02mjmr or a human readable /en/barack_obama. Object means that the argument of the property is another topic in Freebase, e.g. books published by an author. Finally, compound values consist of several fields, e.g. marriage relation contains the person one is married to, it can also contain a start date, optional end date, location of the ceremony, type of union etc.

Note that this data model, in JSON, is quite different from the typically used RDF model of SPO triples. For example, Freebase makes extensive use of nested values of values (e.g., the object of a triple representing a person winning an Acedemy Award contains the year and the category of the award). We deal with this by breaking down composite facts into triples (but we retain the information which pieces belong together). The details are presented below.

Types in Freebase are described by a schema, which defines the types of properties. For example properties of a person are listed at www.freebase.com/people/person, date of birth has the expected type /type/datetime, gender is an enumeration of type /people/gender, and spouse(s) has type /people/marriage which is a *mediator* type. Mediators are types which connect objects which are in a complex relation, for example, marriage as discussed above. Freebase does not have a true type hierarchy, but /freebase/type_hints/included_types simulate it, e.g. /book/author includes /people/person but not all authors are indeed people, some are governments, museums, and so on [70].

The basic building blocks of facts are simple values and predicate name. A simple value can have one of the following types: datetime, string, uri, key, int, float, bool, object. Predicate names occur in two places in facts: as relations names, e.g., /people/person/date_of_birth, and as attribute names in compound values, e.g., /people/marriage/from. We will refer to simple values and predicate names as atoms.

A compound value is a non-empty sequence $\{\langle name, value\rangle\}_{i=1}^{k}$ of identifiers and predicate names. A fact is a triple $\langle S, P, O\rangle$ consisting of a subject, predicate name and object. The subject is is always a simple value of type object, whereas the object is any simple or compound value.

A compound fact $\langle S, P, \{\langle name, value\rangle\}_{i=1}^{k}\rangle$ can be decomposed into a set of simple facts: $\{S, P, < name_i, value_i >\}_{i=1}^{k}$. For example a fact David_Beckham spouse { <from, 1999>, <spouse, Victoria_Beckham> } will be decomposed into two facts: David_Beckham spouse {<from, 1999>} and David_Beckham spouse {<spouse, Victoria_Beckham>} . While in the example it is possible to deduce the the original fact, in general the decomposition is not reversible. If a person is married more than once, we will not be able to tell which date refers to which spouse.

Decomposing facts simplifies both spotting and evaluation of the results, but it would throw away precious information about dependencies among values, which can be used to achieve good results. Therefore, we keep track of atoms

which were parts of a compound value. In the speculative spotting we will refer to two atoms as *siblings* if they occur in the same compound value (this also applies to attribute names and values).

In the other spotting method we make a distinction between different facts derived from the same compound. Some facts temporally or geographically qualify other event-like facts, for example, David_Beckham –MARRIEDIN→ 1999 and David_Beckham –MARRIEDAT→ Lutrellstown_Castle describe the main fact David_Beckham –MARRIED→ Victoria_Beckham. Our working hypothesis is that, within any coherent text source, whenever a *dependent* fact occurs, the main or *central* fact is stated in the text as well. The converse is, however, not true: central facts can occur without dependent facts. Such dependencies between facts are not specific to Freebase, for example, YAGO 2 groups related facts in SPOTLX representation.

5.2.2 Entity Aliases

Entities in a KB have unique identifiers, but their mentions in text use various different names. KBs such as Freebase, YAGO and DBpedia provide such alias names of their entities. Freebase stores them in `/type/object/name` and `/common/topic/alias` relations. The former is the unique, canonical name, similar to article titles in Wikipedia (e.g. Victoria Beckham), while the latter contains alternative names of the entity (e.g. Victoria Caroline Adams, Posh Spice).

While Freebase may know many possible aliases, it often misses their minor variations. For example, in one of the biographies of Arnold Schwarzenegger, *University of Wisconsin-Superior* is referred to as *University of Wisconsin* – a name which is not known to Freebase for this entity. This can be solved by approximate string matching, which is what we do in the Reverse Extraction method, or by carefully expanding the set of aliases, which is our approach in the Speculation method.

To expand the sets of aliases, we drop single words from labels which are longer than three words. To avoid incorrect matches, we add the constraint that the new, shorter labels must not be the same as any label of a known entity. For instance, we expand the set {*Patrick Arnold Schwarzenegger*} (the son of the famous bodybuilder) with the derived labels {*Patrick Schwarzenegger, Patrick Arnold*}. Our derived set does not include *Arnold Schwarzenegger*, because this would clash with the name of another entity (his father). Name clashes still occur among labels which were not derived; for example, *Clint Eastwood* is the name of both the actor and his father.

5.2.3 Relation Patterns

Predicates in a KB denote, in a canonical form, the relations that entities participate in. For instance, textual phrases such as “married to”, “tied the knot with” or “had their wedding” are all captured by the Freebase predicate /PEOPLE/PERSON/MARRIAGE. However, unlike being aware of entity aliases, none of the major KBs captures the different paraphrases of their relations; they do not store information that “had their wedding” is synonymous to the

MARRIAGE relation. In order to find relations in text we need to resort to an external source of paraphrases. If the set of relations were small, a list of such phrases could be constructed manually. While this approach could lead to high precision, it is not scalable and it is also likely to have poor recall. Instead, we pursued two alternative approaches. In the Speculation method we make use of PATTY [104], an existing repository of relational paraphrases, and in the Reverse Extraction method, we generate the paraphrases ourselves using Freebase relation names and WordNet.

5.3 Speculation

In this section we present our first approach to fact spotting. The method is based on matching parts of known facts to text. While matching the entities can be easily achieved using their aliases known to KB, we resort to external resource to find mentions of relations. Matching all three parts of a subject-predicate-triple is insufficient for good recall. To alleviate this problem, we show how to spot facts accurately if only partial evidence was found in text.

Relation Patterns

Spotting the predicate of an $\langle s, p, o \rangle$ triple requires that we know which phrases can be used to express it. For example, occurrence of words like *died, death*, in proximity of a city name means that a sentence expresses `diedIn` and not `bornIn` relation. To obtain such phrases we use PATTY [104] (we discuss it in more details in Chapter 2). PATTY patterns consist of words, part of speech tags, wildcards and types. For example, a pattern *<person>'s [adj] voice * <song>* matches "Amy Winehouse's soft voice in Rehab". In our system we use a simplified form of PATTY patterns. First, we drop the leading and trailing wildcards such as *[[det]]*. Then, we filter out the patterns which, after this step, still contain wildcards. The result is a set of word sequences for relations. The next step is to eliminate redundancy among patterns to speed up matching them in documents. For instance, the pattern *studied with* makes the pattern *who studied with* redundant, therefore we drop patterns which are superstrings of other patterns. Our notion of superstring respects words boundaries, so *child* is not a substring of *children*.

PATTY knows only paraphrases for relations from DBpedia and YAGO; to use it with our data set we had to map patterns to Freebase relations. Since the number of relations is not prohibitively large we decided to perform this step manually. An alternative and feasible approach would be to re-run the PATTY construction algorithms with Freebase relations. In addition to PATTY, we also use Freebase relation names to generate patterns. Since all relation names follow a similar structure, we simply take two trailing parts; for example `/theater/theater_role/play` yields `theater role` and `play`.

Spotting Algorithm

Our fact spotting method assumes that it is presented with a text document about a single entity and that all the facts stated in the document are about this entity. While biographies do contain facts about other entities, they do not pose

Algorithm 3: Speculative Fact Spotting.

```
/* Collect atomic parts of facts                                    */
atoms = ∅
for f ∈ facts do
  | atoms = atoms ∪ atoms(f)
end
/* matches is a mapping from atoms to positions in text             */
for a ∈ atoms do
  | matches[a] = ∅
  | if a is an object then
  |   | for l ∈ getObjectLabels(a) do
  |   |   | matches[a] = matches[a] ∪ occurrences(l, text)
  |   | end
  | else if a is predicate then
  |   | for p ∈ pattyPatterns(a) do
  |   |   | matches[a] = matches[a] ∪ occurrences(p, text)
  |   | end
  | else if a is value then
  |   | matches[a] = occurrences(a, text)
  | end
end
/* found is the set of fact in text                                 */
found = ∅
for f ∈ facts do
  | if predicate and object found in proximity then
  |   | found = found ∪ {f}
  | end
  | if object and a sibling found in proximity then
  |   | found = found ∪ {f}
  | end
end
return found
```

a problem, because they are uncommon and different than the facts about the entity we are interested in. The assumption we make simplifies spotting $\langle s, p, o\rangle$ triples to spotting $\langle p, o\rangle$ pairs.

For the reasons of efficiency, fact matching is divided into two phases. We noticed that facts often share their atomic parts, such as relation names and objects, therefore we first find the matches to atomic objects, and then use them to find matches of whole facts.

The compound facts are decomposed before spotting, so that our algorithm only deals with $\langle s, p, o\rangle$ triples. We retain, however, information about field names in facts derived from compound facts, which hence have the $\langle s, p, \{(attr, value)\}\rangle$ form ($\langle s, p, \{attr, value\}\rangle$ contains an extra $attr$ field, but we consider it a triple).

The spotting algorithm is presented in Alg. 3. We start by extracting the atoms from the facts about the current entity. The atoms are the simple values, relation

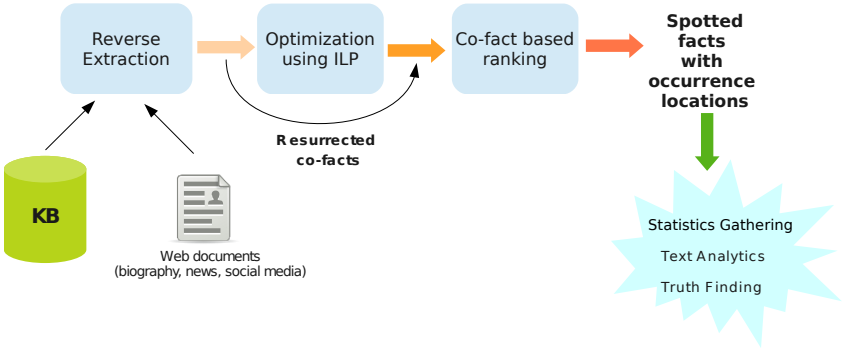


Figure 5.2: Overview of the fact spotting pipeline.

names, and compound attribute names. Subsequently, we find the occurrences of facts in text, and store their locations.

In the simplest case we find the match to the predicate p and the object o, and if such matches occur close enough in the text, we assume that the $\langle s, p, o \rangle$ fact was found. In our implementation "close enough" means that the beginnings of matches of the predicate and object have to be less than 200 characters apart. Oftentimes, it is possible to find an occurrence of the object, but we do not have a match for relation. In such a case we consider presence of a sibling atom as a proof that we indeed spotted the fact.

5.4 Reverse Extraction

Speculation method presented in Section 5.3 uses high quality dictionary of relational paraphrases to find relations in text. This approach yields high quality facts, but may be limited in recall. The problem is then to augment the set of found facts with possible true facts that were not found. We do this by speculatively placing facts in the output, if we find evidence for related facts, even if the relation is not present (by exploiting sibling objects in compound facts).

An alternative is to aggressively find a large set of candidate facts to achieve high recall and then prune false positives for high precision, which is the second approach that we pursue. In this section we present a method for spotting facts with high recall and then in Section 5.5 we show how the false positives can be pruned. In our discussion and experiments we use Freebase as our KB and apply fact spotting to text-centric Web pages. All our techniques can be easily carried over to other KBs and other kinds of input sources. The outline of our approach is shown in Fig. 5.2.

Matching Entities, Dates and Literals

We find mentions of entities and literals in the input text using the name aliases provided by the KB, and assign weights to these occurrences. The mentions include Freebase entities (e.g., Arnold_Schwarzenegger), literal strings (e.g., *Hasta*

la vista, baby), dates, and numeric values. For simplicity, we will refer to all of them as entities. Each mention has a unique textual position, $l = \langle start, end \rangle$, which is a pair of offsets in the text. We generate a bag of matches between entities and mention positions $match(e, l)$ associated with weights $w_{e,l}$.

- First, we perform exact-match lookups of names and aliases provided by the KB (called entity labels). Freebase provides possible labels for entities with the predicates /TYPE/OBJECT/NAME and /COMMON/TOPIC/ALIAS. Exact matches are given weights $w_{e,l} = 1.0$.
- Exact matching misses out on many mentions. For approximate matching, we identify candidates in the text in two ways. First, we run a Named Entity Recognizer (NER), essentially a trained CRF [51], from the Stanford CoreNLP [88] software package. Second, for higher coverage, we also detect noun phrases using TokensRegex [29], which is a framework that allows matching text with regular expressions that can operate over tokens and annotations. For example, "[{ tag : /NN.*/ }] + [/of/] [{ tag : /NN.*/ }] +" matches a sequence of tokens annotated as nouns (NN, NNP, NNPS, etc.), followed by the word "of" and a sequence of nouns. [1] Matches are weighted by the word overlap between a mention text and the best fitting label of the entity:

  $$w_{e,l} = \max_{label \in labels(e)} \frac{|mention \cap label|}{|mention \cup label|}.$$

  When calculating the scores we skip non-capitalized stop words.
- Matching dates from Freebase facts directly in text often misses temporal mentions due to variations in their representations (e.g., *7/4/1999* can occur as "`7th of April, 1999`"). Moreover, we observed that aggressive matching of values produces spurious matches (for instance, "`23`", David Beckham's jersey number in the LA Galaxy team, could match a part of a date). In order to limit false positives, we consider only dates and numbers identified by the Stanford NER tool and assign them the weight $w_{e,l} = 1.0$.

The above techniques that match mentions with entities do not capture references such as pronouns. To resolve co-references, we use the Stanford tool, which uses the multi-pass sieve and rule-based methods of [77]. For computing weights of mentions, we propagate the best score to all mentions within a co-reference chain. In case of our test documents, which are all single entity-centric biographies, we found that a lightweight heuristic can work equally well: we match all occurrences of *he, his* or *she, her* to the subject entity of the facts (we choose the most frequent pronoun). The heuristic improves the run time of our system by avoiding a time-consuming call to the co-reference resolution tool.

For efficiency, we aggregate multiple matches of the same location-entity pairs and assign them the maximum weight over the matches. At this stage it is also possible to do additional steps such as match pruning. For instance,

[1] NN, NNP, NNPS are Penn Treebank part-of-speech annotations for noun (singular or mass), proper noun (singular), and proper noun (plural) respectively.

a mention *Schwarzenegger* at a particular location can have matches to both *Arnold Schwarzenegger* and his son *Patrick Arnold Schwarzenegger*. We can apply heuristic that a match to a selected central entity "bumps out" other matches. Similar effect can be achieved by single-sense constraint in our joint fact spotting later in the pipeline.

Matching Predicates.

Predicates (relations) in Freebase are referenced using human-readable identifiers, following a /DOMAIN/TYPE/PROPERTY scheme, for instance, /SPORTS-/PRO_ATHLETE/TEAMS. We could search for the predicate name in the text directly, but this would miss most relation occurrences. Therefore, for each predicate p, we build a set E_p of phrases that express the relation by replacing parts of the predicate string, with related words. In our system, the related words are defined as neighbors in the WordNet [48] lexical/taxonomic graph: synonyms, hyponyms, morphosemantic links, derived terms, and glosses. For example, phrases such as "sib", "twin", "half blood", "blood relative", and "brother or sister" are generated for the predicate /PEOPLE/PERSON/SIBLING. Similarly, for the predicate /ORGANIZATION/ORGANIZATION_FOUNDER/ORGANIZATION-_FOUNDED, phrases such as "organization establish", "organization cofounder", "company founder", "enterprise founder", and "set up or found" are generated.

Our expansion process is aggressive and can generate the same phrase for multiple relations. To counter the potential problem of diluting the set of predicate phrases, we assign idf-based scores, which down-weight ambiguous expansions:

$$weight(phrase) = \min_{term \in phrase} idf(term).$$

Given a fact $\langle s, p, o \rangle$ we find all matches $match(l_s, s)$ and $match(l_o, o)$ and enumerate paths between all l_s and l_o. Here, a path refers to the grammatical structure of a sentence based on deep parsing (e.g., links between a verb and its subject, its object or objects, an adverbial modifier, etc.). Our notion of a path includes the shortest path in the dependency tree augmented with dependents of its nodes. Since we trace co-references, paths need to be considered only within a sentence. For robustness we skip stopwords and lemmatize words on the relevant paths (i.e., normalize plural form or different tenses to singular and infinitive). A match between the word sequence on a path and a candidate relation is scored as follows:

$$sim(p, path) = \max_{phrase \in E_p} \left[weight(phrase) \cdot \frac{|phrase \cap path|}{|phrase|} \right].$$

We assign each fact $f = \langle s, p, o \rangle$ the score,

$$score(f) = \max_{l_s, l_o, path} w(s, l_s) \cdot w(p, path) \cdot w(o, l_o). \tag{5.1}$$

When the score is non-zero, we consider the fact spotted. For example, the sentence "`Not long after his first son, Brooklyn, was born(...)`" matches the fact David Beckham –/PEOPLE/PERSON/CHILDREN→ Brooklyn Beckham, since "his" matches the subject through a co-reference, "Brooklyn" overlaps with a label of Brooklyn Beckam, and the path "his first son" matches the predicate through our WordNet-based expansions.

5.5 Optimization

The matching methods in Reverse Extraction are designed to capture as many mentions of entities and predicates as possible. The output may contain many false positives, due to the ambiguity of mapping mentions to entities, relations, and literals. Therefore, in the second step called Optimization, we perform joint constraint solving to identify a large set of consistent facts subject to disambiguation and fact dependency constraints.

Consistency Constraints

Disambiguation Constraint on Entities. The sentence *Eastwood's landmark role in Dirty Harry was critically acclaimed* can have two matches: Eastwood –PLAYEDIN→ Dirty_Harry and Eastwood –PLAYEDROLE→ Harry_Callahan. However, as a single text position (*Dirty Harry*) can mean only one entity (either the movie Dirty_Harry or the character Harry_Callahan) and the movie title being the better match, only the first fact should be considered spotted.

Dominant Sense Constraint on Mentions. Word sense disambiguation benefits from the assumption that a discourse uses only one sense of a word [54]. For example an ambiguous word *plant* within a document means either an industrial plant (e.g., a power plant) or a plant in the biological sense; these two senses are never mixed using the same word within the same document. Applying such constraints to our problem helps us avoid spurious matches. For example, the mention *Zidane* can mean the famous soccer player Zinedine Zidane or his wife Véronique, but in a biography of the soccer player his wife would always be referred to by her first name or by her full name. Such a constraint should not be applied blindly in every situation. For example, *Oscar* in a biography of Woody Allen refers to many distinct awards (Best Picture, Best Director, etc.). We therefore enforce the outlined single dominant sense constraint only to selected entities, namely, the topic entities of a document (e.g., the person whose biography we are looking at). If such an entity, for example, Zinedine Zidane, is matched to a mention (*Zidane*), then it is the only entity that other mentions of *Zidane* can be disambiguated to. We will call such entities *capturing entities*.

It is worth noting that not all identical mentions must be assigned. This is desirable, because mentions can overlap and we require that overlapping text positions are assigned to at most one entity. For example, the mention *Véronique Zidane* overlaps with *Zidane*, and we cannot match them to different entities – therefore this particular sub-mention (*Zidane*) will remain unmatched, while other mentions of *Zidane* will be mapped to Zinedine Zidane.

Dependent-Central Constraint on Facts. The sentence *After winning the closely contested 56th presidential election, Obama assumed office in 2009* can incorrectly match Obama –WONAWARDIN→ 2009 known to Freebase. We know, however, that such a fact requires a related fact Obama –WONAWARD→ Nobel_Peace_Prize, which is not stated, and therefore we should not output the fact about winning an award in 2009.

In our experiments we consider dates and numbers in compound facts as dependent on the occurrence of any other fact derived from the same compound facts.

To recap, the dependent-central constraint disallows spotting the secondary fact unless we have also spotted the central fact.

ILP Formulation

In order to enforce the constraints described above, we develop an integer linear program (ILP), which selects a consistent set of facts from the output of the Reverse Extraction stage (for a practical discussion of encoding various problems as integer linear programs see the tutorial [21]). We introduce binary variables $m_{l,e}$ for matches $match(l, e)$ between textual positions and entities (or dates, values, etc.). The variables are set to 1 if the match is deemed correct and 0 otherwise. Matches have weights $w_{l,e}$ coming from the Reverse Extraction stage, contributing $\sum_{l,e} m_{l,e} w_{l,e}$ to the objective function in the ILP. To enforce that overlapping positions cannot be matched to different entities we introduce the following constraint:

$$m_{l,e} + m_{l',e'} \leq 1 \text{ if } e \neq e' \text{ and } l \text{ overlaps } l'$$

Each position has a mention $text.substringAt(location)$, and some of them have the same mention. For each mention s and entity e we define a binary variable $y_{s,e}$. The variable must be 1 when some position of mention s is assigned to e. To enforce this we add the constraints

$$y_{s,e} \geq x_{l,e} \quad \text{forall positions } l \text{ of mention } s.$$

To enforce that if a mention is assigned to a capturing entity e, then it cannot be assigned to another entity e', we use the following constraints:

$$y_{s,e} \leq 1 - y_{s,e'}.$$

We define two types of fact occurrence variables:

$$f_{l1,l2,path} \text{ for occurrence in a particular position, and}$$
$$f \text{ for occurrence anywhere in the document.}$$

The variables are connected by the logical dependency $f \iff \bigvee_{l1,l2,path} f_{l1,l2,path}$, which can be translated to:

$$f \geq f_{l1,l2,path} \quad \text{for all } l_1, l_2, path, \text{and}$$
$$f \leq \sum_{l1,l2,path} f_{l1,l2,path}.$$

Our constraints can "switch off" some of the overlapping positions of entity matches, so we retain fully matched fact occurrences only if their subject and object are retained, i.e., $f_{l1,l2,path} \iff m_{l1,s} \wedge m_{l2,o}$ where $f = f(s, p, o)$, and $path$ between $l1$ and $l2$ matches p. In the ILP this is expressed by the constraints:

$$f_{l1,l2,path} \leq m_{l1,sub},$$
$$f_{l1,l2,path} \leq m_{l2,obj},$$
$$f_{l1,l2,path} \geq m_{l1,sub} + m_{l2,obj} - 1.$$

We associate such a fact occurrence with the weight of the match between the path (obtained by deep parsing) and the predicate of the fact (plus subject and object match weights), so that we preferentially match facts if their predicate matches the path well. The overall objective function therefore takes the form

$$\max \sum_{l,e} m_{l,e} \cdot w_{l,e} + \sum_{l1,l2,path} f_{l1,l2,path} \cdot (w_{path,p} + w_{l1,s} + w_{l2,o})$$

The dependency between central and dependent facts $f_{dep} \implies \bigvee f_{cent}$ is expressed in the ILP as

$$f_{dep} \leq \sum f_{cent}.$$

We use commercial Gurobi software package [59] to solve the ILP.

5.6 Resurrection and Ranking

While the Reverse Extraction stage of our three-phase method aims at high recall, the Optimization stage prunes the candidate to achieve high precision. However, the Optimization may sometimes be too stringent and thus lose good facts. This motivates the third stage of our approach, coined *Resurrection*. Our approach to recover low-scoring facts that were dropped by the Optimization stage is based on the idea that in coherent text facts typically occur together with semantically highly related "co-facts".

Co-facts are KB facts which share entities or the relation. If two facts have subject s in common, we refer to them as *subject co-facts*, that is $\langle s, p, o\rangle$ and $\langle s', p', o'\rangle$ are subject co-facts if $s = s'$. For example, f_1:DavidBeckham –PARENT→ TedBeckham and f_2:DavidBeckham –PARENT→ SandraBeckham are subject co-facts as they are about the same subject. Analogously, facts $\langle s, p, o\rangle$ and $\langle s', p', o'\rangle$ are *predicate co-facts* if $p = p'$, and *object co-facts* if $o = o'$. We also define subject-object co-facts if $s = o'$ or $s' = o$. Note that facts can be co-facts in more than one way. In the example above, f_1 and f_2 are not only subject co-facts but also predicate co-facts.

Let S be the set of facts found by Reverse Extraction (stage 1) and $J \subseteq S$ be the set of facts found by the Optimization step (stage 2). The set J' of facts which are reconsidered in the Resurrection step is

$$J' := \{f \in S \setminus J : \exists g \in J \text{ s.t. } cofacts(f, g)\},$$

that is, the set of facts found by Reverse Extraction that were rejected by Optimization but at least one of their co-facts was accepted. For example, if Optimization accepts DavidBeckham –PLAYEDFOR→ RealMadrid, but rejects DavidBeckham –PLAYEDFOR→ ManchesterUnited and ZinedineZidane –PLAYEDFOR→ RealMadrid, then the Resurrection stage would give the latter two facts a second chance.

Note that we can make this approach more liberal by reconsidering all of the subject co-facts, predicate co-facts, and object co-facts, or we can configure the Resurrection phase more conservative by reconsidering only co-facts of accepted facts that are co-facts with regard to two of the three roles, e.g., subject co-fact and predicate co-fact or predicate co-fact and object co-fact in the above

example. In our experiments in Section 5.7, we configure the Resurrection phase to compute all co-facts that share their subject and predicate or their subject and object with an accepted fact.

Ranking

We have not yet stated how exactly these resurrected candidates get a second chance. One option is to accept all resurrected co-facts. We report on the quality of this option in our experiments.

An alternative is to treat the resurrected facts as another candidate pool for further assessment. Obviously, it does not make sense to run another reasoning algorithm on them; whatever additional criteria we would devise here, could have been integrated into the Optimization stage already. What we do instead is to pool the already accepted facts and the resurrected co-facts together and compute a *confidence ranking* on them, using PageRank-style methods.

To this end, we construct a graph $G(V, E)$ of spotted facts, with vertex set $V = \{J \cup J'\}$ and edge set $E = \{e(f, f') \mid cofacts(f, f')\}$. Thus the vertices include both the previously accepted facts and their resurrected co-facts. We initialize each vertex with fact scores from Reverse Extraction as in Eq. 5.1. On this graph we run random walks with restarts as described below.

We start at a randomly chosen vertex that corresponds to an accepted fact. We choose outgoing edges uniformly at random and this way follow a path in the graph. At each vertex reached, we randomly: with probability ϵ set to a small value (e.g., 0.1) jump to one of the accepted-fact vertexes and restart the walk; or with probability $1 - \epsilon$ we continue the current walk. This process is conceptually repeated many times (but can be computed with the Jacobi power iteration method), to approximate the stationary visiting probability of all nodes. This probabilities are the basis for ranking all facts: previously accepted ones and resurrected co-facts. This is essentially the Personalized PageRank method [61], applied and tailored to our co-facts graph.

From the final ranking computed this way, we obtain additional facts by setting a cut-off point and accepting all facts above that rank. One way of choosing the cut-off position is to pick the lowest-ranked fact that was already accepted by the Optimization stage and accept all co-facts above. Other criteria are conceivable. In our experimental evaluation, we report ranking-quality measure (MAP) on the precision-recall curve that avoid having to choose a cut-off point at all.

5.7 Evaluation

Our experimental evaluation focuses on qualitative evaluation of the proposed spotting methods. To this end we present evaluation measuring not only precision, but more importantly, also recall of individual methods, as well as comparison against a baseline approach. Since measuring recall involves ground-truth generation by laboriously listing all possible facts within a document, we limit this experiment to a sample of biographies derived from the Web. Next, we perform an experiment to determine the effectiveness of our spotting method

on real-world KB curation scenario. We test this by extending Freebase with hand-compiled facts and report spotting results on recent news articles.

Setup. In order to measure the performance of our fact spotting methods we created a test collection of biographies of famous actors and soccer players. We started with a set of 233 names of soccer players and actors. Queries following a pattern `<name> biography` were submitted to the Bing search engine. In the results we identified a set of 10 web sites that feature biographies of famous people, with highly varying styles across sites. We manually inspected them to make sure that they are not copies of Wikipedia pages or simple lists of facts without narrative structure. We kept only the results from those web sites and obtained a set of 848 biographies of 233 entities, about which Freebase knows 20548 compound facts. To remove the HTML markup from the documents we used BeautifulSoup software[2]. The 10 web sites, each with the number of biographies for our collection, are listed in Table 5.3.

To create the ground truth for fact spotting we manually annotated 12 documents with facts indicating their presence or absence anywhere in the document. To ensure that our results are not affected by remaining non-essential parts of their source websites, such as advertising, we manually finished cleaning the text of documents that required it. In total the 12 documents contained 356 ground truth triple facts (compared to the total 2432 triple facts about our test entities known to Freebase).

Note that the ground-truth annotation process is not always straightforward. The documents and Freebase can state the same facts at different detail level. For example, while a document says "won multiple Academy Awards" Freebase may list all of them. Only explicitly stated facts were considered for the ground truth, that is, we did not perform any reasoning. For example, the fact that somebody *won* an Academy Award implies that he or she was also *nominated*, but we did not mark such facts as present, unless they were stated explicitly.

Measures. The performance of our fact spotting methods is evaluated with respect to the manually annotated ground truth. Since all facts returned by fact spotting originate from the knowledge base, they are all correct facts about the entities, but may not correspond to what is stated in the document. The two kind of mistakes that our fact spotting methods make are: *1*) inclusion of spurious facts not stated in the document, and *2*) missing facts which are stated in the documents. Therefore, we define precisions as and recall as:

$$\text{precision} = \frac{|\ \text{found} \cap \text{in document}\ |}{|\ \text{found}\ |},$$
$$\text{recall} = \frac{|\ \text{found} \cap \text{in document}\ |}{|\ \text{in document}\ |}.$$

Baseline method. A natural baseline for fact spotting is to only consider the entities and report the fact as present if both the subject and object are found, irrespective of the predicate. More formally, if the knowledge base knows a fact $f = \langle s, p, o \rangle$ and we find s and o in the same sentence, then we consider f found. Such approach is often applied in relation extraction methods to generate training samples.

[2] `http://www.crummy.com/software/BeautifulSoup/`

			Baseline				Speculation Method			
Biography	**GT**	**KB**	**Fnd.**	**Pr.**	**Rec.**	**F1**	**Fnd.**	**Pr.**	**Rec.**	**F1**
Arnold Schwarzenegger - msn	41	302	112	33.0	90.2	48.4	40	45.0	43.9	44.4
Clint Eastwood - msn	40	307	134	29.9	100.0	46.0	51	60.8	77.5	68.1
David Beckham - hos	42	269	117	35.9	100.0	52.8	49	57.1	66.7	61.5
David Beckham - tbc	48	269	144	33.3	100.0	50.0	67	46.3	64.6	53.9
Elizabeth Taylor - tbc	44	247	93	47.3	100.0	64.2	45	75.6	77.3	76.4
Gianluigi Buffon - hos	18	62	34	52.9	100.0	69.2	16	75.0	66.7	70.6
Jodie Foster - msn	31	204	72	40.3	93.5	56.3	28	57.1	51.6	54.2
Oliver Kahn - hos	27	71	40	65.0	96.3	77.6	12	91.7	40.7	56.4
Pelé - bio	15	108	33	33.3	73.3	45.8	12	66.7	53.3	59.3
Pelé - hos	9	108	30	30.0	100.0	46.2	10	70.0	77.8	73.7
Woody Allen - tbc	19	346	104	18.3	100.0	30.9	25	32.0	42.1	36.4
Zinedine Zidane - bio	22	139	64	34.4	100.0	51.2	28	53.6	68.2	60.0
Aggregated (sum / micro avg.)	356	2432	977	35.3	**96.9**	51.8	383	57.2	61.5	59.3

Table 5.1: Evaluation of fact spotting methods (part 1). Number of facts in ground truth (GT), Freebase (KB), facts found (Fnd.), precision (Pr.), recall (Rec.), F1 of baselines and our proposed methods. Biography sources: biography.com (bio), history-of-soccer.org (hos), movies.msn.com (msn), thebiographychannel.co.uk (tbc).

	Reverse Extraction				Optimization				Resurrection			
Biography	**Fnd.**	**Pr.**	**Rec.**	**F1**	**Fnd.**	**Pr.**	**Rec.**	**F1**	**Fnd.**	**Pr.**	**Rec.**	**F1**
Arnold Schwarzenegger - msn	90	37.8	82.9	51.9	48	47.9	56.1	51.7	64	50.0	78.0	61.0
Clint Eastwood - msn	83	44.6	92.5	60.2	55	67.3	92.5	77.9	62	59.7	92.5	72.5
David Beckham - hos	92	44.6	97.6	61.2	66	62.1	97.6	75.9	73	56.2	97.6	71.3
David Beckham - tbc	110	35.5	81.3	49.4	68	50.0	70.8	58.6	91	42.9	81.3	56.1
Elizabeth Taylor - tbc	69	55.1	86.4	67.3	41	78.0	72.7	75.3	51	70.6	81.8	75.8
Gianluigi Buffon - hos	29	55.2	88.9	68.1	19	78.9	83.3	81.1	21	71.4	83.3	76.9
Jodie Foster - msn	58	46.6	87.1	60.7	39	64.1	80.6	71.4	51	52.9	87.1	65.9
Oliver Kahn - hos	30	66.7	74.1	70.2	20	100.0	74.1	85.1	21	95.2	74.1	83.3
Pelé - bio	18	50.0	60.0	54.5	13	61.5	53.3	57.1	15	53.3	53.3	53.3
Pelé - hos	19	47.4	100.0	64.3	12	50.0	66.7	57.1	12	50.0	66.7	57.1
Woody Allen - tbc	63	28.6	94.7	43.9	37	37.8	73.7	50.0	56	32.1	94.7	48.0
Zinedine Zidane - bio	51	41.2	95.5	57.5	29	65.5	86.4	74.5	34	61.8	95.5	75.0
Aggregated (sum / micro avg.)	712	43.4	86.8	57.9	447	**61.3**	77.0	**68.2**	551	54.4	84.3	66.2

Table 5.2: Evaluation of fact spotting methods (part 2). See Table 5.1 for details.

Web domain	# Biographies
www.netglimse.com	180
people.famouswhy.com	151
www.biography.com	125
movies.msn.com	100
www.thebiographychannel.co.uk	91
www.footballtop.com	65
www.soccer-fans-info.com	41
www.history-of-soccer.org	40
www.footballteamplayers.com	35
www.biographybase.com	20

Table 5.3: Sources of our biography collection.

5.7.1 Results

The results of our experiments are presented in Tables 5.1 and 5.2. The baseline, which performs only entity matching, achieves recall of 96.9%. This shows that our entity matching methods deliver high recall, rarely missing entities stated in text. Reverse Extraction, which extends the simple baseline by matching relations to text, improves the precision by 8%. However, a precision of 43.4% is still low, leaving room for improvement. Optimization produces a large jump in precision with slight drop in recall, delivering the highest F1 measure among all proposed methods and baselines. Since we aim at both precision and recall, we therefore consider it the best method overall. Optimization also outperforms Speculation method on both precision and recall. Resurrecting co-facts after Optimization presents a compelling alternative for applications that require higher recall of spotted facts, as it achieves higher recall compared to Optimization and second highest F1.

While the speculation method is outperformed by Optimization, it does not rely on Stanford CoreNLP and Gurobi software, and therefore can be useful when a lightweight approach is needed. The method is also fairly fast: processing all 848 biographies (i.e., not just the annotated samples for which we have ground-truth) took only 134 seconds in total.

Table 5.4 reports mean average precision for the set of facts obtained by ranking the output of Reverse Extraction and Optimization. Fig. 5.3 shows precision-recall curves for selected documents. The basis for comparison is the performance of the Reverse Extraction with initial scores given by Eq. 5.1. By pruning incorrect facts in Optimization we improve MAP and are able to improve the precision-recall curve. The results are further improved by the application of our PageRank-based ranking. The highest MAP value is obtained by Resurrection with ranking. In terms of precision-recall curve the precision of the Resurrection method is close to Optimization, but brings additional recall.

Results per Relation

We performed an experiment to determine the quality of our WordNet based relation matching method. Table 5.5 presents spotting results aggregated per

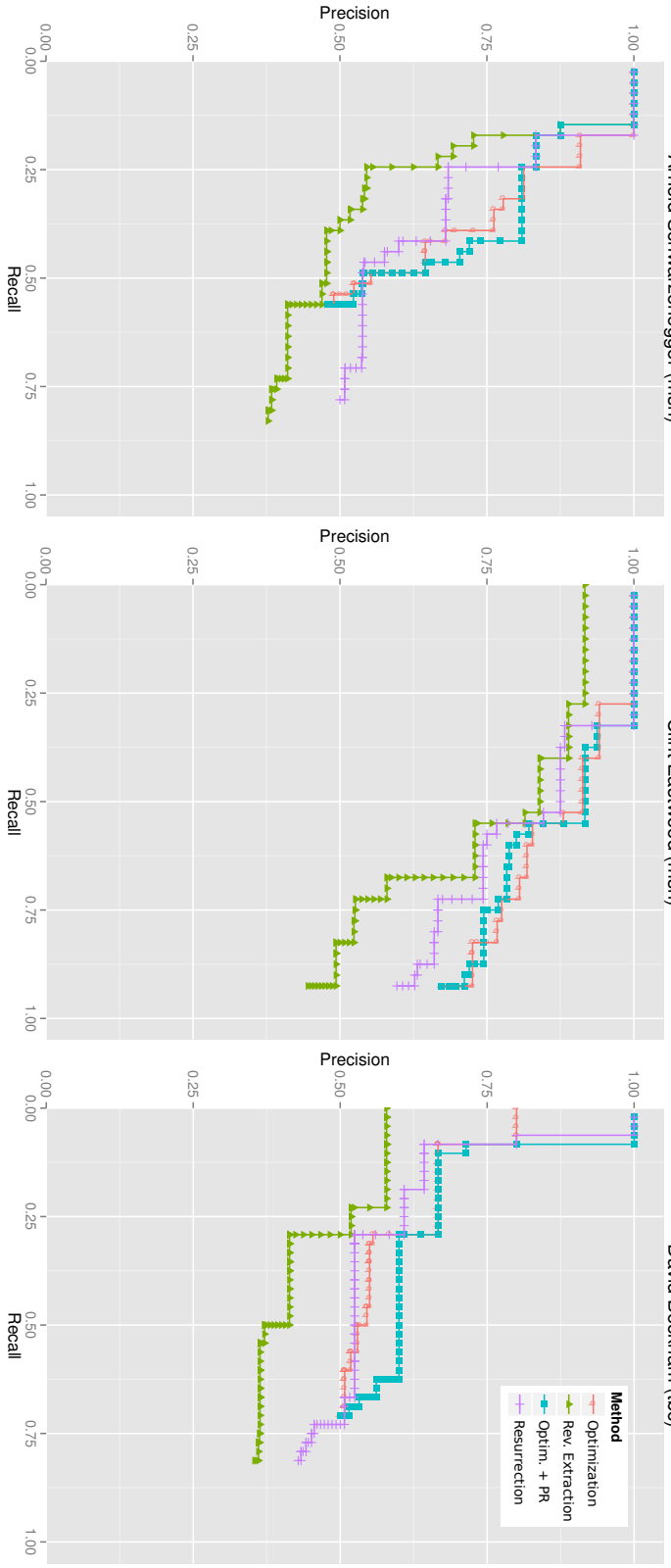


Figure 5.3: Precision-recall curves for fact spotting on selected documents.

	No Ranking		With Ranking		
Biography	**R.E.**	**Opt.**	**R.E.**	**Opt.**	**Res.**
Arnold Schwarzenegger - msn	0.463	0.441	0.465	0.449	0.535
Clint Eastwood - msn	0.638	0.759	0.737	0.811	0.777
David Beckham - hos	0.482	0.587	0.567	0.632	0.622
David Bbeckham - tbc	0.324	0.383	0.401	0.450	0.447
Elizabeth Taylor - tbc	0.657	0.704	0.664	0.702	0.674
Gianluigi Buffon - hos	0.685	0.657	0.759	0.712	0.712
Jodie Foster - msn	0.406	0.534	0.427	0.543	0.513
Oliver Kahn - hos	0.647	0.740	0.726	0.740	0.740
Pele - bio	0.474	0.437	0.488	0.461	0.450
Pele - hos	0.640	0.418	0.665	0.429	0.429
Woody Allen - tbc	0.500	0.418	0.610	0.519	0.615
Zinedine Zidane - bio	0.533	0.669	0.568	0.706	0.695
MAP	0.538	0.562	0.590	0.596	0.601

Table 5.4: Average precision before/after ranking of Reverse Extraction (R.E.), Optimization (Opt.), and Resurrection (Res.) results.

relation. Due to the large number of relations present in our dataset, we selected only those which occur at least 5 times in our ground truth annotations. The results are ordered by the F1 score of the optimization spotting method. In case of relations that feature compound objects, such as /PEOPLE/PERSON/SPOUSE_S, which groups together all information about a marriage, we count each attribute as a separate relation, e.g. /PEOPLE/PERSON/SPOUSE_S: TO.

The facts which are the easiest to spot are those which contain easy to disambiguate objects (name of the spouse, date of birth, titles of movies in which an actor played). Such facts also contain clearly defined relations which are stated in the text in a straightforward manner. On the opposite end of the spectrum are facts with relations like /AWARD/AWARD_NOMINEE/AWARD_NOMINATIONS:-AWARD, where there is often ambiguity in object names and the relation is stated in a complex way. Comparison between the Reverse Extraction and Optimization shows, that we are able to prune incorrect fact candidates, without much affecting the correctly identified facts.

5.7.2 Disambiguation Performance

Our Optimization method disambiguates entities on its own within the fact spotting system instead of using an external component; we are therefore interested in verifying how well we perform against the state of the art. For the reason outlined below, our fact spotting technique can improve over some aspects of NED. Systems, such as AIDA [63] rely on consistency links between named entities. Several similarity measures based on text, links, and keyphrases have been studied in literature. Since our aim is to spot entities participating in facts, our fact spotting uses relations from the facts as natural consistency links between entities. Therefore, our disambiguation component benefits not only from analyzing noun phrase mentions, but also from considering relations

		Baseline			Speculation			Reverse Extr.			Optimization			Resurrection		
Relation	**GT**	**Fnd.**	**Cor.**	**F1**	**Fnd.**	**Cor.**	**F1**	**Fnd.**	**Cor.**	**F1**	**Fnd.**	**Cor.**	**F1**	**Fnd.**	**Cor.**	**F1**
/film/director/film	11	16	11	81	5	5	63	12	11	96	12	11	96	12	11	96
/people/person/date_of_birth	9	9	9	100	9	9	100	9	9	100	8	8	94	8	8	94
/soccer/football_player/statistics: team	17	21	17	89	9	9	69	20	17	92	20	17	92	20	17	92
/people/person/spouse_s: spouse	11	18	11	76	13	11	92	16	10	74	11	10	91	11	10	91
/soccer/football_player/matches_played: team	5	5	5	100	3	3	75	5	5	100	4	4	89	5	5	100
/film/producer/film	8	11	8	84	8	6	75	10	8	89	10	8	89	10	8	89
/film/actor/film: film	22	33	22	80	17	15	77	28	22	88	26	21	88	28	22	88
/people/person/place_of_birth	8	8	7	88	7	7	93	8	7	88	8	7	88	8	7	88
/sports/pro_athlete/teams: team	25	30	25	91	17	16	76	29	24	89	28	23	87	29	24	89
/sports/pro_athlete/teams: from	17	18	17	97	18	17	97	15	14	87	14	13	84	15	14	87
/people/person/parents	8	23	8	52	5	3	46	22	7	47	8	6	75	9	6	71
/people/person/spouse_s: to	6	8	6	86	8	6	86	7	5	77	5	4	73	6	5	83
/people/person/nationality	5	13	5	56	9	4	57	9	5	71	7	4	67	9	5	71
/soccer/football_player/statistics: from	14	17	14	90	9	8	70	12	9	69	11	8	64	12	9	69
/people/person/children	12	52	12	38	7	7	74	40	12	46	10	7	64	18	12	80
/soccer/football_player/statistics: to	9	16	9	72	14	7	61	11	6	60	10	6	63	11	6	60
/award/award_winner/awards_won: year	5	29	5	29	25	5	33	23	5	36	12	5	59	18	5	43
/sports/pro_athlete/teams: to	10	22	10	63	22	10	63	18	8	57	17	8	59	18	8	57
/people/person/places_lived: location	5	18	5	43	0	0	-	13	5	56	9	4	57	12	5	59
/award/aw._winner/awards_won: honored_for	7	19	7	54	18	6	48	14	6	57	14	6	57	14	6	57
/people/person/spouse_s: from	8	11	8	84	11	8	84	8	6	75	6	4	57	8	6	75
/award/award_winner/awards_won: award	8	21	8	55	3	3	55	15	8	70	13	6	57	15	8	70
/people/person/profession	15	31	12	52	15	9	60	25	10	50	15	8	53	20	9	51
/common/topic/alias	7	45	5	19	0	0	-	20	4	30	3	2	40	7	2	29
/award/aw._nominee/award_nominations: award	5	14	3	32	3	1	25	5	2	40	5	2	40	5	2	40
/award/aw._nominee/aw._nominations: nom._for	7	26	7	42	23	6	40	9	3	38	7	2	29	8	3	40
/award/competitor/competitions_won	7	7	7	100	0	0	-	7	7	100	1	1	25	7	7	100

Table 5.5: Results per relation: number of facts in the ground truth (GT), found (Fnd.), correct among found (Cor.), and F1 (%). The results are limited to relations where the ground truth contains at least 5 facts and are sorted by F1 in Optimization method.

		AIDA		Spotting		A. on FS.	
Biography	**Total**	**fnd.**	**cor.**	**fnd.**	**cor.**	**fnd.**	**cor.**
Arnold Schwarzegger	87	52	41	66	52	31	23
Clint Eastwood	94	54	48	68	47	28	27
David Beckham(hos)	106	72	63	72	42	38	30
David Beckham (tbc)	226	184	133	113	69	71	52
Elizabeth Taylor	81	44	26	53	47	16	11
Gianluigi Buffon	51	47	42	24	21	20	18
Jodie Foster	77	40	33	50	40	13	11
Oliver Kahn	48	41	33	18	16	11	9
Pelé (bio)	53	41	25	17	11	5	2
Pelé (hos)	44	30	24	24	22	10	9
Woody Allen	83	61	50	42	23	20	17
Zinedine Zidane	81	72	52	41	34	32	23
Total	1031	738	570	588	424	295	232

Table 5.6: Disambiguation with Fact Spotting vs AIDA. Reported are found and correctly disambiguated mentions. *A. on FS.* columns contain the results of AIDA limited to the mentions that were resolved by FSpot.

in text. For example, in the sentence "David Beckham played for Madrid" Madrid should be disambiguated to the football club *Real Madrid*, and not the city *Madrid*, but this can be achieved only if the phrase "played for" is taken into account. Moreover, this allows for flexible entity-entity relatedness scores depending on the strength of match between the relation and the text (unlike AIDA where entity-entity relatedness is globally fixed). Disambiguation within fact spotting can also benefit from consistency constraints on facts, such as our dependent-central constraints.

Although the objective function in the ILP based joint solving step allows mapping locations to entities even if they do not participate in any fact, we filter out such locations as they are likely to be noisy. Our disambiguation involves resolving dates and literals in conjunction with named entities, which is out of scope for state-of-the-art NED systems, we therefore do not consider such disambiguations in the comparison (fact spotting can tell us that a particular mention of "23" was David Beckham's jersey number when he played for LA Galaxy).

We ran AIDA on the collection of biographies and manually assessed correctness of disambiguated mentions. The results of our evaluation are presented in Table 5.6. For the total number of mentions we take the union of results found by both methods. For both AIDA and our fact spotting method (FSpot) we report the number of mentions found and how many of them were correct. We also report the performance of AIDA on mentions which were found by FSpot (shown in the "AIDA on FSpot" column).

The results show that fact spotting performs better than AIDA at locations where facts are stated. This justifies using our own NED component and also shows that considering relations between entities as expressed by facts is beneficial

	OLLIE			Optimization	
Biography	**Extr.**	**Valid**	**Corr.**	**Spotted**	**Corr.**
Arnold Schwarzegger	101	21	6	48	23
Clint Eastwood	134	27	15	55	37
David Beckham(hos)	160	25	11	66	41
David Beckham (tbc)	261	38	15	68	34
Elizabeth Taylor	141	40	22	41	32
Gianluigi Buffon	76	24	14	19	15
Jodie Foster	89	15	8	39	25
Oliver Kahn	57	15	8	20	20
Pelé (bio)	109	17	14	13	8
Pelé (hos)	70	8	6	12	6
Woody Allen	85	4	3	37	14
Zinedine Zidane	80	21	8	29	19
Total	1363	255	130	447	274

Table 5.7: Fact spotting with OLLIE vs Our approach (Optimization method).

to NED. However, AIDA also disambiguates locations that do not participate in facts, therefore delivering higher recall.

5.7.3 Comparison against Open IE

One could argue that fact spotting can be achieved in a simpler way by first disambiguating entities with an off-the-shelf NED system and then running an IE tool to extract facts from the entity-annotated text. In order to evaluate how many facts can be spotted this way we ran AIDA on the biography dataset and passed the entity-disambiguated text through the Open Information Extraction tool OLLIE [90]. OLLIE analyzes sentence parse structures using trained classifiers to construct meaningful fact triples from the sentence sub-structures (see Chapter 2 for more information). We manually inspected the output of this AIDA-OLLIE pipeline on the biography data.

The results, contrasted with our Optimization method, are shown in Table 5.7. Although OLLIE found a large number of triples, many of these extractions did not result in valid $\langle s, p, o \rangle$ facts, as OLLIE often picked up meaningless phrases for the predicates. Among the valid triples, only a few were spotted facts from Freebase. Table 5.7 clearly shows the superiority of the Optimization method in terms of the number of spotted and correct facts. The experiment therefore shows that standard IE would be a poor performer in the task of fact spotting.

5.7.4 Fact Spotting for KB Curation

The aim of fact spotting in the use-case of KB curation is to reduce the workload of the human curator. To this end, our tool should help to confirm or refute facts. For true facts, this amounts to finding additional evidence. The system retrieves (e.g., via a search engine) and analyzes supplementary sources about the entity of choice and finds mentions of the fact of interest. These mentions would then

	Documents / With Mentions / Useful		
David Beckham –PLAYSFOR→ Chelsea F.C.	16	10	0
Larry Ellison –CEOOF→ SAP	15	9	7
Steve Jobs –BORNIN→ Cupertino	13	3	0
Steve Jobs –CEOOF→ Google	23	5	3
Oliver Kahn –PLAYSFOR→ Brazil	9	8	2
Charles Manson –MARRIEDTO→ Sharon Tate	22	17	16
John McCain –BORNIN→ Arizona	11	4	0
John McCain –GRADUATEDFROM→ Harvard	1	0	0
Total	110	56	28

Table 5.8: KB curation experiment with made-up facts. We report the number of sample documents which mention the subject and object in proximity, the number of documents in the sample in which we found alternative version of the fact, and how many of them can be used to refute the fact.

be shown to the curator for the final assessment. Note that we can incorporate a trustworthiness/authority model for sources like newspapers, blogs, users in review forums, etc. (e.g., [66]). This is orthogonal to fact spotting, and therefore not considered here.

To understand how fact spotting is useful for false facts, we make the assumption that false facts are in the KB because of incorrect extraction from the source document or erroneous input by crowdsourcing workers (but not say an adversary intentionally polluting the KB). For example, the sentence "Larry Ellison has two children with his third wife Barbara Boothe" may lead to the incorrect extraction of Larry Ellison –HASCHILD→ Barbara Boothe. In such cases, the curator would like to compare the source of the (erroneous) fact with other sources for the same fact or variants of the fact including the correct "version" of the fact. Therefore, we decompose the fact triple and search for pairs of its three constituents, to gather a set of supplementary sources. Then we show the mentions of fact variants spotted in these sources to the curator for his or her assessment. In the following experiment, we thus evaluate usefulness for the curator. A mention is *useful* if it either confirms a fact or refutes it by showing the true variant of the fact.

The results of the KB curation study are shown in Table 5.8. We started with a set of made-up facts and a large corpus of about 2 million news document annotated with entities. From the corpus we retrieved 110 documents which contain subjects and objects of the facts in textual proximity. The fact spotting system matched facts in some of the documents. We then manually verified, whether any of the mentions can be used to refute the fact in doubt. This refutation requires common sense reasoning from the curator, for example, inferring that someone does not work for his business competitor from the sentence "For most of Ellison's tenure, Oracle has battled with big rivals IBM and SAP". On the other hand the fact David Beckham –PLAYSFOR→ Chelsea should not be rufeted from similar snippet, since football players can play for multiple clubs during their careers. Our study demonstrates that fact spotting can be used to reduce the workload on the curator by selecting promising

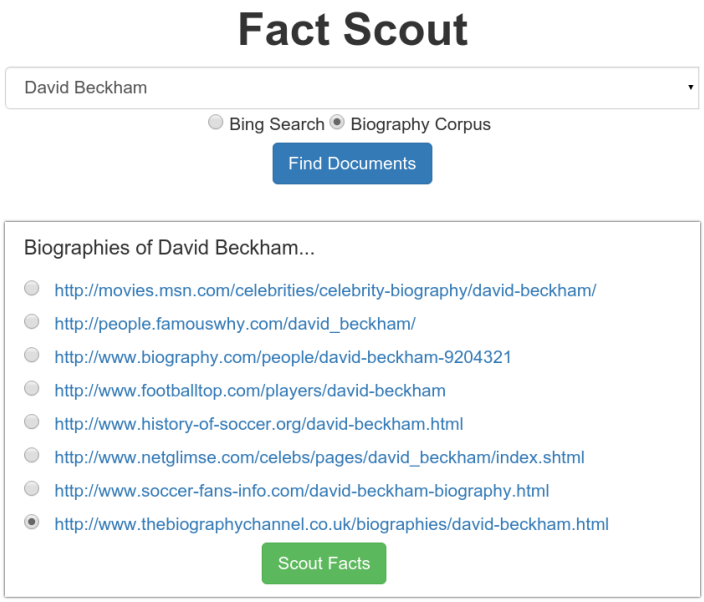


Figure 5.4: Query interface of FactScout. The user can choose between Web search results and locally stored corpus of documents.

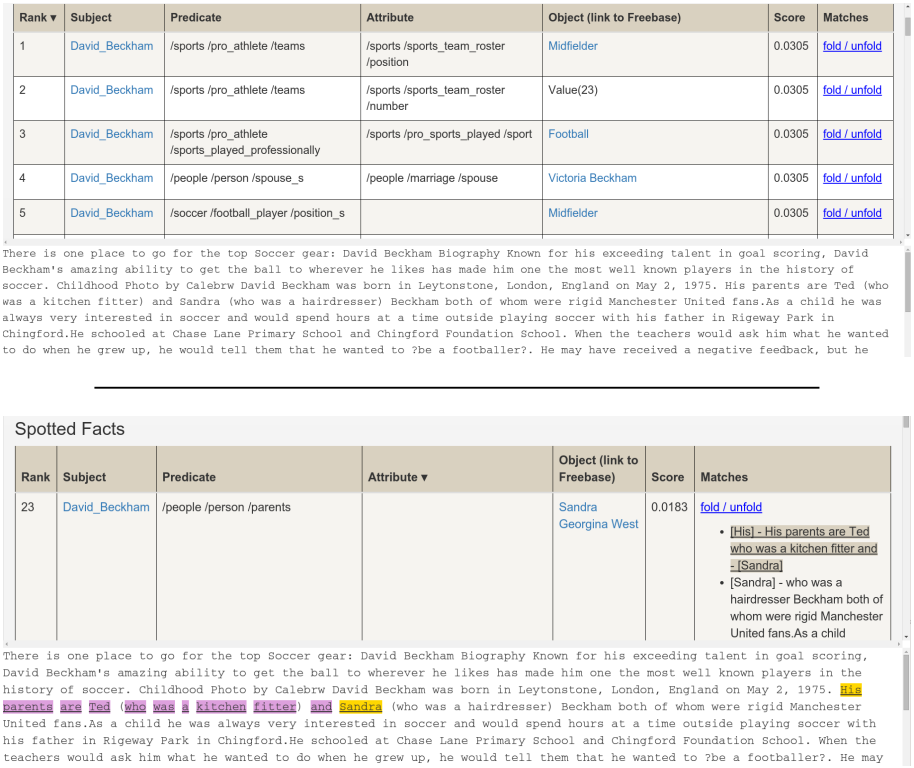
mentions of facts for further analysis.

5.8 System Demonstration

To further demonstrate how fact spotting can be used for knowledge base curation, we developed a system, called FactScout, which spots facts in documents and presents their locations to the user. The system emulates a typical KB curator's workflow and is therefore entity-centric. Fact spotting is performed on entities of user's interest, within documents on the Web or in a document collection. For demonstration purposes, we focus on spotting Freebase facts about people in biographies, news, and other documents. Users can interactively query Freebase entities and pick documents of their interest. The system will then run fact spotting on the text and produce an annotated document with the spotted facts as the result.

A screenshot of the query interface is shown in Fig. 5.4. A query to the system consists of a Freebase entity and a document about it. The interface provides a drop down list populated by Freebase entities. To pick a document relevant to the chosen entity, users can perform Web search using the interface. Alternatively, they can pick a document from a list of biographies provided by the system. FactScout then spots occurrences of Freebase facts about the entity in the document (i.e., facts for which the query entity is the subject).

Fig. 5.5 shows fact spotting results page of FactScout. A table lists the spotted facts, mentions of subject and object entities, and text snippets which match predicates. The list is sorted by the scores obtained from our PageRank-based



Rank ▾	Subject	Predicate	Attribute	Object (link to Freebase)	Score	Matches
1	David_Beckham	/sports /pro_athlete /teams	/sports /sports_team_roster /position	Midfielder	0.0305	fold / unfold
2	David_Beckham	/sports /pro_athlete /teams	/sports /sports_team_roster /number	Value(23)	0.0305	fold / unfold
3	David_Beckham	/sports /pro_athlete /sports_played_professionally	/sports /pro_sports_played /sport	Football	0.0305	fold / unfold
4	David_Beckham	/people /person /spouse_s	/people /marriage /spouse	Victoria Beckham	0.0305	fold / unfold
5	David_Beckham	/soccer /football_player /position_s		Midfielder	0.0305	fold / unfold

Spotted Facts

Rank	Subject	Predicate	Attribute ▾	Object (link to Freebase)	Score	Matches
23	David_Beckham	/people /person /parents		Sandra Georgina West	0.0183	fold / unfold • [His] - His parents are Ted who was a kitchen fitter and - [Sandra] • [Sandra] - who was a hairdresser Beckham both of whom were rigid Manchester United fans.As a child

Figure 5.5: Result visualization in FactScout: spotted facts and highlighted occurrence of a fact.

ranking model. Hyperlinks to Freebase are provided in case the user needs to quickly consult the knowledge base. The fact spotting results are also visualized as annotations on the document. By hovering over a spotted fact, the textual positions of the subject, predicate, and object are highlighted in the document text.

5.9 Related Work

The task of fact spotting is fairly new, so there is relatively little work that is directly related. Most notably is the work of [94] on managing provenance information for KB facts that are automatically acquired by IE methods. However, this work focused on "Reverse IE" by considering only sources to which the IE machinery was applied. Unlike our approach, they did not consider any supplementary, previously untapped sources. Very recently, Dalvi et al. [37] investigated how to automatically obtain text sources to generate entity glosses for the NELL knowledge base. This work is limited to entities, though; it does not extend to relational facts.

Other technical areas that are somewhat related to the task of fact spotting are discussed in Chapter 2; they include: Information Extraction (Section 2.2), truth finding (Section 2.5), question answering (Section 2.7), and named entity

disambiguation (Section 2.6). None of this work is directly applicable to our problem setting.

5.10 Conclusion

We believe that fact spotting is a key asset in the larger picture of digital knowledge management – complementary to information extraction and of great value to tasks like truth discovery and knowledge base curation. We developed methods for fact spotting and demonstrated their practical strength in use-cases with news feeds and Web contents.

Our future work includes looking deeper into such use-cases: leveraging fact spottings as additional evidence for disambiguating textual statements, and – beyond this – investigating the role of fact spottings for deeper analytics over large text corpora with background knowledge bases.

CHAPTER 6

Conclusion

6.1 Thesis Contribution

Current knowledge bases such as DBpedia [8], Freebase [19], and YAGO [118, 64] are constructed by employing automatic information extraction methods on a variety of structured (e.g., Wikipedia infoboxes, gazetteers) and unstructured sources (e.g., news, product descriptions). The abundance of data they contain can be difficult for a human to navigate, especially if salient facts are intertwined with less important information. In order to make knowledge bases more usable, we need to judge importance of facts they contain and present them to human users in a readable form, which is the problem studied in this dissertation.

We presented two methods for generating semantic snippets, which are short summaries of entities based on their types. The first method selects a maximum weighted independent set of vertices in a graph of types of the entity subject to a cost constraint. The structure of the graph guarantees diversity of the solution and the weighting of the nodes leads to selection of high quality types. A user selectable parameter governs the trade off between choosing specialized and abstract types. Our second method views types as sets. To build a summary of an entity we choose a family of large types (in sense of cardinality) with small intersection. Such formulation leads to a diverse summary which avoids small, and therefore exotic and uninformative types. Since the knowledge base structure alone is often not enough to decide how useful a type is for a summary, we present methods to assess the salience of a type using a human computing platform (Amazon Mechanical Turk) and the edit history of categories in Wikipedia. We implemented a browser for YAGO knowledge base, which demonstrates our summarization algorithms.

We studied the problem of finding known facts in new documents, which we call fact spotting. Our original motivation, which links the problem to summarization in knowledge bases, is to allow gathering statistics about fact occurrences. The frequency of a fact in a corpus of documents can be used as a proxy for its importance. However, obtaining such statistics using information extraction may not be possible, since it focuses only on fact occurrences which can be extracted with high confidence and omits textual occurrences which are too difficult, for example, due to complexity of natural language. The fact spotting problem has multiple other applications that include fact credibility

analysis (truth finding), lifting analysis of documents to the level of facts, knowledge base curation, etc.

We proposed two solutions for fact spotting. In the first one we initially find facts where the subject, object, and predicate are all found, and then speculatively add facts with partial evidence if related facts were found. The speculative element is necessary for improving recall, because our predicate spotting method relies on a paraphrase dictionary with limited coverage. In the second fact spotting method, we first aggressively find relation occurrences to build a set of fact candidates. This step is recall oriented and therefore, to improve precision, we then prune the candidate set in an integer linear program based on dependencies between facts.

6.2 Outlook and Future Work

The particular problem discussed in the thesis which opens interesting research directions is fact spotting. We showed that grouping facts together in Freebase style or augmenting central facts with time and locations in YAGO2 style are beneficial for linking them back to the text, since the dependent facts cannot occur without the corresponding main facts. More research is needed to identify situation where such grouping can be useful and how to exploit it. It would be also valuable to mine such dependencies automatically (see [124] for the case of temporal dependencies between facts).

Speculating on spotting a fact when only subject and object are present yields the path between them as a candidate for a relational paraphrase. This could benefit relation paraphrase dictionaries.

While current day knowledge bases contain only factual statements, they could be extended to support any statement or opinion, for example, that a particular drug causes a side effect *according to* a health forum user. Open information extraction system OLLIE already considers such attributions. Spotting such statements could be useful for navigating large corpora such as online communities or bio-medical research articles.

Since our fact spotting performs named entity disambiguation as a part of joint constraint solving, it could be extended to a full named entity disambiguation (NED) system. Such a system would use facts as consistency links between entities and fall back to an established NED method when they are not available.

We see several future directions for the problem of generating semantic snippets. First, they could potentially include not only types, but also other available facts. The importance of facts could be obtained using fact spotting on a large corpus. Second, our summaries considered only single entities. If we use them in a semantic search engine, we may want to summarize entities jointly and take the query into account. So similar structures of the summaries make them easier to read and they do not repeat information from the query. For example, if we search for Nobel Prize laureates, then the summary should not contain this information as it is implied; if we decide to include nationality it should either appear for all result entities or none. The last issue to consider is personalization. Our intersection based algorithm was motivated by the problem of guessing the entity given the summary, where we assumed that all types could be used.

This is obviously not the case, since some types are exotic or require good knowledge of the domain. We could model the user as a set of types they know and use only these. This would both alleviate the problem of exotic types and help us create personalized summaries, for example, for expert users.

Bibliography

[1] Eugene Agichtein, Luis Gravano. ***Snowball*: extracting relations from large plain-text collections**. Proceedings of the 5th ACM conference on Digital libraries, San Antonio, Texas, USA. ACM, 2000, pp. 85–94.

[2] Rakesh Agrawal, Sreenivas Gollapudi, Alan Halverson, Samuel Ieong. **Diversifying search results**. Proceedings of the Second International Conference on Web Search and Web Data Mining (WSDM 2009), Barcelona, Spain, February 9-11, 2009. Ed. by Ricardo A. Baeza-Yates, Paolo Boldi, Berthier A. Ribeiro-Neto, Berkant Barla Cambazoglu. ACM, 2009, pp. 5–14.

[3] Luis von Ahn, Laura Dabbish. **Labeling Images with a Computer Game**. Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI 2004), Vienna, Austria. ACM, 2004, pp. 319–326.

[4] Omar Alonso, Ricardo Baeza-Yates. **Design and Implementation of Relevance Assessments Using Crowdsourcing**. Proceedings of the 33rd European Conference on Advances in Information Retrieval (ECIR 2011), Dublin, Ireland. Springer-Verlag, 2011, pp. 153–164.

[5] Jesse Alpert, Nissan Hajaj. **We knew the web was big...** Official Google Blog. `googleblog.blogspot.de/2008/07/we-knew-web-was-big.html`. July 2008.

[6] Kemafor Anyanwu, Angela Maduko, Amit P. Sheth. **SemRank: ranking complex relationship search results on the semantic web**. Proceedings of the 14th international conference on World Wide Web (WWW 2005), Chiba, Japan, May 10-14, 2005. Ed. by Allan Ellis, Tatsuya Hagino. ACM, 2005, pp. 117–127.

[7] M. Ashburner et al. **Gene ontology: tool for the unification of biology**. *Nature Genetics* 25 (2000), pp. 25–29.

[8] Sören Auer et al. **DBpedia: a nucleus for a web of open data**. Proceedings of the 6th International Semantic Web and 2nd Asian Semantic Web Conference (ISWC'07/ASWC'07), Busan, Korea. Springer-Verlag, 2007, pp. 722–735.

[9] Michele Banko, Michael J. Cafarella, Stephen Soderland, Matthew Broadhead, Oren Etzioni. **Open Information Extraction from the Web**. Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI 2007), Hyderabad, India, January 6-12, 2007. Ed. by Manuela M. Veloso. 2007, pp. 2670–2676.

[10] Michele Banko, Oren Etzioni. **Strategies for lifelong knowledge extraction from the web**. Proceedings of the 4th International Conference on Knowledge Capture (K-CAP 2007), October 28-31, 2007, Whistler, BC, Canada. Ed. by Derek H. Sleeman, Ken Barker. ACM, 2007, pp. 95–102.

[11] Michele Banko, Oren Etzioni. **The Tradeoffs Between Open and Traditional Relation Extraction**. Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics (ACL 2008), June 15-20, 2008, Columbus, Ohio, USA. Ed. by Kathleen McKeown, Johanna D. Moore, Simone Teufel, James Allan, Sadaoki Furui. ACL, 2008, pp. 28–36.

[12] Denilson Barbosa, Haixun Wang, Cong Yu. **Shallow Information Extraction for the knowledge Web**. 29th IEEE International Conference on Data Engineering (ICDE 2013), Brisbane, Australia, April 8-12, 2013. Ed. by Christian S. Jensen, Christopher M. Jermaine, Xiaofang Zhou. IEEE Computer Society, 2013, pp. 1264–1267.

[13] Hannah Bast, Florian Bäurle, Björn Buchhold, Elmar Haußmann. **Semantic full-text search with broccoli**. The 37th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2014), Gold Coast , QLD, Australia - July 06 - 11, 2014. Ed. by Shlomo Geva, Andrew Trotman, Peter Bruza, Charles L. A. Clarke, Kalervo Järvelin. ACM, 2014, pp. 1265–1266.

[14] Jonathan Berant, Andrew Chou, Roy Frostig, Percy Liang. **Semantic Parsing on Freebase from Question-Answer Pairs**. Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP 2013), 18-21 October 2013, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL. ACL, 2013, pp. 1533–1544.

[15] Tim Berners-Lee et al. **Tabulator: Exploring and Analyzing linked data on the Semantic Web**. Procedings of the The 3rd International Semantic Web User Interaction Workshop (SWUI 2006), November 6, 2006, Athens, Georgia, USA. 2006.

[16] Tim Berners-Lee et al. **Tabulator Redux: Browsing and Writing Linked Data**. Proceedings of the WWW 2008 Workshop on Linked Data on the Web (LDOW 2008), Beijing, China, April 22, 2008. Ed. by Christian Bizer, Tom Heath, Kingsley Idehen, Tim Berners-Lee. Vol. 369. CEUR Workshop Proceedings. CEUR-WS.org, 2008.

[17] Gaurav Bhalotia, Arvind Hulgeri, Charuta Nakhe, Soumen Chakrabarti, S. Sudarshan. **Keyword Searching and Browsing in Databases using BANKS**. Proceedings of the 18th International Conference on Data Engineering (ICDE 2002), San Jose, CA, USA, February 26 - March 1, 2002. Ed. by Rakesh Agrawal, Klaus R. Dittrich. IEEE Computer Society, 2002, pp. 431–440.

[18] Olivier Bodenreider. **The Unified Medical Language System (UMLS): integrating biomedical terminology**. *Nucleic Acids Research* 32.Database-Issue (2004), pp. 267–270.

[19] Kurt D. Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, Jamie Taylor. **Freebase: a collaboratively created graph database for structuring human knowledge**. Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD 2008), Vancouver, BC, Canada, June 10-12, 2008. Ed. by Jason Tsong-Li Wang. ACM, 2008, pp. 1247–1250.

[20] Sergey Brin. **Extracting Patterns and Relations from the World Wide Web**. The World Wide Web and Databases, International Workshop (WebDB'98), Valencia, Spain, March 27-28, 1998, Selected Papers. Ed. by Paolo Atzeni, Alberto O. Mendelzon, Giansalvatore Mecca. Vol. 1590. Lecture Notes in Computer Science. Springer, 1998, pp. 172–183.

[21] Gerald G. Brown, Robert F. Dell. **Formulating integer linear programs: A rogues' gallery**. *INFORMS Transactions on Education* 7.2 (2007), pp. 153–159.

[22] Razvan C. Bunescu, Marius Pasca. **Using Encyclopedic Knowledge for Named entity Disambiguation**. Proceedings of the 11st Conference of the European Chapter of the Association for Computational Linguistics (EACL 2006), April 3-7, 2006, Trento, Italy. Ed. by Diana McCarthy, Shuly Wintner. ACL, 2006.

[23] Michael J. Cafarella. **Extracting and Querying a Comprehensive Web Database**. 4th Biennial Conference on Innovative Data Systems Research (CIDR 2009), Asilomar, CA, USA, January 4-7, 2009, Online Proceedings. www.cidrdb.org, 2009.

[24] Michael J. Cafarella, Michele Banko, Oren Etzioni. **Relational web search**. Technical Report 06-04-02. University of Washington, 2006.

[25] Michael J. Cafarella, Alon Y. Halevy, Daisy Zhe Wang, Eugene Wu, Yang Zhang. **WebTables: exploring the power of tables on the web**. *Proceedings of the Very Large Data Bases Endowment (PVLDB),* 1.1 (2008), pp. 538–549.

[26] Jaime G. Carbonell, Jade Goldstein. **The Use of MMR, Diversity-Based Reranking for Reordering Documents and Producing Summaries**. Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '98), August 24-28, 1998, Melbourne, Australia. ACM, 1998, pp. 335–336.

[27] Andrew Carlson, Justin Betteridge, Richard C. Wang, Estevam R. Hruschka Jr., Tom M. Mitchell. **Coupled semi-supervised learning for information extraction**. Proceedings of the 3rd International Conference on Web Search and Web Data Mining (WSDM 2010), New York, NY, USA, February 4-6, 2010. Ed. by Brian D. Davison, Torsten Suel, Nick Craswell, Bing Liu. ACM, 2010, pp. 101–110.

[28] Andrew Carlson et al. **Toward an Architecture for Never-Ending Language Learning**. Proceedings of the 24th AAAI Conference on Artificial Intelligence (AAAI 2010), Atlanta, Georgia, USA, July 11-15, 2010. Ed. by Maria Fox, David Poole. AAAI Press, 2010.

[29] Angel X. Chang, Christopher D. Manning. **TokensRegex: Defining cascaded regular expressions over tokens**. Tech. rep. CSTR 2014-02. Department of Computer Science, Stanford University, 2014.

[30] Gong Cheng, Weiyi Ge, Yuzhong Qu. **Falcons: searching and browsing entities on the semantic web**. Proceedings of the 17th International Conference on World Wide Web (WWW 2008), Beijing, China, April 21-25, 2008. Ed. by Jinpeng Huai et al. ACM, 2008, pp. 1101–1102.

[31] Gong Cheng, Weiyi Ge, Yuzhong Qu. **Generating summaries for ontology search**. Proceedings of the 20th International Conference on World Wide Web (WWW 2011), Hyderabad, India, March 28 - April 1, 2011 (Companion Volume). Ed. by Sadagopan Srinivasan et al. ACM, 2011, pp. 27–28.

[32] Xiao Cheng, Dan Roth. **Relational Inference for Wikification**. Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP 2013), 18-21 October 2013, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL. ACL, 2013, pp. 1787–1796.

[33] Harr Chen, David R. Karger. **Less is more: probabilistic models for retrieving fewer relevant documents**. Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2006), Seattle, Washington, USA, August 6-11, 2006. Ed. by Efthimis N. Efthimiadis, Susan T. Dumais, David Hawking, Kalervo Järvelin. ACM, 2006, pp. 429–436.

[34] Laura Chiticariu et al. **SystemT: An Algebraic Approach to Declarative Information Extraction**. Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL 2010), July 11-16, 2010, Uppsala, Sweden. Ed. by Jan Hajic, Sandra Carberry, Stephen Clark. ACL, 2010, pp. 128–137.

[35] Charles L. A. Clarke et al. **Novelty and diversity in information retrieval evaluation**. Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2008), Singapore, July 20-24, 2008. Ed. by Sung-Hyon Myaeng, Douglas W. Oard, Fabrizio Sebastiani, Tat-Seng Chua, Mun-Kew Leong. ACM, 2008, pp. 659–666.

[36] Silviu Cucerzan. **Large-Scale Named Entity Disambiguation Based on Wikipedia Data**. Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL 2007), June 28-30, 2007, Prague, Czech Republic. Ed. by Jason Eisner. ACL, 2007, pp. 708–716.

[37] Bhavana Bharat Dalvi, Einat Minkov, Partha Pratim Talukdar, William W. Cohen. **Automatic Gloss Finding for a Knowledge Base using Ontological Constraints**. Proceedings of the 8th ACM International Conference on Web Search and Data Mining (WSDM 2015), Shanghai, China, February 2-6, 2015. Ed. by Xueqi Cheng, Hang Li, Evgeniy Gabrilovich, Jie Tang. ACM, 2015, pp. 369–378.

[38] Daniel Diaz. **GNU Prolog**. `www.gprolog.org`. 2013.

[39] Xin Luna Dong, Barna Saha, Divesh Srivastava. **Less is More: Selecting Sources Wisely for Integration**. *Proceedings of the Very Large Data Bases Endowment (PVLDB),* 6.2 (2012), pp. 37–48.

[40] Shady Elbassuoni. **Effective Searching of RDF Knowledge Bases**. PhD thesis. Max-Planck-Institut für Informatik, Germany, 2012.

[41] Shady Elbassuoni, Katja Hose, Steffen Metzger, Ralf Schenkel. **ROXXI: Reviving witness dOcuments to eXplore eXtracted Information**. *Proceedings of the Very Large Data Bases Endowment (PVLDB),* 3.2 (2010), pp. 1589–1592.

[42] Shady Elbassuoni, Maya Ramanath, Ralf Schenkel, Marcin Sydow, Gerhard Weikum. **Language-model-based ranking for queries on RDF-graphs**. Proceedings of the 18th ACM Conference on Information and Knowledge Management (CIKM 2009) Hong Kong, China, November 2-6, 2009. Ed. by David Wai-Lok Cheung, Il-Yeol Song, Wesley W. Chu, Xiaohua Hu, Jimmy J. Lin. ACM, 2009, pp. 977–986.

[43] **Entity Cube**. `entitycube.research.microsoft.com`.

[44] Anthony Fader, Stephen Soderland, Oren Etzioni. **Identifying Relations for Open Information Extraction**. Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP 2011), 27-31 July 2011, John McIntyre Conference Centre, Edinburgh, UK, A meeting of SIGDAT, a Special Interest Group of the ACL. ACL, 2011, pp. 1535–1545.

[45] Anthony Fader, Luke Zettlemoyer, Oren Etzioni. **Open question answering over curated and extracted knowledge bases**. The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2014), New York, NY, USA - August 24 - 27, 2014. Ed. by Sofus A. Macskassy, Claudia Perlich, Jure Leskovec, Wei Wang, Rayid Ghani. ACM, 2014, pp. 1156–1165.

[46] Anthony Fader, Luke S. Zettlemoyer, Oren Etzioni. **Paraphrase-Driven Learning for Open Question Answering**. Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL 2013), 4-9 August 2013, Sofia, Bulgaria, Volume 1: Long Papers. ACL, 2013, pp. 1608–1618.

[47] Lujun Fang, Anish Das Sarma, Cong Yu, Philip Bohannon. **REX: Explaining Relationships between Entity Pairs**. *Proceedings of the Very Large Data Bases Endowment (PVLDB),* 5.3 (2011), pp. 241–252.

[48] Christiane Fellbaum, ed. **WordNet: An Electronic Lexical Database**. Language, speech, and communication. MIT Press, 1998.

[49] Paolo Ferragina, Ugo Scaiella. **TAGME: on-the-fly annotation of short text fragments (by wikipedia entities)**. Proceedings of the 19th ACM Conference on Information and Knowledge Management (CIKM 2010), Toronto, Ontario, Canada, October 26-30, 2010. Ed. by Jimmy Huang et al. ACM, 2010, pp. 1625–1628.

[50] David A. Ferrucci et al. **Building Watson: An Overview of the DeepQA Project**. *AI Magazine* 31.3 (2010), pp. 59–79.

[51] Jenny Rose Finkel, Trond Grenager, Christopher D. Manning. **Incorporating Non-local Information into Information Extraction Systems by Gibbs Sampling**. Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL 2005), 25-30 June 2005, University of Michigan, USA. Ed. by Kevin Knight, Hwee Tou Ng, Kemal Oflazer. ACL, 2005.

[52] Tim Furche, Georg Gottlob, Giovanni Grasso, Christian Schallhart, Andrew Jon Sellers. **OXPath: A language for scalable data extraction, automation, and crawling on the deep web**. *Proceedings of the Very Large Data Bases Endowment (PVLDB),* 22.1 (2013), pp. 47–72.

[53] Luis Galarraga, Geremy Heitz, Kevin Murphy, Fabian M. Suchanek. **Canonicalizing Open Knowledge Bases**. Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management (CIKM 2014), Shanghai, China, November 3-7, 2014. Ed. by Jianzhong Li et al. ACM, 2014, pp. 1679–1688.

[54] William A. Gale, Kenneth W. Church, David Yarowsky. **One Sense Per Discourse**. Proceedings of the Workshop on Speech and Natural Language (HLT '91). ACL, 1992, pp. 233–237.

[55] Alban Galland, Serge Abiteboul, Amélie Marian, Pierre Senellart. **Corroborating information from disagreeing views**. Proceedings of the 3rd International Conference on Web Search and Web Data Mining (WSDM 2010), New York, NY, USA, February 4-6, 2010. Ed. by Brian D. Davison, Torsten Suel, Nick Craswell, Bing Liu. ACM, 2010, pp. 131–140.

[56] Michael R. Garey, David S. Johnson. **Computers and Intractability: A Guide to the Theory of NP-Completeness**. New York, NY, USA: W. H. Freeman & Co., 1979.

[57] Martin C. Golumbic. **Algorithmic Graph Theory and Perfect Graphs**. Annals of Discrete Mathematics. Amsterdam, Netherlands: North-Holland Publishing Co., 2004.

[58] Rahul Gupta, Alon Y. Halevy, Xuezhi Wang, Steven Euijong Whang, Fei Wu. **Biperpedia: An Ontology for Search Applications**. *Proceedings of the Very Large Data Bases Endowment (PVLDB),* 7.7 (2014), pp. 505–516.

[59] Gurobi Optimization, Inc. **Gurobi Optimizer Reference Manual**. `www.gurobi.com`. 2014.

[60] Takaaki Hasegawa, Satoshi Sekine, Ralph Grishman. **Discovering Relations among Named Entities from Large Corpora**. Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics, 21-26 July, 2004, Barcelona, Spain. Ed. by Donia Scott, Walter Daelemans, Marilyn A. Walker. ACL, 2004, pp. 415–422.

[61] Taher H. Haveliwala. **Topic-Sensitive PageRank: A Context-Sensitive Ranking Algorithm for Web Search**. *IEEE Transactions on Knowledge and Data Engineering* 15.4 (2003), pp. 784–796.

[62] Johannes Hoffart, Fabian M. Suchanek, Klaus Berberich, Gerhard Weikum. **YAGO2: A spatially and temporally enhanced knowledge base from Wikipedia**. *Artificial Intelligence* 194 (2013), pp. 28–61.

[63] Johannes Hoffart et al. **Robust Disambiguation of Named Entities in Text**. Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP 2011), 27-31 July 2011, Edinburgh, UK, A meeting of SIGDAT, a Special Interest Group of the ACL. ACL, 2011, pp. 782–792.

[64] Johannes Hoffart et al. **YAGO2: exploring and querying world knowledge in time, space, context, and many languages**. Proceedings of the 20th International Conference on World Wide Web (WWW 2011), Hyderabad, India, March 28 - April 1, 2011 (Companion Volume). Ed. by Sadagopan Srinivasan et al. ACM, 2011, pp. 229–232.

[65] Thomas Huet, Joanna Biega, Fabian M. Suchanek. **Mining History with Le Monde**. Proceedings of the 2013 workshop on Automated Knowledge Base Construction (AKBC 2013) (2013), pp. 49–54.

[66] Mohsen Jamali, Martin Ester. ***TrustWalker*: a random walk model for combining trust-based and item-based recommendation**. Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Paris, France, June 28 - July 1, 2009. Ed. by John F. Elder IV, Françoise Fogelman-Soulié, Peter A. Flach, Mohammed Javeed Zaki. ACM, 2009, pp. 397–406.

[67] Gjergji Kasneci, Shady Elbassuoni, Gerhard Weikum. **MING: mining informative entity relationship subgraphs**. Proceedings of the 18th ACM Conference on Information and Knowledge Management (CIKM 2009), Hong Kong, China, November 2-6, 2009. Ed. by David Wai-Lok Cheung, Il-Yeol Song, Wesley W. Chu, Xiaohua Hu, Jimmy J. Lin. ACM, 2009, pp. 1653–1656.

[68] Gjergji Kasneci, Maya Ramanath, Mauro Sozio, Fabian M. Suchanek, Gerhard Weikum. **STAR: Steiner-Tree Approximation in Relationship Graphs**. Proceedings of the 25th International Conference on Data Engineering (ICDE 2009), March 29 2009 - April 2 2009, Shanghai, China. Ed. by Yannis E. Ioannidis, Dik Lun Lee, Raymond T. Ng. IEEE, 2009, pp. 868–879.

[69] Gjergji Kasneci, Fabian M. Suchanek, Georgiana Ifrim, Maya Ramanath, Gerhard Weikum. **NAGA: Searching and Ranking Knowledge**. Proceedings of the 24th International Conference on Data Engineering (ICDE 2008), April 7-12, 2008, Cancún, México. Ed. by Gustavo Alonso, José A. Blakeley, Arbee L. P. Chen. IEEE, 2008, pp. 953–962.

[70] Philip Kendall. **Re: Retrieve all the subtypes of a specified type**. `markmail.org/message/daul65gtobgawf6q`. Message in Freebase-discuss mailing list. 2013.

[71] Mitchell Koch, John Gilmer, Stephen Soderland, Daniel S. Weld. **Type-Aware Distantly Supervised Relation Extraction with Linked Arguments**. Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014), October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL. Ed. by Alessandro Moschitti, Bo Pang, Walter Daelemans. ACL, 2014, pp. 1891–1901.

[72] Stanley Kok, Pedro Domingos. **Extracting Semantic Networks from Text Via Relational Clustering**. Machine Learning and Knowledge Discovery in Databases, European Conference (ECML/PKDD 2008), Antwerp, Belgium, September 15-19, 2008, Proceedings, Part I. Ed. by Walter Daelemans, Bart Goethals, Katharina Morik. Vol. 5211. Lecture Notes in Computer Science. Springer, 2008, pp. 624–639.

[73] Krzysztof Kuchcinski, Radosław Szymanek. **JaCoP Library User's Guide**. `jacop.osolpro.com`.

[74] Sayali Kulkarni, Amit Singh, Ganesh Ramakrishnan, Soumen Chakrabarti. **Collective annotation of Wikipedia entities in web text**. Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Paris, France, June 28 - July 1, 2009. Ed. by John F. Elder IV, Françoise Fogelman-Soulié, Peter A. Flach, Mohammed Javeed Zaki. ACM, 2009, pp. 457–466.

[75] Kow Kuroda, Masaki Murata, Kentaro Torisawa. **When nouns need co-arguments: A case study of semantically unsaturated nouns.** Proceedings of the 5th International Conference on Generative Approaches to the Lexicon. Pisa, Italy, Sept. 2009, pp. 193–200.

[76] Dustin Lange, Christoph Böhm, Felix Naumann. **Extracting structured information from Wikipedia articles to populate infoboxes**. Proceedings of the 19th ACM Conference on Information and Knowledge Management (CIKM 2010), Toronto, Ontario, Canada, October 26-30, 2010. Ed. by Jimmy Huang et al. ACM, 2010, pp. 1661–1664.

[77] Heeyoung Lee et al. **Stanford's Multi-Pass Sieve Coreference Resolution System at the CoNLL-2011 Shared Task**. Proceedings of the 15th Conference on Computational Natural Language Learning: Shared Task, CoNLL 2011, Portland, Oregon, USA, June 23-24, 2011. Ed. by Sameer Pradhan et al. ACL, 2011, pp. 28–34.

[78] Douglas B. Lenat. **CYC: A Large-Scale Investment in Knowledge Infrastructure**. *Communications of the ACM* 38.11 (1995), pp. 32–38.

[79] Chengkai Li, Ning Yan, Senjuti Basu Roy, Lekhendro Lisham, Gautam Das. **Facetedpedia: dynamic generation of query-dependent faceted interfaces for wikipedia**. Proceedings of the 19th International Conference on World Wide Web (WWW 2010), Raleigh, North Carolina, USA, April 26-30, 2010. Ed. by Michael Rappa, Paul Jones, Juliana Freire, Soumen Chakrabarti. ACM, 2010, pp. 651–660.

[80] Jimmy Lin, Chris Dyer. **Data-Intensive Text Processing with MapReduce**. Morgan and Claypool Publishers, 2010.

[81] Peng Li, Jing Jiang, Yinglin Wang. **Generating Templates of Entity Summaries with an Entity-Aspect Model and Pattern Mining**. Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, ACL 2010, July 11-16, 2010, Uppsala, Sweden. Ed. by Jan Hajic, Sandra Carberry, Stephen Clark. ACL, 2010, pp. 640–649.

[82] Qi Li et al. **Resolving conflicts in heterogeneous data by truth discovery and source reliability estimation**. International Conference on Management of Data (SIGMOD 2014), Snowbird, UT, USA, June 22-27, 2014. Ed. by Curtis E. Dyreson, Feifei Li, M. Tamer Özsu. ACM, 2014, pp. 1187–1198.

[83] Xian Li, Xin Luna Dong, Kenneth Lyons, Weiyi Meng, Divesh Srivastava. **Truth Finding on the Deep Web: Is the Problem Solved?** *Proceedings of the Very Large Data Bases Endowment (PVLDB),* 6.2 (2012), pp. 97–108.

[84] Xian Li, Weiyi Meng, Clement T. Yu. **T-verifier: Verifying truthfulness of fact statements**. Proceedings of the 27th International Conference on Data Engineering (ICDE 2011), April 11-16, 2011, Hannover, Germany. Ed. by Serge Abiteboul, Klemens Böhm, Christoph Koch, Kian-Lee Tan. IEEE Computer Society, 2011, pp. 63–74.

[85] Jayant Madhavan et al. **Structured Data Meets the Web: A Few Observations**. *IEEE Data Engineering Bulletin* 29.4 (2006), pp. 19–26.

[86] Farzaneh Mahdisoltani, Joanna Biega, Fabian Suchanek. **YAGO3: A Knowledge Base from Multilingual Wikipedias**. 7th Biennial Conference on Innovative Data Systems Research. CIDR Conference, 2014.

[87] Christopher D. Manning, Prabhakar Raghavan, Hinrich Schütze. **Introduction to Information Retrieval**. Cambridge University Press, 2008.

[88] Christopher D. Manning et al. **The Stanford CoreNLP Natural Language Processing Toolkit**. Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL 2014), June 22-27, 2014, Baltimore, MD, USA, System Demonstrations. ACL, 2014, pp. 55–60.

[89] Cynthia Matuszek et al. **Searching for Common Sense: Populating Cyc™ from the Web**. Proceedings of the 20th National Conference on Artificial Intelligence and the 17th Innovative Applications of Artificial Intelligence Conference, July 9-13, 2005, Pittsburgh, Pennsylvania, USA. Ed. by Manuela M. Veloso, Subbarao Kambhampati. AAAI Press / The MIT Press, 2005, pp. 1430–1435.

[90] Mausam, Michael Schmitz, Stephen Soderland, Robert Bart, Oren Etzioni. **Open Language Learning for Information Extraction**. Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL 2012), July 12-14, 2012, Jeju Island, Korea. ACL, 2012, pp. 523–534.

[91] Arturas Mazeika, Tomasz Tylenda, Gerhard Weikum. **Entity timelines: visual analytics and named entity evolution**. Proceedings of the 20th ACM Conference on Information and Knowledge Management (CIKM 2011), Glasgow, United Kingdom, October 24-28, 2011. Ed. by Craig Macdonald, Iadh Ounis, Ian Ruthven. ACM, 2011, pp. 2585–2588.

[92] Gerard de Melo, Gerhard Weikum. **MENTA: inducing multilingual taxonomies from wikipedia**. Proceedings of the 19th ACM Conference on Information and Knowledge Management (CIKM 2010), Toronto, Ontario, Canada, October 26-30, 2010. Ed. by Jimmy Huang et al. ACM, 2010, pp. 1099–1108.

[93] Pablo N. Mendes, Max Jakob, Andrés García-Silva, Christian Bizer. **DBpedia spotlight: shedding light on the web of documents**. Proceedings the 7th International Conference on Semantic Systems (I-SEMANTICS 2011), Graz, Austria, September 7-9, 2011. Ed. by Chiara Ghidini, Axel-Cyrille Ngonga Ngomo, Stefanie N. Lindstaedt, Tassilo Pellegrini. ACM International Conference Proceeding Series. ACM, 2011, pp. 1–8.

[94] Steffen Metzger, Shady Elbassuoni, Katja Hose, Ralf Schenkel. **S3K: seeking statement-supporting top-K witnesses**. Proceedings of the 20th ACM Conference on Information and Knowledge Management (CIKM 2011), Glasgow, United Kingdom, October 24-28, 2011. Ed. by Craig Macdonald, Iadh Ounis, Ian Ruthven. ACM, 2011, pp. 37–46.

[95] Jean-Baptiste Michel et al. **Quantitative analysis of culture using millions of digitized books**. *Science* 331.6014 (2011), pp. 176–182.

[96] Rada Mihalcea, Andras Csomai. **Wikify!: linking documents to encyclopedic knowledge**. Proceedings of the 16th ACM Conference on Information and Knowledge Management (CIKM 2007), Lisbon, Portugal, November 6-10, 2007. Ed. by Mário J. Silva et al. ACM, 2007, pp. 233–242.

[97] George A. Miller. **WordNet: A Lexical Database for English**. *Communications of the ACM* 38.11 (1995), pp. 39–41.

[98] David N. Milne, Ian H. Witten. **Learning to link with wikipedia**. Proceedings of the 17th ACM Conference on Information and Knowledge Management (CIKM 2008), Napa Valley, California, USA, October 26-30, 2008. Ed. by James G. Shanahan et al. ACM, 2008, pp. 509–518.

[99] Mike Mintz, Steven Bills, Rion Snow, Daniel Jurafsky. **Distant supervision for relation extraction without labeled data**. Proceedings of the 47th Annual Meeting of the Association for Computational Linguistics (ACL 2009) and the 4th International Joint Conference on Natural Language Processing of the AFNLP, 2-7 August 2009, Singapore. Ed. by Keh-Yih Su, Jian Su, Janyce Wiebe. ACL, 2009, pp. 1003–1011.

[100] T. Mitchell et al. **Never-Ending Learning**. Proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI-15), January 25 -30, 2015, Austin, Texas USA. AAAI Press, 2015.

[101] Raymond J. Mooney, Razvan C. Bunescu. **Mining knowledge from text using information extraction**. *SIGKDD Explorations Newsletter* 7.1 (2005), pp. 3–10.

[102] Andrea Moro, Roberto Navigli. **WiSeNet: building a wikipedia-based semantic network with ontologized relations**. 21st ACM International Conference on Information and Knowledge Management, (CIKM 2012), Maui, HI, USA, October 29 - November 02, 2012. Ed. by Xue-wen Chen, Guy Lebanon, Haixun Wang, Mohammed J. Zaki. ACM, 2012, pp. 1672–1676.

[103] Ndapandula Nakashole, Tom M. Mitchell. **Language-Aware Truth Assessment of Fact Candidates**. Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL 2014), June 22-27, 2014, Baltimore, MD, USA, Volume 1: Long Papers. ACL, 2014, pp. 1009–1019.

[104] Ndapandula Nakashole, Gerhard Weikum, Fabian M. Suchanek. **PATTY: A Taxonomy of Relational Patterns with Semantic Types**. Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL 2012), July 12-14, 2012, Jeju Island, Korea. ACL, 2012, pp. 1135–1145.

[105] Roberto Navigli, Simone Paolo Ponzetto. **BabelNet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network**. *Artificial Intelligence* 193 (2012), pp. 217–250.

[106] Jeff Pasternack, Dan Roth. **Latent credibility analysis**. 22nd International World Wide Web Conference (WWW 2013), Rio de Janeiro, Brazil, May 13-17, 2013. Ed. by Daniel Schwabe, Virgílio A. F. Almeida, Hartmut Glaser, Ricardo A. Baeza-Yates, Sue B. Moon. International World Wide Web Conferences Steering Committee / ACM, 2013, pp. 1009–1020.

[107] Jeff Pasternack, Dan Roth. **Making Better Informed Trust Decisions with Generalized Fact-Finding**. Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI 2011), Barcelona, Catalonia, Spain, July 16-22, 2011. Ed. by Toby Walsh. IJCAI/AAAI, 2011, pp. 2324–2329.

[108] Thomas Penin, Haofen Wang, Thanh Tran, Yong Yu. **Snippet Generation for Semantic Web Search Engines**. The Semantic Web, 3rd Asian Semantic Web Conference (ASWC 2008), Bangkok, Thailand, December 8-11, 2008. Proceedings. Ed. by John Domingue, Chutiporn Anutariya. Vol. 5367. Lecture Notes in Computer Science. Springer, 2008, pp. 493–507.

[109] Simone Paolo Ponzetto, Michael Strube. **Deriving a Large-Scale Taxonomy from Wikipedia**. Proceedings of the 22nd AAAI Conference on Artificial Intelligence, July 22-26, 2007, Vancouver, British Columbia, Canada. AAAI Press, 2007, pp. 1440–1445.

[110] Martin F. Porter. **An algorithm for suffix stripping**. *Program* 14.3 (1980), pp. 130–137.

[111] Lev-Arie Ratinov, Dan Roth, Doug Downey, Mike Anderson. **Local and Global Algorithms for Disambiguation to Wikipedia**. Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, 19-24 June, 2011, Portland, Oregon, USA. Ed. by Dekang Lin, Yuji Matsumoto, Rada Mihalcea. ACL, 2011, pp. 1375–1384.

[112] Álvaro Rodrigo, Anselmo Peñas, Felisa Verdejo. **Overview of the Answer Validation Exercise 2008**. Evaluating Systems for Multilingual and Multimodal Information Access, 9th Workshop of the Cross-Language Evaluation Forum (CLEF 2008), Aarhus, Denmark, September 17-19, 2008, Revised Selected Papers. Ed. by Carol Peters et al. Vol. 5706. Lecture Notes in Computer Science. Springer, 2008, pp. 296–313.

[113] Benjamin Roth, Tassilo Barth, Michael Wiegand, Dietrich Klakow. **A survey of noise reduction methods for distant supervision**. Proceedings of the 2013 workshop on Automated Knowledge Base Construction (AKBC 2013). San Francisco, California, USA: ACM, 2013, pp. 73–78.

[114] Sunita Sarawagi. **Information Extraction**. *Foundations and Trends in Databases* 1.3 (2008), pp. 261–377.

[115] Wei Shen, Jianyong Wang, Jiawei Han. **Entity Linking with a Knowledge Base: Issues, Techniques, and Solutions**. *IEEE Transactions on Knowledge and Data Engineering* 27.2 (2015), pp. 443–460.

[116] Amit Singhal. **Introducing the Knowledge Graph: things, not strings**. Official Google Blog. `googleblog.blogspot.de/2012/05/introducing-knowledge-graph-things-not.html`. May 16th, 2012.

[117] Grzegorz Sobczak, Mariusz Pikula, Marcin Sydow. **AGNES: A Novel Algorithm for Visualising Diversified Graphical Entity Summarisations on Knowledge Graphs**. Proceedings of the 20th International Conference on Foundations of Intelligent Systems (ISMIS 2012), Macau, China, December 4-7, 2012. Ed. by Li Chen, Alexander Felfernig, Jiming Liu, Zbigniew W. Ras. Vol. 7661. Lecture Notes in Computer Science. Springer, 2012, pp. 182–191.

[118] Fabian M. Suchanek, Gjergji Kasneci, Gerhard Weikum. **Yago: a core of semantic knowledge**. Proceedings of the 16th International Conference on World Wide Web (WWW 2007), Banff, Alberta, Canada, May 8-12, 2007. Ed. by Carey L. Williamson, Mary Ellen Zurko, Peter F. Patel-Schneider, Prashant J. Shenoy. ACM, 2007, pp. 697–706.

[119] Fabian M. Suchanek, Gjergji Kasneci, Gerhard Weikum. **YAGO: A Large Ontology from Wikipedia and WordNet**. *Journal of Web Semantics* 6.3 (2008), pp. 203–217.

[120] Fabian M. Suchanek, Mauro Sozio, Gerhard Weikum. **SOFIE: a self-organizing framework for information extraction**. Proceedings of the 18th International Conference on World Wide Web (WWW 2009), Madrid, Spain, April 20-24, 2009. Ed. by Juan Quemada, Gonzalo León, Yoëlle S. Maarek, Wolfgang Nejdl. ACM, 2009, pp. 631–640.

[121] Fabian M. Suchanek, Gerhard Weikum. **Knowledge harvesting from text and Web sources**. 29th IEEE International Conference on Data Engineering (ICDE 2013), Brisbane, Australia, April 8-12, 2013. Ed. by Christian S. Jensen, Christopher M. Jermaine, Xiaofang Zhou. IEEE Computer Society, 2013, pp. 1250–1253.

[122] Marcin Sydow, Mariusz Pikula, Ralf Schenkel. **The notion of diversity in graphical entity summarisation on semantic knowledge graphs**. *Journal of Intelligent Information Systems* 41.2 (2013), pp. 109–149.

[123] Guido Tack. **Constraint Propagation - Models, Techniques, Implementation**. PhD thesis. Saarland University, Germany, 2009.

[124] Partha Pratim Talukdar, Derry Tanti Wijaya, Tom M. Mitchell. **Acquiring temporal constraints between relations**. Proceedings of the 21st ACM International Conference on Information and Knowledge Management (CIKM 2012), Maui, HI, USA, October 29 - November 02, 2012. Ed. by Xue-wen Chen, Guy Lebanon, Haixun Wang, Mohammed J. Zaki. ACM, 2012, pp. 992–1001.

[125] Chun How Tan, Eugene Agichtein, Panos Ipeirotis, Evgeniy Gabrilovich. **Trust, but verify: predicting contribution quality for knowledge base construction and curation**. Proceedings of the 7th ACM International Conference on Web Search and Data Mining (WSDM 2014), New York, NY, USA, February 24-28, 2014. Ed. by Ben Carterette, Fernando Diaz, Carlos Castillo, Donald Metzler. ACM, 2014, pp. 553–562.

[126] Hanghang Tong, Christos Faloutsos. **Center-piece subgraphs: problem definition and fast solutions**. Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Philadelphia, PA, USA, August 20-23, 2006. Ed. by Tina Eliassi-Rad, Lyle H. Ungar, Mark Craven, Dimitrios Gunopulos. ACM, 2006, pp. 404–413.

[127] Tran Anh Tuan, Shady Elbassuoni, Nicoleta Preda, Gerhard Weikum. **CATE: context-aware timeline for entity illustration**. Proceedings of the 20th International Conference on World Wide Web (WWW 2011) Hyderabad, India, March 28 - April 1, 2011 (Companion Volume). Ed. by Sadagopan Srinivasan et al. ACM, 2011, pp. 269–272.

[128] Giovanni Tummarello et al. **Sig.ma: Live views on the Web of Data**. *Journal of Web Semantics* 8.4 (2010), pp. 355–364.

[129] Tomasz Tylenda, Sarath Kumar Kondreddi, Gerhard Weikum. **Spotting Knowledge Base Facts in Web Texts**. Proceedings of the 2014 workshop on Automated Knowledge Base Construction (AKBC 2014). Montreal, Canada, 2014.

[130] Tomasz Tylenda, Mauro Sozio, Gerhard Weikum. **Einstein: physicist or vegetarian? summarizing semantic type graphs for knowledge discovery**. Proceedings of the 20th International Conference on World Wide Web (WWW 2011), Hyderabad, India, March 28 - April 1, 2011 (Companion Volume). Ed. by Sadagopan Srinivasan et al. ACM, 2011, pp. 273–276.

[131] Tomasz Tylenda, Yafang Wang, Gerhard Weikum. **Spotting Facts in the Wild**. Workshop on Automatic Creation and Curation of Knowledge Bases (WACCK 2014). Snowbird, Utah, USA, 2014.

[132] **Wolfram Alpha**. `www.wolframalpha.com`.

[133] Fei Wu, Raphael Hoffmann, Daniel S. Weld. **Information extraction from Wikipedia: moving down the long tail**. Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Las Vegas, Nevada, USA, August 24-27, 2008. Ed. by Ying Li, Bing Liu, Sunita Sarawagi. ACM, 2008, pp. 731–739.

[134] Fei Wu, Daniel S. Weld. **Autonomously semantifying wikipedia**. Proceedings of the 16th ACM Conference on Information and Knowledge Management (CIKM 2007), Lisbon, Portugal, November 6-10, 2007. Ed. by Mário J. Silva et al. ACM, 2007, pp. 41–50.

[135] Minji Wu, Amélie Marian. **A framework for corroborating answers from multiple web sources**. *Information Systems* 36.2 (2011), pp. 431–449.

[136] Minji Wu, Amélie Marian. **Corroborating Facts from Affirmative Statements**. Proceedings of the 17th International Conference on Extending Database Technology (EDBT 2014), Athens, Greece, March 24-28, 2014. Ed. by Sihem Amer-Yahia et al. OpenProceedings.org, 2014, pp. 157–168.

[137] Wentao Wu, Hongsong Li, Haixun Wang, Kenny Qili Zhu. **Probase: a probabilistic taxonomy for text understanding**. Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD 2012), Scottsdale, AZ, USA, May 20-24, 2012. Ed. by K. Selçuk Candan, Yi Chen, Richard T. Snodgrass, Luis Gravano, Ariel Fuxman. ACM, 2012, pp. 481–492.

[138] Mohamed Yahya et al. **Natural Language Questions for the Web of Data**. Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL 2012), July 12-14, 2012, Jeju Island, Korea. ACL, 2012, pp. 379–390.

[139] Xiaoxin Yin, Jiawei Han, Philip S. Yu. **Truth Discovery with Multiple Conflicting Information Providers on the Web**. *IEEE Transactions on Knowledge and Data Engineering* 20.6 (2008), pp. 796–808.

[140] Xiaoxin Yin, Jiawei Han, Philip S. Yu. **Truth discovery with multiple conflicting information providers on the web**. Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Jose, California, USA, August 12-15, 2007. Ed. by Pavel Berkhin, Rich Caruana, Xindong Wu. ACM, 2007, pp. 1048–1052.

[141] Mohamed Amir Yosef, Johannes Hoffart, Ilaria Bordino, Marc Spaniol, Gerhard Weikum. **AIDA: An Online Tool for Accurate Disambiguation of Named Entities in Text and Tables**. *Proceedings of the Very Large Data Bases Endowment (PVLDB),* 4.12 (2011), pp. 1450–1453.

[142] Hugo Zaragoza et al. **Ranking very many typed entities on wikipedia**. Proceedings of the 16th ACM Conference on Information and Knowledge Management (CIKM 2007) Lisbon, Portugal, November 6-10, 2007. Ed. by Mário J. Silva et al. ACM, 2007, pp. 1015–1018.

[143] Xiang Zhang, Gong Cheng, Yuzhong Qu. **Ontology summarization based on rdf sentence graph**. Proceedings of the 16th International Conference on World Wide Web (WWW 2007) Banff, Alberta, Canada, May 8-12, 2007. Ed. by Carey L. Williamson, Mary Ellen Zurko, Peter F. Patel-Schneider, Prashant J. Shenoy. ACM, 2007, pp. 707–716.

[144] Bo Zhao, Benjamin I. P. Rubinstein, Jim Gemmell, Jiawei Han. **A Bayesian Approach to Discovering Truth from Conflicting Sources for Data Integration**. *Proceedings of the Very Large Data Bases Endowment (PVLDB),* 5.6 (2012), pp. 550–561.

[145] Jun Zhu, Zaiqing Nie, Xiaojiang Liu, Bo Zhang, Ji-Rong Wen. **StatSnowball: a statistical approach to extracting entity relationships**. Proceedings of the 18th International Conference on World Wide Web (WWW 2009), Madrid, Spain, April 20-24, 2009. Ed. by Juan Quemada, Gonzalo León, Yoëlle S. Maarek, Wolfgang Nejdl. ACM, 2009, pp. 101–110.

List of Figures

List of Tables

List of Algorithms