# Towards Precise and Convenient Semantic Search on Text and Knowledge Bases

Dissertation zur Erlangung des Doktorgrades
der Ingenieurwissenschaften (Dr.-Ing.)
der Technischen Fakultät
der Albert-Ludwigs-Universität Freiburg im Breisgau

vorgelegt von
**Elmar Haußmann**

Albert-Ludwigs-Universität Freiburg
Technische Fakultät
Institut für Informatik
2017

Datum der Disputation: 30. November 2017

| | |
|---|---|
| Dekan: | Prof. Dr. Oliver Paul |
| Erstgutachterin: | Prof. Dr. Hannah Bast (Universität Freiburg) |
| Zweitgutachter: | Prof. Dr. Gerhard Weikum (Max-Planck-Institut Saarbrücken) |
| Vorsitz: | Prof. Dr. Frank Hutter (Universität Freiburg) |
| Beisitz: | Dr. Joschka Bödecker (Universität Freiburg) |

## Abstract

In this dissertation, we consider the problem of making semantic search on text and knowledge bases more *precise* and *convenient*. In a nutshell, semantic search is *search with meaning*. To this respect, text and knowledge bases have different advantages and disadvantages. Large amounts of text are easily available on the web, and they contain a wealth of information in natural language. However, text represents information in an *unstructured* form. It follows no pre-defined schema, and without further processing, a machine can understand its meaning only on a superficial level. Knowledge bases, on the other hand, contain *structured* information in the form of *subject predicate object* triples. The meaning of triples is well defined, and triples can be retrieved precisely via a query language. However, formulating queries in this language is inconvenient and compared to text only a small fraction of information is currently available in knowledge bases.

In this document, we summarize our contributions on making semantic search on text and knowledge bases more precise and convenient. For knowledge bases, we introduce an approach to answer natural language questions. A user can pose questions conveniently in natural language and ask, for example, *who is the ceo of apple?*, instead of having to learn and use a specific query language. Our approach applies learning-to-rank strategies and improved the state of the art on two widely used benchmarks at the time of publication. For knowledge bases, we also describe a novel approach to compute relevance scores for triples from type-like relations like *profession* and *nationality*. For example, on a large knowledge base, a query for *american actors* can return a list of more than 60 thousand actors in no particular order. Relevance scores allow to sort this list so that, e.g., frequent lead actors appear before those who only had single cameo roles. In a benchmark that we generated via crowdsourcing, we show that our rankings are closer to human judgments than approaches from the literature. Finally, for text, we introduce a novel natural language processing technique that identifies which words in a sentence "semantically belong together". For example, in the sentence *Bill Gates, founder of Microsoft, and Jeff Bezos, founder of Amazon, are among the wealthiest persons in the world*, the words *Bill Gates*, *founder*, and *Amazon* do not belong together, but the words *Bill Gates*, *founder*, and *Microsoft* do. We show that when query keywords are required to belong together in order to match, search results become more precise.

Given the characteristics of text and knowledge bases outlined above, it is promising to consider a search that combines both. For example, for the query *CEOs of U.S. companies who advocate cryptocurrencies*, a list of CEOs of U.S. companies can be retrieved from a knowledge base. The information who is advocating cryptocurrencies is rather specific and changes frequently. It is, therefore, better found in full text. As part of this thesis, we describe how a combined search could be achieved and present and evaluate a fully functional prototype. All of our approaches are accompanied by an extensive evaluation which show their practicability and, where available, compare them to established approaches from the literature.

## Kurzzusammenfassung

Diese Dissertation beschäftigt sich mit der Aufgabenstellung, semantische Suche in Text und Wissensdatenbanken präziser und komfortabler zu machen. Semantische Suche ist kurz gesagt eine „Suche mittels Bedeutung". In diesem Kontext haben Text und Wissensdatenbanken unterschiedliche Vor- und Nachteile. So ist Text in großer Menge im World Wide Web verfügbar und enthält eine Fülle an Informationen. Für Nutzer von Suchmaschinen ist die Suche nach Informationen über Schlagwörter einfach handzuhaben und hat sich für viele Anfragen als effektiv herausgestellt. Allerdings sind Informationen in Text *unstrukturiert*. Sie folgen keinem vorgegebenen Schema und ohne weitere Verarbeitung ist die Bedeutung von Text für eine Maschine nur oberflächlich erkennbar. In der Konsequenz ist es oft schwierig, präzise nach Informationen in Text zu suchen. Dagegen enthalten Wissensdatenbanken *strukturierte* Information in Form von Tripeln aus *Subjekt Prädikat Objekt*. Die Bedeutung von Tripeln ist für eine Maschine klar definiert und Tripel können präzise über eine spezielle Abfragesprache gefunden werden. Allerdings ist das Formulieren einer Abfrage in dieser Sprache umständlich. Zudem sind im Vergleich zu Text weniger Informationen in Wissensdatenbanken verfügbar.

Angesichts dieser Eigenschaften ist eine kombinierte Suche in Text und Wissensdatenbanken vielversprechend. Zum Beispiel kann für die Anfrage *CEOs von US-Firmen, die Kryptowährungen unterstützen* eine Liste mit CEOs von US-Firmen aus einer Wissendatenbank bezogen werden. Die Information, welche CEOs Kryptowährungen unterstützen, ist relativ spezifisch und kann sich häufig ändern. Derartige Information ist daher schwierig in einer Wissensdatenbank aktuell zu halten und lässt sich besser in Volltext, wie z.B. aktuellen Nachrichten, finden.

In dieser Arbeit beschreiben wir in Kapitel 3.1 zunächst die Idee der *Semantischen Volltextsuche*, eine kombinierte Suche in Text und Wissensdatenbanken. Wir präsentieren einen voll funktionsfähigen Prototypen, der alle wichtigen Probleme adressiert, um Suchen einfacher und Ergebnisse präziser zu machen. Dazu gehören eine natürliche Sprachverarbeitung, ein einfach zu verwendendes Benutzerinterface und eine einfach zu verstehende Wissensdatenbank. Anhand des Prototypen evaluieren wir das Potential der Semantischen Volltextsuche. Im Weiteren Verlauf der Arbeit präsentieren wir drei individuelle Problemstellungen, um semantische Suche auf Text oder Wissensdatenbanken präziser und komfortabler zu machen. Diese Problemstellungen sind nicht nur relevant für Semantische Volltextsuche, sondern darüber hinaus auch für andere Arten der semantischen Suche.

In Kapitel 3.2 stellen wir eine natürliche Sprachverarbeitungstechnik vor, um Ergebnisse von Schlagwortsuchen in Text präziser zu machen. Anstatt der bloßen Existenz von Schlagwörtern an beliebigen Stellen in einem Dokument müssen die Schlagwörter innerhalb eines Satzes in „inhaltlichem Zusammenhang" stehen, um als Treffer in Betracht zu kommen. In

dieser Arbeit zeigen wir anhand von Experimenten, dass Semantische Volltextsuche durch diese Technik präziser wird. Wir zeigen auch, dass die Technik verwendet werden kann, um Tripel - ähnlich derer in Wissensdatenbanken - aus Text zu extrahieren.

Des Weiteren stellen wir in Kapitel 3.3 eine Technik vor um die Relevanz von Tripeln aus Wissensdatenbanken für typähnliche Relationen wie *profession* oder *nationality* zu berechnen. Anhand der Relevanz von Tripeln lassen sich Ergebnisslisten sortieren. Eine Liste amerikanischer Schauspieler in der Wissensdatenbank Freebase enthält z.B. mehr als 60.000 Einträge in beliebiger Reihenfolge. In einer nützlichen Sortierung erscheinen Schauspieler mit häufigen Hauptrollen vor Schauspielern mit nur vereinzelten Nebenrollen. Unser Ansatz bestimmt die Relevanz von Tripeln anhand von Informationen aus Text. Da dies ein neuartiges Problem ist, erstellen wir mittels Crowdsourcing einen Datensatz zur Evaluation. Dieser Datensatz ermöglicht es, unseren Ansatz mit Ähnlichen aus der Literatur zu vergleichen. Er ist außerdem öffentlich für weitere Forschung verfügbar und findet bereits Verwendung (z.B. WSDM Cup 2017).

In Kapitel 3.4 stellen wir einen Ansatz vor, um Anfragen in natürlicher Sprache aus einer Wissensdatenbank zu beantworten. Ein Benutzer kann Fragen in natürlicher Sprache stellen, wie zum Beispiel *who is the ceo of apple?*, anstatt eine spezielle Abfragesprache erlernen zu müssen. Ein großes Problem hierbei ist die Entitätserkennung und -zuordnung: welche Wörter aus der Frage entsprechen welchen Entitäten aus der Wissensdatenbank. Im Beispiel bezieht sich *apple* auf die Entität *Apple Inc.*, die Firma, nicht auf *Apple*, die Frucht, und *ceo* auf die Entität mit der synonymen Bezeichnung *Managing Director*. Unser Ansatz löst gleichzeitig sowohl die Entitätserkennung und -zuordnung, als auch das Übersetzen in die Abfragesprache durch eine Abbildung auf ein Learning-to-Rank Problem. Wir evaluieren unseren Ansatz auf zwei, weit verbreiteten Benchmarks und übertreffen zum Zeitpunkt der Veröffentlichung den Stand der Technik.

Semantische Suche stellt kein einzelnes, wohldefiniertes Problem dar. Daher haben wir eine umfangreiche Zusammenfassung (158 Seiten) über das Gebiet der semantischen Suche auf Text und Wissensdatenbanken erstellt. Darin wird das Gebiet in neun Bereiche klassifiziert, basierend auf den zugrundeliegenden Daten (Text, Wissensdatenbanken, Kombinationen davon) und der Art der Suche (Stichwörter, strukturiert, natürliche Sprache). Diese Klassifizierung beschreiben wir im letzten Kapitel dieser Dissertation und vermitteln so einen kurzen Überblick über dieses weitreichende Gebiet.

## Acknowledgments

First and foremost, I am grateful for my time at the University of Freiburg and the freedom I was provided to learn and explore during my studies at the Chair of Algorithms and Data Structures. Semantic search is a fascinating topic and the inspiring discussions, collaborations, and opportunities I encountered fostered my enthusiasm for it. I am especially thankful to my supervisor Hannah Bast who always provided guidance and support when I needed it, and who encouraged me to pursue a Ph.D. in the first place. My graduate studies were certainly one of the most challenging but also most rewarding times I have experienced so far.

I want to thank my colleagues Björn Buchhold, Claudius Korzen, Sabine Storandt, and Florian Bäurle. Our fruitful collaboration, lively discussions, as well as our travels to conferences around the world, were what made life at university exciting and diverse. Presenting months worth of research at conferences, receiving feedback from the audience, and discussing with peers is a reward on its own.

Last but not least, I want to thank my long-time partner and now wife, Simonette, who supported me throughout this dissertation and special period in my life. I cannot imagine having accomplished this without her.

# Contents

# 1 Introduction

This thesis contributes to the field of semantic search on *text* and *knowledge bases*. Semantic search is not a single well-defined problem, but an umbrella term for all kinds of *search with meaning*. The kind of search and queries we consider in this work can be characterized by the kind of result. Queries always ask for one or more entities. An example query is *who is the ceo of apple?*[1], but we also consider more specific queries like *male friends of Ada Lovelace who are philosophers*. In all cases, we are concerned with making the search *convenient*. This means making it easy to use, for example, by allowing a user to pose a question in natural language instead of a complicated query language. We are also concerned with answering queries *precisely*. That is, for a given query we want to match and return the correct result or, in the case of a result list, rank it such that most relevant results come first. The two data sources we consider for answering queries are text and knowledge bases. With respect to convenience and preciseness, these are two complementary sources of information. Let us understand this by first looking at search on text followed by search on knowledge bases. Afterwards, we describe the individual problems we address in this context.

Given a query consisting of keywords, search on text returns a list of documents that match the keywords, or variations of them. For example, issuing the query *first computer programmer* to a web search engine returns a list of web pages which match these keywords in their text, title, or domain. As a top hit, we get a document with the title "Ada Lovelace: The First Computer Programmer". And we get the same match for variations of query keywords like synonyms (*computing*) or spelling mistakes (*computre*). To match this document, the search engine doesn't have to understand the query or the document's content. In fact, the keywords match prominently in the title as well as in the document's text: *Ada Lovelace has been called the world's first computer programmer*. Together, this indicates to the search engine that this is a good hit for the query.

This kind of search has turned out to be very successful, which is somewhat surprising since the underlying idea is rather simple. A major factor in the success is the massive amount of data that is now available on the World Wide Web. For example, the English Wikipedia alone contains about five million articles with detailed information on popular topics. For popular queries, a web page with the answer that also matches the keywords is likely to exist and formulating a keyword query is easy and intuitive for a user. Furthermore, the search engine requires no deep understanding of the text or query. Instead, these are treated as a mostly opaque and *unstructured* source of information.

[1] This example is from a benchmark [20] we use in one of our publications [17].

Of course, major search engines nowadays apply a myriad of techniques to improve this kind of search. For example, they analyze query logs and search sessions [77, 32], personalize results [59], and learn sophisticated ranking strategies [57]. However, the underlying search paradigm still reaches a limit for queries that require a deeper understanding or the combination of multiple sources of information.

Consider issuing the query *male friends of ada lovelace who are philosophers* on a web search engine. Certainly, the information is available somewhere on the Web, but likely not in a single document.[2] For example, the Wikipedia article on Ada Lovelace mentions that she was friends with Charles Baggage, but only his Wikipedia article mentions that he was a philosopher and nowhere is explicitly mentioned that Charles Baggage is male. However, the search still produces results, because keywords match "randomly" in various parts of documents, e.g., some keywords may match in the title, some in the abstract, and some in the main text. The user then needs to inspect each document, examine whether the content is helpful, and manually compile the list of persons she is looking for. What is needed is a deeper understanding of the query and documents.

Long queries with a narrow and specific intent as in this example are more common than one might expect. It has been observed that the length of queries on major search engines follows a power law distribution [46, 3]. Therefore, a significant proportion of queries consists of long queries. Bendersky and Gupta [46] cite that 17% of search engine queries contain five or more keywords. They note that long queries often have a narrow and specific intent and consist of composite queries with different subqueries - like the query above. They further argue that such queries may become even more frequent with the increasing availability of voice search, dialogue, and question answering systems.

While text contains a wealth of information in unstructured form, *knowledge bases* contain *structured* information. Here are some example statements about Ada Lovelace and Grace Hopper in the form of *subject predicate object* triples:

| | | |
|---|---|---|
| Ada Lovelace | is-a | Computer Scientist |
| Ada Lovelace | gender | Female |
| Ada Lovelace | place-of-birth | London |
| Grace Hopper | is-a | Computer Scientist |
| Grace Hopper | gender | Female |
| Grace Hopper | place-of-birth | New York City |

Such statements, or *triples*, follow a precise and pre-defined schema. For example, there is only one way of expressing the birthplace of Ada Lovelace, whereas, in text, the same fact can be expressed with various sentences.

[2] We ignore the unlikely case where someone has manually compiled such a list.

Triples can be extracted automatically, e.g., from Wikipedia Infoboxes, or collected manually. The currently largest general purpose knowledge base is Freebase[3] [24], which contains a mix of automatic extractions and manual curations totaling about three billion statements on 40 million entities.

To retrieve statements from a knowledge base a *structured query language* called SPARQL[4] is used. The following shows an example query for *female computer scientists and their place of birth*:[5]

```
select ?entity ?place where {
  ?entity  is-a  Computer Scientist .
  ?entity  gender  Female .
  ?entity  place-of-birth  ?place .
}
```

The result is an unordered list of tuples of persons and their place of birth, containing, for example, *(Ada Lovelace, London)* and *(Grace Hopper, New York City)*. In contrast to search in text, the semantics of the query is precisely defined by the query language, as is the content of the knowledge base. If the answer is part of the knowledge base it can be retrieved via the correct query. However, formulating the query can be difficult for a user in the first place. It requires understanding the query language and the knowledge base schema, for example, what predicates and entities exist and what their identifiers are.

Knowledge bases allow precise and semantic queries, however, there will always be some information that is too specific to be included in a knowledge base, like who is friends with whom.[6] As a result, only a relatively small part of the world's knowledge is available in structured form. This is supplemented by a wealth of information available in full text. Combining search on knowledge bases and search on full text allows answering queries which cannot be easily answered by either one. Consider again the query for *male friends of ada lovelace who are philosophers*. A knowledge base query can be used to obtain a list of male philosophers. Finding out who was friends with Ada Lovelace can then be achieved via a full-text query. We consider the combined search on text and knowledge bases a promising search paradigm and present a viable approach as part of this work.

[3] Freebase was acquired by Google in 2010 and has been discontinued in 2015. All of the data is supposed to be migrated to Wikidata [82].

[4] `https://www.w3.org/TR/rdf-sparql-query/`

[5] We glance over a few syntax details of SPARQL for better readability, e.g., we omit namespaces.

[6] This has two reasons: First, adding new information in an accurate and complete way and keeping it up to date is expensive. Second, some circumstances are difficult to express in structured form, for example, sentiment or controversial or vague statements.

In this document, we summarize our work on problems that arise when answering semantic search queries using text, knowledge bases, or their combination. The queries above are good examples of the kind of queries we consider. They involve entities and their attributes, and they have one or more entities as a result. Such *entity queries* have been shown to make up a significant proportion of queries. In a query log analysis of a major web search engine, [70] found that almost 60% of queries had entities or their attributes as the desired result and more than 70% of all queries contained an entity or attribute. For academic search, [54] find that even 92% of all queries contain an entity.

We are concerned with making search *convenient*. This means making the search easy to use. For example, we present an approach that answers natural language questions, arguably the most convenient way of searching for a user. We are also concerned with answering queries *precisely*. That is, for a given query we want to match and return the correct result or, in the case of a result list, rank it such that most relevant results come first. In this context, we address and contribute to the following problems.

In Section 3.1, we introduce *semantic full-text search*, a combined search on text and a knowledge base. We present a fully functional prototype that includes our approaches on problems to make the search more precise and convenient. For example, we apply a novel natural language processing technique, *contextual sentence decomposition*, that returns more precise and semantic matches from the full text than the conventional matching strategy described above. We also provide an intuitive user interface with context-sensitive suggestions to construct queries, and we create a knowledge base that is easy to understand and use. In a quality evaluation, we show the potential of semantic full-text search.

Afterwards, we describe three individual problems that make search on text or knowledge bases more precise and convenient. These problems are not only relevant for semantic full-text search, but have applications beyond and on their own.

In Section 3.2, we describe how to perform contextual sentence decomposition. We already show in Section 3.1 that this technique can be used to get more precise results for semantic full-text search. In Section 3.2, we extend the technique to extract triples from text, similar to those contained in a knowledge base. In an evaluation, we compare our approach to existing approaches from the literature and show that our extracted triples are preferable for applications in semantic search.

In Section 3.3, we address how to return precise results for certain knowledge base queries. For example, on Freebase, a query for *american actors* returns a list of 64,757 actors in no particular order.[7] We present a technique to compute relevance scores in order to rank results such that, e.g., actors with frequent leading roles are ranked before those who only had few supporting roles. Crucially, we compute relevance scores from text, because

[7] This example is taken from our publication [15].

the required information is not reliably present in the knowledge base. Since this is a novel problem, we provide a benchmark created via crowdsourcing. We propose a variety of algorithms to compute ranking scores and evaluate them using our benchmark. The benchmark is also publicly available and already fosters further research on the problem (e.g., WSDM Cup 2017 [48]).

In Section 3.4, we show how to answer natural language questions from a knowledge base by automatically translating into a SPARQL query. A user can, for example, ask *who is the ceo of apple?* and is presented with the result from the knowledge base. This makes the search more convenient since the user doesn't have to formulate a structured query. A major problem in the translation is *entity recognition and disambiguation*: which words from the question correspond to which entities from the knowledge base. In the example, *apple* refers to the entity *Apple Inc.*, the company, not to *Apple*, the fruit, and *ceo* to the entity *Managing Director*, a synonym. Our approach solves both, entity recognition and disambiguation and query translation, by mapping to a single learning-to-rank problem. The approach focuses on giving precise results and improved the state of the art on two widely used benchmarks at the time of publication.

As we stated at the very beginning, semantic search is not a single well-defined problem. To give an overview of the vast field, we wrote an extensive survey on semantic search on text and knowledge bases [16] (156 pages). The survey classifies the field into nine groups based on the type of data that is used (text, knowledge bases, and their combination) and the kind of search that is performed (keyword, structured, and natural language). We describe the survey and our classification in Section 3.5 and thereby provide a short overview of the vast field.

This document summarizes our publications on the problems introduced above and is structured as follows. Chapter 2 lists all of our publications and attributes the work to individual authors. In Chapter 3, we present the core of our work on these problems. Each section, 3.1 to 3.4, describes one problem and corresponds to one or more publications. In each section, we first give a concise description of the problem, related work, and our contributions. Then, we present the main ideas behind our approach, followed by a description of our experiments and main results. In all parts, we focus on the core of our approach. Detailed technical descriptions are available in the corresponding publications. In many places, we refer to our survey, which contains an extensive overview of a lot of recent related work. Finally, we conclude this dissertation with an outlook on future work in Chapter 4.

# 2 List of Publications

The following first lists our peer-reviewed publications followed by non peer-reviewed publications. For each publication, we give a short description and attribute the work to individual authors. The publications are grouped by topic which are ordered by how we describe them in the rest of this document. Note that authors for each publication are listed alphabetically by convention.

In the digital version of this dissertation, titles below (and in the references) are clickable and lead to an electronic copy.

## 2.1 Publications With Peer Review

**A Case for Semantic Full-Text Search [9], Position Paper, SIGIR-JIWES 2012**
*DOI*: *10.1145/2379307.2379311*
*Hannah Bast, Florian Bäurle, Björn Buchhold, and Elmar Haussmann*

Position paper that motivates semantic full-text search and describes the requirements and resulting challenges. We address these as part of the papers below.

All authors wrote the paper.

**Semantic Full-Text Search with Broccoli [12], SIGIR 2014**
*DOI: 10.1145/2600428.2611186*
*Hannah Bast, Florian Bäurle, Björn Buchhold, and Elmar Haussmann*

Demo paper that presents our semantic full-text search prototype, its web application, and its public API. Covered in Section 3.1.

Research and implementation is based on the work listed below. All authors wrote the paper.

**Easy Access to the Freebase Dataset [11], WWW 2014**
*DOI: 10.1145/2567948.2577016*
*Hannah Bast, Florian Bäurle, Björn Buchhold, and Elmar Haussmann*

Demo paper that describes how to transform Freebase into a knowledge base that is easy to use and search. The paper also presents a web application that provides convenient access. Covered in Section 3.1.

HB, BB, and EH conducted the research. BB and EH implemented the ideas. FB adapted the semantic full-text search interface for the web application. HB, BB, and EH wrote the paper.

**Open Information Extraction via Contextual Sentence Decomposition [19], ICSC 2013**, *DOI: 10.1109/ICSC.2013.36*
*Hannah Bast and Elmar Haussmann*

Research paper that describes how to identify the parts of a sentence that semantically "belong together". The paper also describes how to extend the technique to extract triples from text (Open Information Extraction). Covered in Section 3.2.

Both authors conducted the research and designed the evaluation. EH implemented the ideas and performed the evaluation. Both authors wrote the paper.

**More Informative Open Information Extraction via Simple Inference [18], ECIR 2014**, *DOI: 10.1007/978-3-319-06028-6_61*
*Hannah Bast and Elmar Haussmann*

Research paper that describes how to apply simple inference to increase informativeness of triples from Open Information Extraction. The technique can be incorporated into the triple extraction process from above [19]. Covered in Section 3.2.

Both authors conducted the research and designed the evaluation. EH implemented the ideas and performed the evaluation. EH wrote most of the paper with guidance and input by HB.

**Relevance Scores for Triples from Type-Like Relations [15], SIGIR 2015**
*DOI: 10.1145/2766462.2767734*
*Hannah Bast and Björn Buchhold and Elmar Haussmann*

Research paper that describes how to compute relevance scores for knowledge base triples. The scores can be used to properly rank results of entity queries on a knowledge base. Covered in Section 3.3.

All authors conducted the research and designed the crowdsourcing experiment and evaluation. BB and EH implemented the ideas and performed the evaluation. All authors wrote the paper.

**WSDM Cup 2017: Vandalism Detection and Triple Scoring [48], WSDM 2017**
*DOI*: *10.1145/3018661.3022762*
*Stefan Heindorf, Martin Potthast, Hannah Bast, Björn Buchhold, and Elmar Haussmann*

Overview paper for the WSDM Cup 2017 and its two tasks, vandalism detection and triple scoring. The task is based on the work described in Section 3.3.

SH and MP organized the vandalism detection task. HB organized the Triple Scoring task. HB, BB and EH defined the triple scoring task, created a benchmark via crowd sourcing, and defined sensible evaluation metrics. SH, MP, and HB wrote the paper.

**More Accurate Question Answering on Freebase [17], CIKM 2015**
*DOI*: *10.1145/2806416.2806472*
*Hannah Bast and Elmar Haussmann*

Research paper that describes how to translate natural language questions to SPARQL queries. This can be used to perform question answering on a knowledge base. Covered in Section 3.4.

Both authors conducted the research and designed the evaluation. EH implemented the ideas and performed the evaluation. Both authors wrote the paper.

**Semantic Search on Text and Knowledge Bases [16], FNTiR 2016**
*DOI*: *10.1561/1500000032*
*Hannah Bast and Björn Buchhold and Elmar Haussmann*

An extensive survey (156 pages) over the huge field of semantic search on text and knowledge bases. Covered in Section 3.5.

All authors contributed in deciding the overall structure and scope of the survey. All authors surveyed the literature and prepared summaries for systems to include or exclude. All authors wrote the survey.

## 2.2 Publications Without Peer Review

The following lists one publication at a non peer-reviewed venue and one invited publication.

**Broccoli: Semantic Full-Text Search at your Fingertips [10], CoRR 2012**
*Hannah Bast, Florian Bäurle, Björn Buchhold, and Elmar Haussmann*

Research paper that describes a semantic full-text search prototype, Broccoli, including all important components: natural language pre-processing, search index, and user interface. Covered in Section 3.1.

All authors conducted the research on the general search paradigm, system design, and user interface. HB and EH conducted the research on contextual sentence decomposition. EH implemented it. HB and BB conducted the research on the search index and query processing and BB implemented it. BB, FB, and EH implemented the data pre-processing steps for the prototype (Wikipedia XML parsing, tokenizing, entity recognition and linking). FB implemented the user-interface. All authors performed the evaluation and wrote the paper.

**A Quality Evaluation of Combined Search on Knowledge Base and Text [14], Invited Paper at KI-Journal 2017**, DOI: *10.1007/s13218-017-0513-9*
*Hannah Bast, Björn Buchhold, and Elmar Haussmann*

Research paper that describes a detailed quality evaluation and error analysis of our KB+Text search paradigm. Covered in Section 3.1.

All authors designed the evaluation and analysed results. BB and EH performed most of the manual evaluation and error analysis. All authors wrote the paper.

# 3 Contributions

## 3.1 Semantic Full-Text Search

Semantic full-text search combines the capabilities of full-text search and search in knowledge bases. We have already motivated the idea in the introduction. In this section, we describe the idea in detail and present a fully functional prototype. The prototype addresses all important problems involved, including a natural language pre-processing technique to find precise matches in full text, an easy-to-use knowledge base, and an intuitive user interface to make the search convenient. We also describe an extensive quality evaluation that shows that our natural language pre-processing considerably improves results and that our prototype can answer a wide range of questions. A demo of our prototype, Broccoli, is available at `http://broccoli.cs.uni-freiburg.de`.

*This section summarizes the work published in [10] at CoRR 2012, [9] as a position paper at SIGIR-JIWES 2012, [12] at SIGIR 2014, [11] at WWW 2014, and [14] at KI-Journal 2017 (under submission).*

### 3.1.1 Problem, Related Work, and Contributions

Consider the example query for *plants with edible leaves and native to Europe*.[8] This query can be expressed in semantic full-text search in the following way:

**select** ?entity **where** {
  ?entity is-a Plant .
  ?entity native-to Europe .
  ?entity occurs-with "edible leaves"
}

The query language is a subset of SPARQL extended with the special *occurs-with* relation. For the example, it combines search in full text and a knowledge base as follows. The first two triple patterns (*is-a Plant*, *native-to Europe*) match entities that are a plant and native to Europe in the knowledge base. One of the matches is, besides many other plants, broccoli. The special relation *occurs-with* requires that each plant, such as broccoli, is mentioned along the words *edible* and *leaves* somewhere in the full text. This requires that mentions of entities, e.g., broccoli, have been identified in the full text. For example, in the following sentence, matching words and entities are underlined:

*The edible portions of broccoli are the stem tissue, the flower buds, as well as the leaves.*

[8] This and the following example queries and sentences are taken from our publication [10].

In principle, this co-occurrence doesn't have to be restricted to a sentence but could also be within a larger unit, like a paragraph or document. The co-occurrence can also be in any text document, for example, in the Wikipedia article for broccoli or on a gardening website about leafy green vegetables. What is essential for a correct result, however, is that the co-occurrence provides evidence that broccoli indeed has edible leaves. We come back to this important problem below. Together, the results from the knowledge base and the full text can be used to infer a list of plants that match the query.

This kind of search combines the strengths of search on text and search on knowledge bases. Large knowledge bases, such as Freebase, focus on facts that are easy to define and extract automatically. A plant's origin is, for example, often part of Wikipedia's infobox. This makes it easy to find for an extraction system in order to populate the knowledge base. Once part of the knowledge base, it is straightforward to obtain a list of plants that are native to Europe, given the correct query.

Finding a full-text query to obtain this list from text is difficult. An answer requires factual knowledge in the first place, for example, that broccoli is a plant or that Italy (the originating country of broccoli) is a part of Europe. To determine whether broccoli has edible leaves it is the other way around. Corresponding facts are rather specific, more difficult to extract automatically, and, therefore, unlikely to be included in a knowledge base. However, the information is likely to be present in text, and it is easy to formulate a full-text query. The combination of both search paradigms allows answering queries that cannot be easily answered by either one.

Semantic full-text search uses *combined data*: a knowledge base as well as text in which entity mentions from the knowledge base are identified (like broccoli in the example sentence above). Queries are *semi-structured*: They are from a subset of SPARQL where the structured part matches facts in the knowledge base while the keywords given via the *occurs-with* relation match in the full text. There are two major approaches for searching in combined data that is followed in related work.

One prominent approach is that of creating *virtual documents*. All of the information pertaining to a specific entity is collected in a virtual document. For example, a document can be constructed for an entity by adding the subject, predict, and object of triples that have the entity as subject or object, together with text associated with that entity, like its Wikipedia page. Search can then be performed via classical keyword search. Since each document corresponds to an entity, results are lists of entities. This also allows applying traditional ranking functions from information retrieval, like BM25. Compared to semantic full-text search, keyword queries are more convenient to formulate for a user. However, our query language is far more powerful and allows more precise searches. The drawback is that a structured query needs to be constructed in the first place.

The virtual document approach has been popular, especially for searching the Semantic Web. The Semantic Web consists of triples similar to those in a knowledge base. Triples can be contributed by anybody, for example, via semantic markup in web pages. Because content can also be referenced and interlinked, the data from the Semantic Web is also referred to as *linked open data (LOD)*. No global schema of triples is enforced. Many of the triples contain literals so that a huge part of the data is actually text. Therefore, the two main challenges that approaches need to overcome are the sheer amount of data and the inconsistent schema.

Two established benchmarks to evaluate approaches for searching the Semantic Web are the TREC Entity Tracks (2010 and 2011) [4, 5] and the SemSearch challenges (2010 and 2011) [47, 81]. Two example queries from these benchmarks are *airlines that use boeing 747 airplanes* and *astronauts who walked on the moon*. The benchmarks come with a set of queries and desired results as well as a collection of triples extracted from the Semantic Web. Indeed, the best performing systems used keyword search on virtual documents with special attention to ranking. For example, [23] use a fielded inverted index with an adaptation of BM25 that boosts matches from certain relations. Since the kind of queries in these benchmarks are good examples for semantic full-text search, we used the two benchmarks in the evaluation of our prototype (see below).

The second prominent approach for searching in combined data is to perform keyword search on text, identify entities in the results, and rank them. Information from a knowledge base is usually applied in the ranking phase, but without formulating or translating to a precise SPARQL query that is used to restrict the result set. For example, the system described in [42] (the winning submission at the TREC Entity Track in 2009 [6]) issues queries to Google and identifies entities in the result documents using named entity recognition and disambiguation techniques. This establishes a link between the entity and its mention in text. Entities are then ranked using a combination of relevance scores based on how well the entity and the text surrounding the entity mention (document, sentence, or web table) match the query. In [76], the authors use the knowledge base to identify related similar entities (e.g., via the number of hops from one entity to another) that weren't matched prominently in text but that should be ranked higher. In contrast, semantic full-text search allows very precise and complex queries on the knowledge base that restrict the result set in the first place. Then, a simplistic ranking suffices to give good results as we show in our evaluation.

Besides the two approaches discussed above, there are various approaches for performing semantic search on text and knowledge bases alone. Details on related work in all of these areas is available in our survey on semantic search [16]. There are a few systems that perform combined search using (semi-) structured queries in a similar way to ours. We describe their difference to our semantic full-text search prototype below.

A good prototype implementation of semantic full-text search should give correct results and be easy to use. This requires the following, which we elaborate on next: finding correct matches in text with the *occurs-with* relation, an easy to use knowledge base, an intuitive user interface, and an efficient index and query processing. We have addressed these problems in our fully functional prototype, called Broccoli. Figure 1 shows a screenshot of our final prototype answering the query from above.

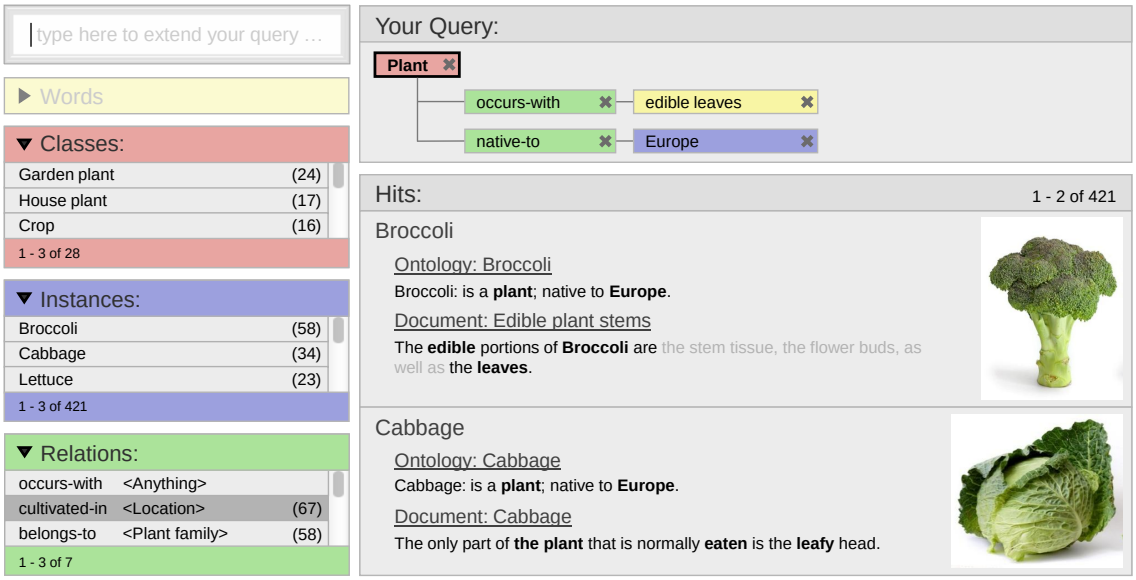


**Figure 1.** From our publication [10]: A screenshot for our example query. The structured query is visualized as a simplified tree on the top (the triple *?entity is-a Plant* is expressed by the root). Below, results are grouped by entity. Each result provides evidence from both the knowledge base and the full text. The search field on the top left can be used to extend the query. Below, suggestions for sub*classes* and *instances* of plants and additional *relations* are shown. These can be used to further refine the query. The suggestions are *context-sensitive*: They take the query so far into account and always lead to results. More screenshots can be found in [10].

As mentioned above, an important component of semantic full-text search is the *occurs-with* relation. It allows specifying co-occurrence of entities with words in text. Identifying that this co-occurrence is *semantic* is essential for the full-text part of queries. Consider the query *plants with edible leaves* and the following sentence with underlined matches:

*The usable parts of rhubarb are the edible stalks, however its leaves are toxic.*

All query elements (a plant, *rhubarb*, and words *edible leaves*) co-occur in the sentence, however, rhubarb is clearly a wrong answer (its leaves are toxic). Indeed, this problem is even more pronounced when restricting co-occurrence to the same paragraph or document, as we show in the experiments below. Ideally, the above match should be avoided, but it should still match a query for *plants with edible stalks*. We address this problem by

identifying words that "semantically belong together", which we call *contexts*. Intuitively the words *rhubarb*, *edible* and *leaves* do not belong together in the sentence above but the words *rhubarb*, *edible*, *stalks* do. Once identified, co-occurrence can be restricted to contexts and matches are more likely follow the intent of the query. The technique behind this idea is called *contextual sentence decomposition* and described in detail in Section 3.2. In this section, we introduce the basic idea and present an evaluation of the impact on semantic full-text search.

For finding results in text, formulating a keyword query via the *occurs-with* relation is easy, and the challenging part is semantically matching the keywords. This is the opposite for structured queries on a knowledge base. The result set is well defined and easy to retrieve because it is fully specified via the query language. However, the structured query can be difficult to construct in the first place. It uses unique entity as well as relation identifiers from the knowledge base. These are often opaque and difficult to determine. On top, the query must adhere to the (usually complex) schema of the knowledge base. For example, in Freebase, which we use for our prototype, the (seemingly) simple query for the *winners of the Palme d'Or* looks like this:[9]

```
select ?name where {
  ?x  ns:award.award_winner.awards_won  ?m  .
  ?m ns:award.award_honor.award  ?a  .
  ?a  ns:type.object.name  "Palme d'Or"@en  .
  ?x  ns:type.object.name  ?name  .
}
```

Who won a Palme d'Or is expressed via an intermediate *award nomination* object (*?m*) that connects the award (*?a*) and the person receiving the award (*?x*). The intermediate object is often referred to as *mediator* and allows to express n-ary relations, e.g., to link the person not only to the award but also to the date it was received and for which work it was awarded. Constructing the query above is not feasible without being acquainted with the schema of the knowledge base. The unwieldy relation and entity identifiers are especially problematic. This is apparent in Figure 1, which would look unpleasant without readable entity and relation identifiers.

We follow two approaches to tackle this problem. On the one hand, we construct a curated version of Freebase with simplified schema and readable identifiers. In this process, we resolve, for example, the complex award nomination relation from above to a binary *awards-won* relation. We also give entities their canonical and readable name as an identifier. Given the size of Freebase, this curation needs to be an automatic process.

[9] For readability, we omit namespace prefix definitions. The example query is from our publication [11].

On the other hand, we design a user interface that helps in incrementally constructing queries. A user can start to construct a query by selecting a class or entity (like *Palme d'Or*) and is presented with relations with meaningful names (e.g., *awards-won*) for this class or entity. At each step, she is guided by meaningful *context-sensitive* suggestions. Whenever the query is extended, new suggestions take the query so far into account so that all suggestions actually lead to results. This avoids constructing queries without results. Together, the curated knowledge base and user interface make it more convenient to explore the knowledge base and construct the correct query.

Once a query is constructed, the text and knowledge base must be searched to retrieve results. Besides reasonable query times ($<$ 100ms per query) several features are essential for our user interface, in particular, context-sensitive suggestions for words, classes, entities, and relations. Since no existing index structure provides this, we developed a special index that can answer our queries efficiently. The index and query engine is explained in detail in [13]. We don't consider it here any further since the corresponding work is not part of this thesis.

There are several related systems that perform *Semi-Structured Search on Combined Data* [16, Section 4.6.2]. ESTER [8] was one of the first systems. It uses a special-purpose index that also provides query suggestions from the text and knowledge base after each keystroke. Compared to ESTER our user interface is far more involved and instead of documents our system returns lists of entities. KIM [69] and its successor Mímir [80] allow semi-structured search via two different indices based on off-the-shelf software, one for text and one for the knowledge base. This causes efficiency issues when the structured part of the query matches many entities (e.g., a query that requires the list of all persons in Freebase). Both systems return lists of documents and provide no context-sensitive user interface. STICS [49] finds documents with mentions of keywords, entities, or entities of a certain category. A knowledge base provides the information which entity belongs to which category, but no further relations from the knowledge base can be used in queries. Suggestions for entities and categories are provided by a user interface and ranked by coherence. For example, as the authors note, the suggestions for the category *Ukranian politicians* rank the Klitschko brothers higher, once *Angela Merkel* is added to the query, since they are long-time German residents. All of the systems above lack our natural language processing technique to improve matches in the full text.

In the next subsection, we describe our approaches to the problems discussed above by means of our semantic full-text search prototype, Broccoli. Our main contributions are the following:

- The novel idea of semantic full-text search, which combines search in structured data with search in full text. Queries are a subset of SPARQL with a special *occurs-with* relation that can be used to specify co-occurrence of entities and words in text. In contrast to previous approaches, results are lists of entities (not documents).
- A novel kind of pre-processing that decomposes a sentence into contexts: parts that semantically "belong together". This is important for precise matches with the *occurs-with* relation. The technique behind this decomposition is called *contextual sentence decomposition* and described in detail in Section 3.2. In our evaluation in this section, we show that it makes the search more precise.
- A curated version of Freebase with unique readable entity and relation names and a simplified schema. We also improved the taxonomy, and the knowledge base comes with entity popularity scores for ranking. This makes it is easy-to-use, both for semantic full-text search and in other applications.
- An implementation of above components into a fully functional prototype system with an intuitive user interface. The user can incrementally construct her queries and is guided by meaningful suggestions on how to extend the query at each point.
- An extensive evaluation showing the potential of semantic full-text search in general and the improvements due to contextual sentence decomposition in particular.

### 3.1.2 Approach

On a high level, our semantic full-text search prototype, Broccoli, works as follows. First, a pre-processing pipeline takes as input a knowledge base and text. The pipeline links entity mentions in text to the knowledge base and decomposes sentences into contexts. The resulting contexts and knowledge base facts are stored in an index, which also provides a query engine with context-sensitive suggestions. The query engine is instrumented by a user interface which helps in constructing and executing queries and displays results. We start by explaining the knowledge base we use, followed by a description of the pre-processing pipeline and the user interface.

**Easy-to-use knowledge base**. As we illustrated above, the knowledge base should be convenient for an end user. In particular, it should have readable entity identifiers and an intuitive schema. This is not provided by the largest publicly available knowledge

base, Freebase. We transform Freebase into an easy-to-use version using a few simple yet effective techniques, which we describe in [11] and summarize below.

First, we obtain popularity scores for each entity by combining the number of mentions in a reference text corpus and the number of knowledge base triples where the entity occurs as subject or object. As a reference corpus, we use ClueWeb'12 [34] with entity mentions by [45]. We use these scores to assign the most popular entity the canonical name and resolve conflicts for other entities with the same name via heuristics. For example, for persons, we append the notable profession, as in *Michael Moore (Soccer Forward)*, and for locations the containing country, as in *Berlin (United States)*.

To simplify the schema, we resolve n-ary relations using a heuristic based on frequencies of involved facts. Intuitively, more facts connect award winners to their award than for what work or at what date it was won. Therefore, we create a new binary relation *Awards-Won* and ignore facts about the date or award-winnning work involved. The query for the winners of the Palme d'Or is now simple and intuitive:[10]

**select** ?$x$ **where** { ?$x$ Awards-Won "Palme d'Or" }

As part of the curation, we also merge duplicate types (like *Person* and *person*) by checking for overlap on their members. Finally, we compute the transitive closure for manually selected relations like *profession* and *specialization-of*. We use this to enhance the taxonomy, to ensure that, for example, a person who has the type *Physicist* (the specialization) also has the type *Scientist*. All of the above problems are addressed in an automatic way. The only required manual input is which relations to compute the transitive closures over. Details on the applied heuristics can be found in the corresponding publication [11].

**Pre-processing pipeline**. The pre-processing pipeline takes a knowledge base and text as input. Its main tasks are to link entity mentions in the text to the knowledge base, split the text into sentences, and decompose the sentences into contexts. We describe the pre-processing in more detail in [10].

Consider the following sentences, where mentions of the entity rhubarb are underlined:

> *The stalks of rhubarb are edible and its roots are medicinally used. However, the leaves of the plant are toxic.*

Identifying the first mention of rhubarb in the sentence is also known as *named entity recognition and disambiguation* or *entity linking*. This is an established problem, and a large body of approaches exists that solve this for any text input, e.g., [89, 72, 43]. The next two mentions, *its* and *the plant*, are references to the first mention of rhubarb. Identifying these is also known as *co-reference* or *anaphora resolution*.

[10] The example query is from our publication [11].

In our prototype, we use the following simple heuristics to identify all of these mentions. Since we use Wikipedia as text, we can obtain initial entity mentions from its markup: as an annotation rule (by Wikipedia) the first mention of an entity always references its Wikipedia page. By mapping Wikipedia pages to entities in the knowledge base[11] we obtain entity mentions of higher precision than would be possible with automatic approaches. We then resolve references as follows. Whenever a part or the full name of an entity is mentioned again, we recognize it as that entity (for example, *Ada* as *Ada Lovelace*). This is restricted to the same section of a document, which our pipeline is also able to identify. Additionally, we resolve pronouns to the last identified entity of matching gender and identify the pattern *the <class>* as the last entity of matching class. This identifies the references of *its* and *the plant* to rhubarb in the example above.

Given entity mentions in each sentence, the next step is to decompose sentences into parts that "belong together". This is important to achieve precise matches with the *occurs-with* relation. For example, consider the query *plants with edible leaves* and the following matching sentence with underlined matches:[12]

> *The usable parts of rhubarb, a plant from the Polygonaceae family, are the medicinally used roots and the edible stalks, however its leaves are toxic.*

We decompose this sentence into the following contexts:

(C1) *rhubarb, a plant from the Polygonaceae family*
(C2) *The usable parts of rhubarb are the medicinally used roots*
(C3) *The usable parts of rhubarb are the edible stalks*
(C4) *however rhubarb leaves are toxic*

After decomposition, the query for *plants with edible leaves* no longer matches any of these contexts, since *rhubarb* no longer co-occurs with the words *edible* and *leaves*. Note that entities and co-references (*its* refers to *rhubarb*) have been identified beforehand. Thus, no information is lost as part of this process. The technique for this is called *contextual sentence decomposition* and covered in detail in Section 3.2. It takes a deep grammatical parse tree of the sentence as input and applies tree transformations to retrieve contexts.

The pipeline performing entity linking and sentence decomposition as described above is implemented as a UIMA[13] chain. This allows us to easily exchange components, for example, to switch to a different entity linking strategy. The pipeline also contains components that parse the Wikipedia markup and split the text into (Wikipedia) sections, sentences, and words. UIMA also allows to scale out the pre-processing to a cluster of servers for computation intensive tasks like the required sentence parsing for our decomposition.

[11] Freebase provides the corresponding Wikipedia page for each entity, if it has one.
[12] The example is taken from our publication [10].
[13] http://uima.apache.org/

**Index and user interface**. Given the knowledge base and pre-processed text, we construct our search index. The index and query engine are explained in detail in [13], which is not part of the work of this thesis. The query engine reads results from the index and applies a simplistic but, as our evaluation shows, effective ranking. For a query that only uses the knowledge base, result entities are ranked by their popularity (see above). For queries that use the full text, results are ranked by their number of matches in text. The query engine also provides context-sensitive suggestions to the user interface.

We designed the user interface in an iterative process in which we developed ideas and features, tested, and then revised them. In the end we came up with the user interface already shown in Figure 1. It has the following main features as listed in our publication [10], where we also show additional screenshots:

- Search as you type: Each keystroke updates the results and suggestions. Suggestions are context-sensitive. They actually lead to hits and the higher-scored the hits, the higher scored their suggestion.
- Pre-selected suggestions: New users may be overwhelmed by the multitude of suggestions, therefore, the most likely suggestion is automatically highlighted.
- Visual query representation: The current query is always visualized as a tree. Suggestions are displayed for the element in focus, which can be changed using clicks.
- Transparency: Results are grouped by entity, and displayed together with context snippets that provide full evidence from the knowledge base and text.

The features above make it more convenient to construct queries. Furthermore, they make results transparent. While the first is important for new users the latter is especially important for researchers to identify mistakes. This helped in the analysis and evaluation of the system which we describe next.

### 3.1.3 Experiments and Results

Here, we report on our quality evaluation of our semantic full-text search prototype, Broccoli, which we describe in our publication [14]. We performed the evaluation with two goals in mind. First, we wanted to draw conclusions on the general potential of semantic full-text search. Second, we wanted to evaluate to what extent decomposing sentences into contexts makes results more precise.

The following was our experimental setup. As a text corpus, we used the English Wikipedia from January 2013. We performed entity recognition and contextual sentence decomposition as described above. This way we recognized a total of 285 million entity occurrences and decomposed 200 million sentences into 418 million contexts. To keep running times reasonable we scaled out this computation to a cluster of eight servers, each with an 8-core CPU and 16 GB of main memory. As a knowledge base, we used YAGO [79], which has about 26.6 million facts on 2.6 million entities. YAGO is smaller than Freebase but was the most promising knowledge base at that time. A larger knowledge base, like our curated version of Freebase from above, would improve all results presented here since more questions are answerable directly from the knowledge base. The conclusions we draw on the potential of semantic full-text search in general and the improvement due to context decomposition are valid nonetheless.

We evaluated search quality using three benchmarks. Each benchmark consists of a set of queries and a corresponding ground truth of relevant results, i.e., lists of entities. We used 15 queries from the TREC Entity Track in 2009 [6]. An example query is *Airlines that currently use Boeing 747 planes.* From the SemSearch Challenge 2011 [81], we used 46 queries, for example, *Apollo astronauts who walked on the moon.* We also derived ten queries from manually compiled Wikipedia lists. These can be thought of as a query with the ground truth provided by the corresponding list, for example, the *List of participating nations at the Winter Olympic Games.*

For each benchmark, we manually translated each query into our query language in a straightforward way (i.e., we didn't tune the queries to the results). All constructed queries make use of the text corpus via the *occurs-with* relation. The ground truth for each query is usually a small, well-defined result set. We believe that in this case, the quality of the result set as a whole is more important than the ranking within the result set, i.e., the result set should be as precise as possible. Therefore, we first focused on set related measures: precision, recall, and F1, the harmonic mean of precision and recall.

We distinguished between three variants of semantics of *occurs-with*: co-occurrences restricted to *sections*, *sentences*, and *contexts*. Table 1 shows our results. For all benchmarks, sections achieve the highest recall. However, a lot of returned results are due to "random" co-occurrences within a section. This causes a large amount of false-positives resulting in the smallest precision and F1 score. Compared to sections, sentences decrease the number of false-positives resulting in a higher precision but at a cost of recall. Our contexts further decrease the number of false-positives and also slightly increase the number of false-negatives[14]. This results in an increased precision with a slight decrease in recall. However, the increase in precision is far more pronounced, so that overall, contexts yield the best precision and F1 score on all three benchmarks.

| | | #FP | #FN | Prec. | Recall | F1 |
|---|---|---|---|---|---|---|
| SemSearch | sections | 44,117 | 92 | 6 % | 78 % | 9 % |
| | sentences | 1361 | 119 | 29 % | 75 % | 35 % |
| | contexts | 676 | 139 | 39 % | 67 % | 43 %† |
| Wikipedia lists | sections | 28,812 | 354 | 13 % | 84 % | 21 % |
| | sentences | 1758 | 266 | 49 % | 79 % | 58 % |
| | contexts | 931 | 392 | 61 % | 73 % | 64 %∗ |
| TREC | sections | 6890 | 19 | 5 % | 82 % | 8 % |
| | sentences | 392 | 38 | 39 % | 65 % | 37 % |
| | contexts | 297 | 36 | 45 % | 67 % | 46 %∗ |

**Table 1.** From our publication [14]: sum of false-positives and false-negatives and averages for other measures over all SemSearch, Wikipedia list, and TREC queries. ∗ and † denote a p-value of $< 0.02$ and $< 0.003$, respectively, for the two-tailed t-test compared to the figures for sentences.

To evaluate the potential of semantic full-text search, we also computed and compared ranking related measures. In a ranked list of results, *Precision at K* (P@K) is the precision within the top $K$ results. *R-precision* is the precision within the top $R$ results, where $R$ is the number of relevant results according to the ground truth. The best system [28] at the TREC Entity Track in 2009 achieved an average P@10 of 45% and R-precision of 55%. Our system achieved an average P@10 of 58% and R-precision of 62%, which is a considerable improvement given the small range of results for previous systems.

[14]For TREC, the number of false-negatives actually decreases. This is due to how our parser preprocesses Wikipedia lists. It appends each list item to the preceding sentence allowing contexts to cross sentence boundaries. See our publication [14].

However, these results are not directly comparable for several reasons. First, the TREC system only used a text corpus, ClueWeb'09 [34] Category B, which, however, is bigger (50 million web pages) than our corpus. Second, we constructed our queries manually (albeit not tuning them to the result). This was permitted for this benchmark, albeit submissions that used automatic approaches yielded better results. Still, we believe this makes the problem easier for our system. This also motivates automatically translating into structured queries, which we address for knowledge base search in Section 3.4. Last, the ground truth was approximated via pooling results from the participants. This may put systems that are evaluated later on the same ground truth at a disadvantage [74].

A detailed error analysis on the TREC Entity Track questions revealed that most errors are caused by an incomplete ground truth (55% of false-positives) or errors in third party components (33% of false-positives and 63% of false-negatives): the knowledge base, the constituent parser used for context decomposition, or our entity recognition. If we assume all of these errors can be corrected, our system achieves an average F1 score of 86%, P@10 of 94%, and R-precision of 92%. This motivates work on third party components for correcting these errors. If we only assume a fixed ground truth (by adding missing entities), our system achieves an average F1 score of 65%, P@10 of 79%, and R-precision of 77%. Together, these results show the high potential of semantic full-text search.

## 3.2 Contextual Sentence Decomposition

Contextual sentence decomposition is the task of decomposing sentences into parts that semantically "belong together". We have already presented the basic idea in the previous section, where we also showed how it makes semantic full-text search more precise. As will become clear, a closely related problem is to extract *(subject) (predicate) (object)* triples from a sentence. Extracting these triples is an established task known as *Open Information Extraction* (OpenIE) [7].

In this section, we describe an approach for contextual sentence decomposition and how to extend it to extract OpenIE triples. Our goal is to extract as many correct triples as possible but also to keep them *minimal* and *informative*. These properties are neglected in previous work, but important in applications like semantic full-text search, where precise facts are essential. In our evaluation, we show that our approach matches the state of the art with respect to correctness but improves upon minimality and informativeness.

*This section summarizes the work published in [19] at ICSC 2013 and [18] at ECIR 2014.*

### 3.2.1 Problem, Related Work, and Contributions

The goal of contextual sentence decomposition is to compute, for a given sentence, all sub-sequences of words in that sentence that semantically "belong together". The sub-sequences are called the *contexts* of the sentence. Consider the following sentence:[15]

(S1) *Ruth Gabriel, daughter of the actress and writer Ana Maria Bueno, was born in San Fernando.*

A correct decomposition yields the following contexts:

(C1) *Ruth Gabriel was born in San Fernando*
(C2) *Ruth Gabriel, daughter of Ana Maria Bueno*
(C3) *actress Ana Maria Bueno*
(C4) *writer Ana Maria Bueno*

If we restrict co-occurrence of entities and words to these contexts, as we did with semantic full-text search (see Section 3.1), matches become more precise. For example, if we search for co-occurrences of a person with the word *writer* (with the intent of finding writers) these only match in C4 for Ana Maria Bueno. Her daughter Ruth Gabriel no longer co-occurs with the word *writer*.

[15] The example is taken from our publication [19].

A highly related problem is that of extracting triples from a sentence. For example, from the sentence above the following triples can be derived:

(T1) *(Ruth Gabriel) (was born in) (San Fernando)*
(T2) *(Ruth Gabriel) (is) (daughter of Ana Maria Bueno)*
(T3) *(Ana Maria Bueno) (is) (actress)*
(T4) *(Ana Maria Bueno) (is) (writer)*

Extracting such triples from a sentence is known as Open Information Extraction (OpenIE) [7]. As can be seen, the triples T1-T4 are already close to the contexts C1-C4. What is missing is an assignment of each word to subject, predicate, or object, and inserting the implicit relation *is* in T3 and T4.

In this section, we first describe how to perform contextual sentence decomposition and then how to extend it to an OpenIE system. We have already shown in the previous section that our approach for decomposition can considerably improve results for semantic full-text search. By extending our approach to an OpenIE system, we can directly compare it to other approaches from the literature. This allows an intrinsic evaluation of the technique based on extracted triples in addition to the extrinsic evaluation in the previous section.

Note that mapping subject, predicate, and object of an OpenIE triple to their corresponding elements in a knowledge base is not considered part of the problem. For example, in T1, the subject *Ruth Gabriel*, the object *San Fernando*, and the predicate *was born in* could be mapped to their identifiers in Freebase: *m.02z4mfm*, *m.0r0wy*, and *people.person.place_of_birth*, respectively. This distinguishes OpenIE from *relation extraction*, where only triples of a given knowledge base relation, like *place of birth*, should be extracted, but with many possible variations of mentioning it. An overview of relation extraction and OpenIE can be found in our survey [16, Section 3.6]. Both problems are examples of *information extraction* [75].

OpenIE triples can be used in, for example, knowledge base construction (after linking to canonical entities and relations) or simple semantic querying, for example, to find all subjects matching *(?x) (was born in) (San Fernando)*.[16]

Early OpenIE systems, e.g., TextRunner [87], ReVerb [41], and KnowItAll [40], use shallow natural language processing techniques, for example, part-of-speech tagging or chunking. Using this shallow syntactic information they either apply hand-crafted or automatically learned rules to identify triples. More recent systems, like OLLIE [58], ClausIE [37], and our system, use a deeper linguistic analysis in the form of a syntactic parse tree of the sentence. The parse tree expresses the grammatical structure of the sentence, which clearly identifies nested phrases and clauses. This allows more powerful rules that also work

[16] A demo of such a system is available at: `http://openie.allenai.org/`.

in complicated sentences, e.g., with nested relative clauses. OLLIE uses automatically learned rules on this parse tree. ClausIE and our system use manually defined rules. In contrast to ClausIE, our rules were developed with the additional goal of minimality (see below) and a direct application for semantic full-text search. Our publications [19, 18] and semantic search survey give more details on related approaches as well as related natural language processing techniques like pos-tagging, chunking, and parsing.

OpenIE systems are usually compared based on accuracy and number of extracted triples. Accuracy refers to the percentage of triples that are correct, i.e., that express a meaningful fact which is also expressed in the original sentence. In addition to this, we also consider *minimality* to be essential. Consider the following triple:

*(Ruth Gabriel) (is) (daughter of the actress and writer Ana Maria Bueno)*

This triple is correct, but it also contains two other facts, namely, that Ana Maria Bueno is an actress and a writer. Hence the triple is not minimal. This is problematic for applications like semantic full-text search, which assume that co-occurrences are "semantic". In our evaluation, we consider minimality in addition to the standard measures of accuracy and the number of extractions. This has not been addressed or evaluated in previous work.

Besides correctness and minimality, another important aspect of extracted triples is that of *informativeness*. Consider the sentence[17]

(S2) *The ICRW is a non-profit organization headquartered in Washington.*

and the extracted triples:

(U1) *(The ICRW) (is) (a non-profit organization)*
(U2) *(a non-profit organization) (is headquartered in) (Washington)*

Both triples are correct and minimal, but triple U2 is, by itself, not informative. The information that it is the ICRW that is headquartered in Washington is not explicit and cannot be found with a search in these triples (without the information that the object of U1 and subject of U2 refer to the same organization). Therefore, we propose to integrate a set of simple inference rules into the extraction process to increase informativeness. An informative triple that should be extracted instead of U2 is U3: *(The ICRW) (is headquartered in) (Washington).*

Informativeness has previously been addressed in [41] but only as part of correctness (uninformative triples were labeled as incorrect) and with a different definition, focusing on the predicate part of triples (is the predicate by itself informative). We explicitly consider informativeness of whole triples and in relation to the originating sentence.

[17] The example is taken from our publication [18].

To summarize, our main contributions are the following:

- An approach for contextual sentence decomposition that is based on a grammatical parse of a sentence. Compared to similar approaches, our output is tailored to an application in semantic search.
- An extension of contextual sentence decomposition to extract OpenIE triples. In an evaluation, we show that extracted triples match the state of the art with respect to correctness, but are better than existing systems with respect to minimality.
- A way to increase the informativeness of extracted triples by using inference rules during the extraction process. Our evaluation shows that a few simple inference rules can mitigate many uninformative triples.

### 3.2.2 Approach

We first describe how to perform contextual sentence decomposition followed by how to derive triples. This is described in more detail in [19]. We then outline how to improve informativeness of triples, with details available in [18].

Contextual sentence decomposition is performed in two steps. In the *sentence constituent identification* phase (SCI), we identify the basic "building blocks" of contexts and arrange them in a tree. *Sentence constituent recombination* (SCR) combines the identified constituents to form contexts.

Figure 2 shows an SCI tree for our example sentence:

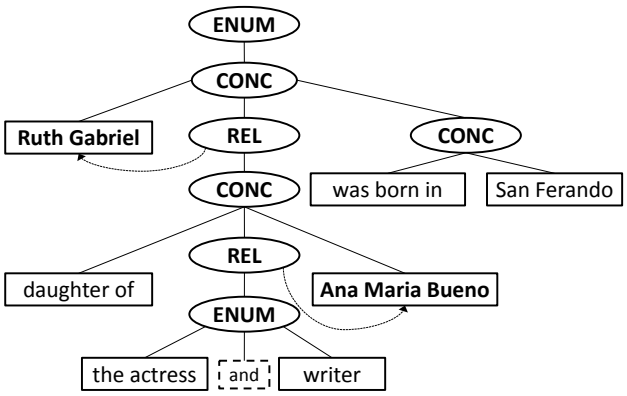


**Figure 2.** From our publication [19]: The SCI tree for our example sentence. The head of each relative clause is printed in bold, filler words in striped rectangles.

(S1) *Ruth Gabriel, daughter of the actress and writer Ana Maria Bueno, was born in San Fernando.*

The goal of SCI is to compute such a tree for a given sentence. We distinguish between three different types of internal nodes. ENUM nodes identify enumerations, where child nodes belong to different contexts. In our example, *the actress* and *writer* are two separate facts that describe Ana Maria Bueno, but have nothing to do with the rest of the sentence. REL nodes mark relative clauses, which form separate contexts with their optional head[18]. In our example, the nominal modifier (which we consider a type of relative clause) starting with *daughter of...* is connected to its head, *Ruth Gabriel*, whom it describes. Finally, CONC nodes group child nodes that belong to the same context. As is illustrated in Figure 2, words of a sentence (terminals) are only contained in leaf nodes. Nodes can be nested recursively and arbitrarily deep. In practice, deep nestings are rare, since the sentence also becomes hard to read for humans. Note how the SCI tree already expresses a lot of the semantic structure of the sentence.

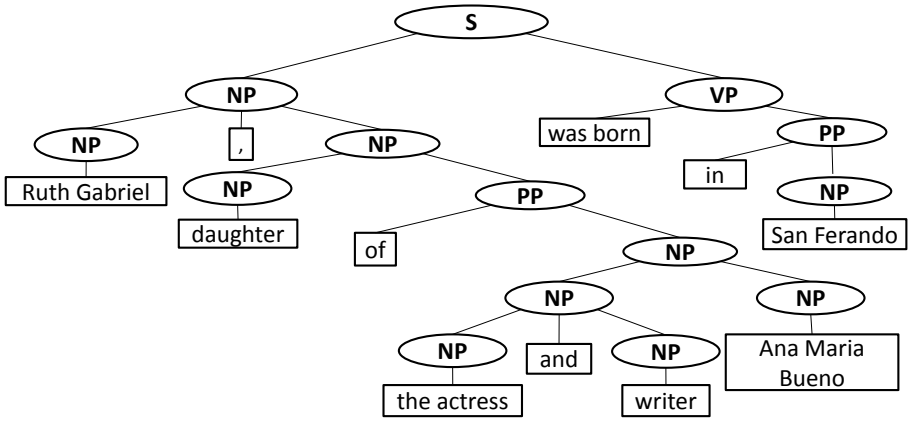


**Figure 3.** From our publication [19]: The constituent parse tree for our example sentence. It arranges noun phrases (NP), verb phrases (VP) and prepositional phrases (PP), according to the syntax of the sentence. For the sake of readability, the parse tree has been simplified.

A good starting point to derive the SCI tree is the constituent parse tree of the sentence. Figure 3 shows the parse tree for our example sentence. It can be seen that the tree hierarchically groups important constituents for the SCI tree. Identifying these constituents based on a more shallow analysis, like part-of-speech tags, is difficult.

We carefully designed a small set of 14 (prioritized) rules to transform a parse tree into an SCI tree. An applied rule in our example is: "in a sequence consisting of two NPs, split by a comma, mark the second NP as REL and the first NP as its head" [19]. This identifies the relative clause *daughter of the actress and writer Ana Maria Bueno* and its head, *Ruth Gabriel*. Another applied rule is: "mark a node as ENUM for which children all have the same type (e.g., NP)" [19]. This identifies the enumeration in *the actress and*

[18]In our publication, we also describe SUB nodes, which are REL nodes without the optional head.

*writer*. The complete list of rules can be found in our publication [19]. After applying all of the rules, we get the tree shown in Figure 2.

Given the SCI tree, computing contexts is straightforward. We first take out subtrees labeled REL, and change the root of this new tree to CONC. If the relative clause had a head, we attach it as first left child to the new tree. Then we recursively compute contexts for each tree, which now only contain leafs, ENUM, or CONC nodes. The context of a leaf consists exactly of the words contained in it. The contexts of an ENUM node is computed as the union of the sets of all child node contexts. The contexts of a CONC node is computed as the cross-product of the sets of its child node contexts. This gives us the desired contexts C1 to C4 shown in the beginning.

To transform contexts into triples, we apply a set of (relatively) simple heuristics. In each context, we identify the first explicit verb phrase and surrounding adverbs or prepositions to be the predicate. The words before the predicate belong to the subject and the words after it to the object. For example, in the context *Ruth Gabriel was born in San Fernando* we identify *was born in* as predicate and can derive the subject and object accordingly. Our heuristics also insert implicit verbs, for example, we use the verb *is* between the head and its REL attachment (if it doesn't begin with a verb). This allows deriving the triple *(Ana Maria Bueno) (is) (writer)* from the context *writer Ana Maria Bueno*.

In a final step, we improve informativeness of extracted triples using inference rules. For each triple, we first classify the predicate into one of five semantic relation classes.

| | | |
|---|---|---|
| $OTHER(A', B)$ | $\leftarrow$ | $OTHER(A, B) \wedge SYN(A, A')$ |
| $OTHER(A', B)$ | $\leftarrow$ | $OTHER(A, B) \wedge SYN(A', A)$ |
| $OTHER(A, B')$ | $\leftarrow$ | $OTHER(A, B) \wedge SYN(B, B')$ |
| $OTHER(A, B')$ | $\leftarrow$ | $OTHER(A, B) \wedge SYN(B', B)$ |
| $IN(A, C)$ | $\leftarrow$ | $IN(A, B) \wedge PART\text{-}OF(B, C)$ |
| $IN(A, C)$ | $\leftarrow$ | $IN(A, B) \wedge IS\text{-}A(B, C)$ |
| $OTHER(A, C)$ | $\leftarrow$ | $IS\text{-}A(A, B) \wedge OTHER(B, C)$ |

**Table 2.** From our publication [18]: Inference rules for new triples.

For example, IS-A expresses hyponymy, the relation between a specific instance and its more generic term, in triple U1: *(The ICRW) (is a) (non-profit organization)*. The predicate *is headquartered in* expresses a location placement, IN, in triple U2: *(a non-profit organization) (is headquartered in) (Washington)*. Other relation classes are SYN (synonymy), PART-OF (meronomy, for "part-whole" relationships), and OTHER for all remaining relations. Based on the semantic relations, we apply a set of seven inference rules shown in Table 2 to derive new triples.

In the example, the rule "if A IS-A B and B IN C → A IN C"[19] matches. We, therefore, conclude *(The ICRW) (is headquartered in) (Washington)*. Crucially, this inference step can only be performed during triple extraction for a given sentence, when involved subjects and objects refer to the same words in a sentence. In the example, it must be clear that *a non-profit organization* in triples U1 and U2 actually refer to the same real-world entity.

We also remove existing triples that we consider uninformative depending on how they were used to derive new triples. For example, we remove triple U2 in favor of the inferred triple above. While our rules are manually selected and simplistic, they improve informativeness of extracted triples as our evaluation shows.

### 3.2.3 Experiments and Results

We have already described an extrinsic evaluation of contextual sentence decomposition in Section 3.1, which showed that contexts make semantic full-text search more precise. Here, we describe an evaluation based on extracted OpenIE triples from [19]. Our system is called CSD-IE in the following.

For evaluation, we used two datasets from [37]: 200 random sentences from the English Wikipedia and 200 random sentences from the New York Times. We compared our system against three state-of-the-art OpenIE systems: ReVerb [41], using learned rules on part-of-speech tags, OLLIE [58], using learned rules on a parse tree, and ClausIE [37], using manually defined rules on a parse tree. In the first experiment, which we describe in [19], we didn't apply our inference technique to improve informativeness. We evaluated this separately in a second experiment.

For the first experiment, we extracted triples with each of the four systems and manually assigned two labels for each triple: one for correctness (yes or no) and one for minimality (yes or no). From these labels, we computed the following accumulated measures for each system as defined in [19]:

| | |
|---|---|
| *precision wrt accuracy (prec-a)*: | the percentage of triples labeled as correct |
| *precision wrt minimality (prec-m)*: | the percentage of correct triples labeled as minimal |
| *coverage*: | the percentage of words of the sentence that occur in at least one extracted triple for that system |
| *average triple length in words*: | average length of extracted triples for that system in words (ignoring special characters) |

[19] More formally: $\mathit{IS\text{-}A}(A, B) \wedge \mathit{IN}(B, C) \rightarrow \mathit{IN}(A, C)$

| | ReVerb | OLLIE | ClausIE | CSD-IE |
|---|---|---|---|---|
| #facts | 249 | 408 | 610 | 677 |
| #facts correct | 188 | 230 | 421 | 474 |
| prec-a | 75.5 % | 56.4 % | 69.0 % | 70.0 % |
| prec-m | 87.2 % | 80.4 % | 57.0 % | 76.8 % |
| coverage | 47.2 % | 62.7 % | 95.4 % | 97.5 % |
| triple length | 7.3 | 9.7 | 11.0 | 8.4 |

**Table 3.** From our publication [19]: Results of our quality evaluation on the Wikipedia dataset.

Table 3 shows the results for the Wikipedia dataset. The results on the New York Times dataset are very similar and gave no additional insights. For both datasets, the results for previous systems closely agree with those reported in [37], confirming our labeling.

In comparison to the other systems, CSD-IE extracts the largest number of correct facts (*#facts correct*). It also provides the highest coverage (*coverage*) and largest number of extracted facts overall (*#facts*). ReVerb produces a higher percentage of minimal and correct triples, however, for a considerably lower number of extracted facts. In an application for semantic search this can be detrimental to search quality. In particular, this is likely to cause missing extractions, which in turn lead to missing results (lower recall).

Compared to ClausIE, the most similar system, CSD-IE achieves similar precision with respect to accuracy (*prec-a*) and coverage. The triples extracted by CSD-IE are shorter on average (*triple length*). Furthermore, the precision wrt minimality (*prec-m*) is 20% higher. This is a considerable improvement that can be attributed to the fact that our rules were explicitly tailored towards minimality.

An investigation of errors for CSD-IE on both datasets reveals that most of the inaccurate extractions are caused by mistakes in the parse trees. This is also what we observed in the analysis of our results in Section 3.1. Small mistakes, like attaching a subtree to the wrong parent, cause wrong extractions for all contexts and triples involving that parent-child relationship. Current state-of-the-art parsers still make such mistakes for about 8% of parent-child relationships.[20] More details on parsing and an overview on the current state of the art are available in our survey [16, Section 3.3].

We believe there are two worthwhile directions to handle wrong extractions caused by the parser. First, we could compute confidence scores for triples. Some parsers can provide an estimate of their confidence for each subtree being correct. Wrong subtrees should get assigned a low confidence score and triples from such subtrees could be assigned low

[20]Results in [52, Table 1] show that the best performing parser attaches 92% of words correctly. This is for a dependency tree, which is similar to a constituent tree with respect to parent-child relationships.

scores. Second, it would be interesting to extract triples without the help of a parse tree. This could be achieved via learning a task-specific extractor from scratch, i.e., without a pipeline of cascading tools that propagate errors. This has been shown to work well for other natural language processing tasks, especially sentence parsing [35].

In a second experiment, which we describe in [18], we investigated how our inference rules improve the informativeness of extracted triples. We manually assigned two labels for triples extracted from the 200 Wikipedia sentences from above: one for correctness (yes or no) and one for informativeness (yes or no). A correct triple was considered informative if there is no extraction that is more precise, according to the sentence it was extracted from. We gave an example for this above.

| | #facts | #facts corr | #facts corr-inf | prec corr | prec corr-inf |
|---|---|---|---|---|---|
| No Inference | 649 | 429 | 385 | 66% | 90% |
| Inference | 762 | 484 | 444 | 64% | 92% |

**Table 4.** From our publication [18]: Results for our quality evaluation with inference (top row) and without inference (bottom row) over the labels correct (corr) and informative (inf). *prec corr* refers to the percentage of all triples labeled correct, *prec corr-inf* to the percentage of correct triples labeled informative. The experiments used a faster parser, hence the results differ slightly from those in Table 3.

The results in Table 4 show that about 90% of all correctly extracted triples are informative. After applying our inference rules this increases to 92%. In addition recall increases. The number of correctly extracted triples increases by 13% (#facts corr) and the number of triples both correct and informative by 15% (#facts corr-inf). Since a few incorrect triples are inferred the overall percentage of correct triples (prec corr) drops by 2%, but at a higher recall.

An error evaluation shows that many incorrectly inferred triples are caused by wrong parses or a wrong mapping of predicates to their semantic class. Eliminating these should obviate the small negative effect on precision. A more sophisticated mapping to semantic classes could be achieved by utilizing "relational patterns" as, e.g., described in [65]. To further increase informativeness, it may also be interesting to automatically learn inference rules, as, e.g., described in [56, 84, 63]

Overall, the results show that CSD-IE achieves good precision and high recall, while providing very good coverage and minimality. We attribute this to the fact that our original motivation for this problem is an application in a semantic search engine. Hence our rules were developed and tailored with minimality in mind. The results also show that already a few simple inference rules can improve informativeness of extracted triples. Especially minimality and informativeness are important in applications like semantic search where a precise representation of facts is beneficial.

## 3.3 Relevance Scores for Triples

In the previous section, we have described a technique that makes semantic search in full text more precise. In this section, we show how to improve the results of structured queries on a knowledge base. In particular, we address how to rank results of queries that return a list of entities, such as *american actors*. For the queries we consider, the information required for a good ranking is not contained in the knowledge base. Therefore, we describe a variety of algorithms to compute relevance scores from a text corpus. Since this is a novel problem, we also describe a benchmark that we designed. On this benchmark, our methods achieve an agreement of about 80% with the ground truth and outperform existing methods from the literature. The benchmark and our code are publicly available via `http://ad.informatik.uni-freiburg.de/publikationen`.

*This section summarizes the work published in [15] at SIGIR 2015.*

### 3.3.1 Problem, Related Work, and Contributions

Knowledge bases are queried using a structured query language called SPARQL. A structured query has precisely defined semantics, and the corresponding result set is well-defined. For example, here is a query for *american actors* in SPARQL:[21]

```
select ?entity where {
  ?entity  has-nationality  American .
  ?entity  has-profession Actor .
}
```

This returns 64,757 matches on Freebase in no particular order.[22] Clearly, when presenting this huge result set to a user, a ranking is desirable. A straightforward ranking would be by some form of popularity. This can be measured, e.g., by the number of occurrences in a reference text corpus, which leads to the following top-5 results:

*George Bush*, *Hillary Clinton*, *Tim Burton*, *Lady Gaga*, *Johnny Depp*

All persons in this list are actors in the sense that they had a role in some movie. For example, George Bush appeared in the documentary Capitalism: A Love Story and Tim Burton had various cameo roles. However, in this list, only Johnny Depp is best known for being an actor. Hillary Clinton and George Bush are better known for being politicians, Lady Gaga as a musician, and Tim Burton as a film director. Consequently, one would expect Johnny Depp to be ranked before all others.

[21] This example, including results and relevance scores, is taken from our publication [15].

[22] An order can be specified explicitly, e.g., with an ORDER BY clause, but this doesn't solve the problem we consider here.

The set of *american actors* is huge, but an ordering also makes sense for smaller result sets. Consider the query for professions of Ronald Reagan in Freebase:

*Actor*, *Lifeguard*, *Politician*, *Radio Personality*, *Soldier*, *Spokesperson*,

All of them are correct (indeed, Reagan worked as a lifeguard in his youth), but his main professions are certainly *Politician* and *Actor*. In this case, ranking by the popularity of a profession makes no sense at all.

Queries as above are also typical queries for semantic full-text search (see Section 3.1). However, such queries are likely to appear for any kind of semantic search that utilizes a knowledge base. Hence, we consider this an interesting and relevant problem on its own that warrants a separate study.

In this section, we address the problem behind the examples above, which we defined in [15] as follows:

> **Definition:** Given a type-like relation from a knowledge base, for each triple from that relation compute a score from $[0, 1]$ that measures the degree to which the subject belongs to the respective type (expressed by the predicate and object). In the remainder, we often refer to these scores simply as *triple scores*.

Another way to describe the problem for the score of the profession *Actor* of Johnny Depp is: "how surprised would we be to see Actor in a list of professions of Johnny Depp" [15].

The desired scores for some of the entities in the queries above might look as follows:

| | | | |
|---|---|---|---|
| *Tim Burton* | *has-profession* | *Actor* | 0.3 |
| *Tim Burton* | *has-profession* | *Director* | 1.0 |
| *Johnny Depp* | *has-profession* | *Actor* | 1.0 |
| *Ronald Reagan* | *has-profession* | *Actor* | 0.6 |
| *Ronald Reagan* | *has-profession* | *Lifeguard* | 0.1 |

The actual score of a triple is ill-defined in our definition above. This is similar to the notion of relevance in information retrieval. There, a document is assigned a relevance score (in the case of graded relevance) given a query. It is common practice to determine document relevance via judgments from different people, simply because the actual relevance score of a document can be subjective. In our experiments (see below), we also collect feedback from multiple judges and show that there is a strong consensus on our relevance scores.

Once relevance scores are computed, it becomes straightforward to use these scores to rank the results of our example queries. In the first case (*american actors*), a ranking can

be derived by combining the scores with a popularity for each entity. In the second case ("professions of a single person"), the scores directly infer a ranking.

In this work, we consider scoring triples from *type-like relations*, such as *profession* or *nationality* (we use both in our evaluation). These present the biggest challenge in terms of ranking. For functional relations like *date of birth* or *height*, no ranking is needed or trivially achieved. For example, for *height*, simply ranking by this value will often be sufficient. Triples from non-functional relations between two concrete entities like *invested in* (between two organizations) or *featured in song* (between a musician and a song she featured in) are often better ranked by a single scalar from the knowledge base, like the value of the investment or the length of the song feature. Finding such a single scalar is difficult for type-like relations. For example, for the *profession* relation, the (computed) scalar depends on the actual profession: For a musician, it may be the number of records sold, while for an actor the appearances in high-grossing movies are more relevant.

Our motivation is ranking results of entity queries like *american actors*. Here, we focus on computing scores for individual triples in the first place. This has several reasons. First, scores as above are often all that is needed to obtain a good ranking, for example, for the query for professions of a single person. Second, our scores can serve as a crucial ingredient for approaches that rank results of entities queries. We discuss some of these approaches below. Finally, as our evaluation shows, computing good relevance scores is difficult and requires tailored approaches. These warrant a separate study.

To the best of our knowledge this is the first work addressing how to compute this kind of relevance scores for triples. There is some work on estimating scores for the correctness or accuracy (is the numeric value, e.g., a population count, off by some margin) of triples. Correctness scores can come from the system that extracted the triple, e.g., a probability provided by a machine learning classifier [58, 62]. They can also be inferred, for example, via a witness count as indicated by the number of times a given triple was extracted or found. Correctness scores are important, e.g., in knowledge fusion [38] or knowledge base construction [68, 63], where the task is to construct a consistent knowledge base from extractions from different sources. In this case, correctness scores can help resolving conflicting statements, e.g., if extracted triples disagree on the place of birth of a person. Correctness and accuracy scores are different to our relevance scores, where we assume non-conflicting facts and estimate a "degree of belonging". A beneficial side effect of our appoches is that incorrect triples are likely to get a low score.

A related line of work is that of detecting fine-grained types of entities. There, the motivation is, given a sentence and the identified mention of an entity, infer its types from a given type system. For example, from the input *Ada wrote the first computer program* and the identified named entity (underlined), it could be inferred that Ada belongs to the

*person* and *programmer* types of Wordnet [61]. The given input entity is not linked to a knowledge base (or does have to exist in one). Recent approaches focus on providing very fine-grained types. In [64], the authors match learned text patterns that are associated with a type and devise a probabilistic model and integer linear program to avoid type inconsistencies (e.g., a *person* cannot also have type *organization*). In [36], the authors design a huge set of patterns and rules based on, e.g., the mention's text (the mention *University of Freiburg* contains the type *university*) or verb phrases. The system is evaluated with 16k different types. In contrast to triple scores, approaches for type detection assign types in a binary fashion. Assigning Ronald Reagan the types *Lifeguard* and *Politician* to the same degree is both, correct and desired. Furthermore, approaches are designed to identify types from a taxonomy. While the *profession* relation describes part of a taxonomy, this is not the case for other type-like relations, e.g., *nationality*, which we also evaluate on. Our approaches are designed to work with all type-like relations.

For ranking results of queries on a knowledge base, a typical approach is to adapt and apply techniques from ranking in information retrieval. The adapted techniques usually rank entities based on existing scores that are assumed to be given or from ranking signals (machine learning features) that are computed from the knowledge base and query. For example, in [39], the authors investigate how to define language models for a structured SPARQL query and result graphs. The language models incorporate witness counts to estimate the confidence and importance of a triple. These are assumed to be given. In [31], the authors compare learning to rank methods for keyword queries on a knowledge base. They devise a set of 26 features that incorporate, for example, the type of an entity and the similarity between an entity and the query. Both of these example approaches could benefit from using our triples scores in addition or instead of the scores or type information they already use. More approaches for ranking results of entity queries can be found in the ranking section of our survey [16, Section 5.1].

As part of our work on this problem we have created and published a benchmark which has lead to a public challenge on triple scoring at WSDM 2017 [48] with good participation (21 teams). Most of the approaches (including the winning approach) relied on finding witnesses for each type from text and applying supervised learning methods. This is similar to one of the approaches we suggest below.

There are several challenges we face for computing triple scores. First, we cannot compute scores from the knowledge base alone. We found that the required information, especially for less popular entities, is not available reliably, even in a large knowledge base like Freebase. Therefore, we focus on computing relevance scores from text. Second, we cannot rely on purely supervised learning. There are more than 3,552 different professions in Freebase and relevance is expressed differently for each. Hence, a different model and therefore labeled examples are required for each profession. This rules out manually la-

beling examples, even with crowdsourcing. Finally, since this is a novel problem, we must also establish a ground truth that allows a realistic and reliable comparison of approaches.

In summary, our main contributions are:

- The novel and interesting research problem of triples scores for type-like relations. The scores are an essential component for properly ranking many popular kinds of entity queries. Such queries occur, for example, as part of semantic full-text search.
- A benchmark to evaluate triple scores for the *profession* and *nationality* relations. The benchmark consists of more than 14 thousand relevance judgments that were obtained via crowdsourcing. The judgments confirm that there is a broad general consensus on the problem definition of triple scores.
- A variety of approaches to computing triple scores from a text corpus. On our benchmark, our best methods perform significantly better than non-trivial baselines and achieve an agreement of about 80% with the ground truth.

### 3.3.2 Approach

We describe our approach that computes triple scores from a text corpus. Initially, we also experimented with using a knowledge base, either as a supplement to a text corpus or by itself. The scores we are trying to compute are certainly not explicitly expressed as facts but may be implicit in other facts. For example, if a person has acted in a lot of high-grossing movies, she is likely to be well-known as an actress, and that triple should receive a high score. However, experiments showed that this knowledge is consistently less available, especially for less popular entities.[23] Therefore, we compute triple scores from a large text corpus.

In the text corpus, we try to find "witnesses" for each triple. Optionally, each witness has an associated significance, which we also compute. The higher the count and significance of witnesses for a triple, the larger the score. For example, consider the following triples for two of *Johnny Depp*'s professions:

| | | |
|---|---|---|
| *Johnny Depp* | *has-profession* | *Actor* |
| *Johnny Depp* | *has-profession* | *Musician* |

We assume that, because he is more of an *Actor* than a *Musician*, he is more likely to be mentioned along words like: *actor*, *film*, *cast* than *album*, *band*, or *singer*. The word *performed* can be associated with *Actor* as well as *Musician* and has less significance for

[23] For example, in Freebase, out of 612 thousand persons with more than one profession, 400 thousand have less than ten and 243 thousand have less than six facts (besides type and profession information).

determining these professions. By counting the number of mentions of these words (and their significance) along *Johnny Depp* in the text corpus, we can compute a score for each profession triple. Ideally, this gives a much higher score for *Actor* than for *Musician*.

This approach involves three steps. First, we need to learn *indicator words* (like *actor*, *film*, *cast* above) for each profession. Each word can be associated with a weight, depending on how significant it is for the profession. Next, for a given entity, we need to find co-occurrences with these words from which we derive an *intermediate score*. The scores should be comparable across entities. Therefore, in a final step, we *normalize* the intermediate scores to a desired output range, for example, the interval $[0, 1]$.

We designed multiple variants of how to learn indicator words, how to use them to derive intermediate scores, and how to normalize to the final output range. We shortly summarize these in the following. Note that we use *profession* as an example relation, but our approaches work for any type-like relation.

Our first step to learning indicator words is to derive positive and negative training examples for each profession. This is done in a weakly supervised fashion using the following criteria: For a given profession, the positive training examples are persons who *only* have that profession (according to the knowledge base). The intuition is that these triples naturally have the highest score assigned. Correspondingly, negative training examples are persons who don't have the given profession at all. Clearly, these should have the lowest score. For example, Marlon Brando is a positive example for the profession *Actor*, because this is his only profession according to Freebase. Alan Turing would be a negative example, because he is a *Computer Scientist*, *Mathematician*, and more, but not an *Actor*.

Next, we represent each person with a *virtual document*. This document consists of all the words the person co-occurs with. To identify semantically co-occurring words we use contextual sentence decomposition as described in Section 3.2. The virtual document of a person now consists of all (of the words) of the contexts she appears in. In our experiment, this gave better results than using whole sentences.

We now have a set of positive and negative examples (virtual documents of persons) for each profession. From here, we consider multiple variants of learning indicator words and deriving intermediate scores. We shortly mention the basic ideas, in our experiments we also considered many slight variations:

- *Binary classifier*: Learn a classifier (logistic regression) that makes a binary decision for each profession. Each person is represented by the bag-of-words of her virtual document with words weighted by tf-idf. This way, the classifier will learn a weight for each word. The intermediate score for a profession is the binary decision of the classifier whether the profession is primary or not.

- *Count profession words*: First, identify indicator words. A simple variant is to use manually chosen prefixes of the profession, such as *act* for *Actor*. This as a baseline in our experiments. A more complex variant is to compute and sum tf-idf scores for words of the positive examples, sort by that score, and assign a weight of $1/r$ for word at rank $r$. This gives weighted words for each profession. To get intermediate scores, for each profession, count the prefix matches or sum up the word weights in the virtual document of a person.
- *Generative model*: Identify weighted indicator words for each profession as with the counting-based approach. Then apply a model, which, intuitively, generates the text in the virtual document of a person. Maximizing the likelihood of the person's text gives a probability distribution over her professions as intermediate scores.

The approaches output different types of intermediate scores for each triple: word counts, weighted sums, binary judgments (logistic regression), or probabilities. In the final step, we normalize these for each person by mapping to the desired output range. In our experiments, we map to the integers $[0, 7]$ in order to facilitate comparison with our crowdsourcing results. We either map to these scores on a linear scale or on a logarithmic scale, where the next highest score corresponds to twice the intermediate score (sum or probability).

### 3.3.3 Experiments and Results

Since this is a novel problem, we started by creating a new benchmark. To obtain a large number of relevance judgments we used crowdsourcing. Each human judge is given a description of the task. In general, the judges aren't familiar with the problem and a major challenge was describing the task in way that results in consistent and reliable relevance scores. This gave us valuable insight into our problem and its definition, also allowing us to verify whether there is a general consensus on our scores. We first describe our crowdsourcing task, followed by results on the new benchmark.

#### Crowdsourcing Benchmark

We experimented with several different crowdsourcing tasks, refining the task definition and description along the way. Figure 4 shows the final task that worked well.


**Figure 4.** Adapted from our publication [15]: Excerpt of the crowdsourcing task for Ludwig van Beethoven. His professions must be classified into *primary* or *secondary* by dragging each profession into the respective box. The Wikipedia link to his page can be used to find additional information (important for lesser known persons). In addition, example classifications and further instructions were provided (not shown here).

Judges must classify all of the professions of a single person into either *primary* or *secondary*. This worked better than asking for the label of a single profession of a person (without showing the other professions). Judges then often simply labeled the first mentioned profession of a person in the Wikipedia article as primary. We used this strategy as one of the (simple) baselines in our experiments.

Each person is judged seven times by different judges. The number of primary judgments for each profession of a person is aggregated to give a score from 0 to 7. For the profession relation, we ran the above task for a random selection of 298 persons (1225 triples) which gave us 8575 judgments. The random selection takes into account the popularity (i.e. the number of mentions in our text corpus) of persons, providing a fair selection across

all popularity levels. We also evaluated the nationality relation, for which we selected 77 different people (162 triples) resulting in 1134 judgments.

To verify inter-annotator agreement we performed a *control* run of the above experiment (with different judges) on one-third of the persons selected for the profession relation. We also created *random* judgments choosing primary or secondary with equal probability.

Figure 5 shows a histogram of the results. As can be seen, judgments are far from random and the control run provided very similar results.. This, together with a Fleiss' Kappa [44] of 0.47, shows that there is broad general consensus on the problem.

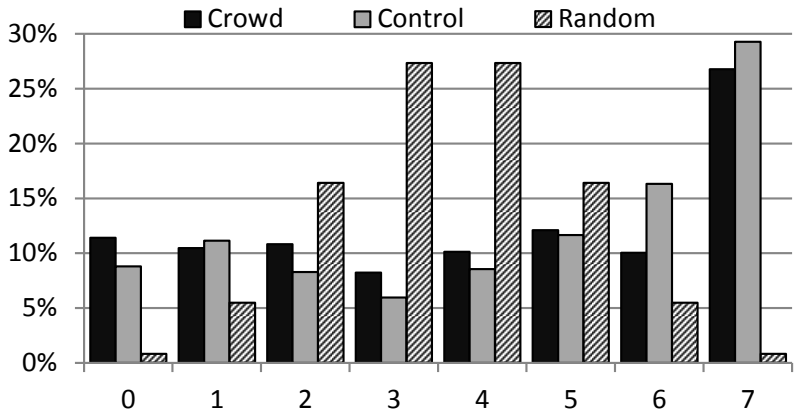


**Figure 5.** From our publication [15]: Histogram of score distribution of our crowdsourcing task, the control run, and expected results for randomly (with $p = 0.5$) guessing judges (rounded).

### Main Results

We evaluated our approaches using the above benchmark. As a text corpus, we used the English Wikipedia where we recognized entity mentions as described in Section 3.1. As a knowledge base, we used Freebase.

We performed experiments for the profession and the nationality relation. Experiments on the nationality relation gave no major additional insights, hence we only describe the main results for the profession relation here.

We compared our variants against several baselines. The *first* baseline selects the first literal occurrence of a profession in the person's textual description of Freebase to be primary. The *prefixes* baseline applies the counting approach from above using prefixes that were manually derived from the profession names (e.g., *act* for *Actor*). The *Labeled Latent Dirichlet Allocation (LLDA)* [71] baseline is a topic model that can be learned in a supervised fashion using our weakly-supervised training data. Given the text associated with a person, it estimates a probability distribution over topics (e.g., professions).

All scores were mapped to the range 0 to 7, which is the range we get from our crowdsourcing experiments. To map our intermediate scores to this range, we used the mapping that gave best results: linear mapping for count-based approaches and logarithmic mapping for approaches that output probabilities. As main evaluation measures, we computed accuracy-$\delta$, which measures the percentage of triples that deviate by at most $\delta$ from the ground truth score, and average score deviation, the average over all deviations. We also computed rank based measures: Kendalls' Tau, footrule distance, and nDCG. These correlated well with accuracy-$\delta$ and we omit them for brevity.

| Method | Accuracy (Acc) | | | Average Score Diff |
|---|---|---|---|---|
| | $\delta = 1$ | $\delta = 2$ | $\delta = 4$ | |
| First | 41 % | **53 %** | 71 % | 2.71 |
| Prefixes | 50 % | **64 %** | 83 % | 2.07 |
| LLDA | 50 % | **68 %** | 89 % | 1.86 |
| Binary Classifier | 47 % | **61 %** | 78 % | 2.09 |
| Weighted Indicators | 57 % | **75 %** | 94 % | 1.61 |
| Generative Model | 57 % | **77 %** | 95 % | 1.61 |
| Combined (GM + Classifier) | 63 % | **80 %** | 96 % | 1.57 |
| Control Judges | 76 % | **94 %** | 99 % | 0.92 |

**Table 5.** Adapted from our publication [15]: Accuracies and average score deviation for the *profession* relation. The top part shows baselines, the middle part our approaches, and the bottom part our control run.

Table 5 summarizes the results for the profession relation. We consider accuracy-2 the most intuitive measure (percentage of triples with difference to ground truth score at most two), hence it is emphasized. Note that accuracy-2 can be optimized by truncating results to range between 2 and 5, which is never worse than predicting more extreme values. However, this goes at the cost of other measures like average score difference [48]. We performed no such truncation and our scores correlate well with average score difference.

The baselines perform well, but our more sophisticated approaches clearly outperform them. Our best approach consists of a combination of the generative model and the binary classifier (use the average of the two predicted scores if the binary classifier predicts score 7). It achieves performance not far from human judges with a gap of 14%. For all approaches we also evaluated many variants, including: using word pairs instead of single words, using stemming, utilizing the knowledge base type hierarchy, other non-linear score mappings, using tf instead of tf-idf etc. None of these improved results.

An evaluation of errors showed two main sources. The first source of errors is word co-occurrences that refer to the wrong entity. For example, Michael Jackson gets a high score as a *Film Director* because he is often mentioned along *directed* or *director*, which however, does not refer to him, but to a person directing one of his shows.[24] Using co-occurrence based on our context decomposition instead of full sentences helps in some, but not all of these cases. The second source of errors are words that occur with a different meaning. For example, John F. Kennedy gets a high score as a *Military Officer*, simply because he had many political actions during his presidency with respect to the military. In some of these cases, a deeper linguistic processing of text as, e.g., in fine-grained entity typing [64, 36] may help. In general, it appears that a deep natural language understanding will be necessary to close the gap to human judges.

[24] This and the following error example are taken from our publication [15].

## 3.4 Question Answering on Knowledge Bases

The previous section was concerned with making knowledge base search more precise. In this section, we address how to make the search more convenient. We present an approach that automatically translates a natural language question into a structured query on a knowledge base. This makes searching the knowledge base easy by allowing the user to freely formulate her questions in natural language, without knowing about a query language or the structure of the underlying data. In addition to providing precise translations, we also consider efficiency. We make sure that questions can be answered in an interactive manner within at most one second. At the time of publication, our approach improved the state of the art in terms of quality and efficiency on two large established benchmarks.

*This section summarizes the work published in [17] at CIKM 2015. In the following, the examples are taken from this paper.*

### 3.4.1 Problem, Related Work, and Contributions

Mapping a search desire to the corresponding structured query on a knowledge base can be difficult, even for an expert. For example, consider answering the (seemingly) simple question *who is the ceo of apple?* on the knowledge base Freebase.[25] The SPARQL query to retrieve the answer is:

```
select ?name where {
  ?0 leadership.role Managing_Director .
  ?0 leadership.company Apple_Inc .
  ?0 leadership.person ?name
}
```

This matches the following triples in the knowledge base:[26]

| | | |
|---|---|---|
| m.0k8z | leadership.role | Managing_Director |
| m.0k8z | leadership.company | Apple_Inc |
| m.0k8z | leadership.person | Tim_Cook |

The triples contain an abstract "leadership" entity (*m.0k8z*) with three relations: *role*, *company*, and *person*. The relations and the leadership entity connect the entities *Managing_Director* and *Apple_Inc* to the answer *Tim_Cook* we are looking for.

[25] This query is from the WebQuestions benchmark [20] which we use in our evaluation. For illustration purposes, we omit namespace prefixes and use readable identifiers in the corresponding SPARQL query.

[26] We have updated the triples with the current CEO. The original Freebase data still mentions Steve Jobs. Freebase is being migrated to Wikidata [82], which is constantly updated with new information.

To construct the query above, a user would need to find the identifiers of the correct entities and relations and connect them in the correct way. This can be facilitated by an interactive construction guided by suggestions like our semantic full-text search prototype provides (see Section 3.1). However, this still passes some of the complexity to the user. It would be far more convenient to ask the question in natural language and automatically get the structured query that produces the answer. This is the problem we address here.

Answering natural language questions is a difficult task. Here, we focus on answering questions from a knowledge base, albeit a very large one. As we show in our evaluation in this section, this is hard enough to warrant a separate study. Our survey [16, Section 4.6, 4.7, 4.8] provides an overview of the state of the art on question answering on text, knowledge bases, and the few works that attempt a combination.

The challenge in answering natural language questions stems from the high variation and ambiguity inherent in natural language. This becomes apparent when looking at two important subproblems of the translation process: identifying knowledge base entities and identifying knowledge base relations that are mentioned in the question. If these subproblems were solved perfectly, the correct query would be trivial to infer in most cases. However, matching entities is difficult because there are many ways of mentioning a specific entity (synonymy) but a single mention might also refer to many different entities (polysemy). For example, the mention of *apple* in the query above refers to 218 entities in Freebase but only one entity, *Apple_Inc*, is actually correct. The same problem holds for matching relations but is even harder. It can involve $n$-ary relations and some of the relations might only be mentioned implicitly. For example, the question above contains no word or synonym of the query's relation names *person*, *company*, or *role*. This problem becomes even harder with the size of the knowledge base. Allowing weak and fuzzy matches drastically increases the search space. On the other hand, only allowing strict and lexical matches misses many correct matches.

Our goal is to answer questions from a large knowledge base like Freebase, which contains about 2.9 billion facts on 44 million entities. To keep the problem manageable, we focus on "structurally simple" queries. These involve two or three entities and either a single binary or $n$-ary ($n > 2$) relation. In the example above, the result entity *Tim_Cook* is connected to the entities *Apple_Inc* and *Managing_Director* via an $n$-ary leadership relation. In SPARQL, this $n$-ary relation is represented using several binary relations, and the special leadership entity (*m.0k8z*) also referred to as *mediator* in Freebase. This is a typical way of representing $n$-ary relations for $n > 2$ in triple stores.[27] While this may sound restrictive, it still allows answering a wide variety of questions, as can be seen from our evaluation.

[27] More generally, one can represent a $k$-ary relation by a special entity (one for each $k$-tuple in the relation) and $k$ binary relations between the special entity and the $k$ entities of the tuple.

Given that the involved problems are mainly concerned with natural language, it is unsurprising that a lot of recent work on question answering on knowledge bases has come from the natural language community. There, it is addressed as part of *semantic parsing*: translating the meaning of a sentence (or in this case, question) to a formal representation of its meaning (in this case, the SPARQL query)

Recent work can be roughly categorized into two groups. The first group only considers a fixed set of query structures as we do, e.g., [86, 85, 25, 22, 83, 88]. This puts the focus on the hard problems of matching relations and entities in the question. The second group of approaches allows a translation to (in theory) arbitrary queries, e.g., [30, 20, 53, 21, 1]. These systems either rely on a pre-trained syntactic parser or, essentially, are parsers by themselves. This increases complexity and introduces new problems. In particular, the huge space of possible patterns must be searched efficiently. Note, however, that the two approaches are complementary. Any system will have to deal with the problem of matching relations and entities. In contrast to our approach, previous approaches lack a proper addressing of the entity matching problem and either assume matching entities are given or only perform very simplistic matching. A very recent system that is close to ours is [1]. It follows our approach in applying a learning-to-rank strategy and focuses on answering more complex queries. More details on related approaches and systems can be found in our publication [17] and survey on semantic search [16, Section 4.7].

Besides quality, we also consider efficiency aspects, which is neglected in previous work. In particular, we took care that questions can be answered in an interactive manner, that is, within a second. Furthermore, we made sure that our approach works well on multiple datasets (with different kind of questions). A lot of previous approaches were evaluated only on one dataset.

Our main contributions are the following:

- A new end-to-end system that translates a natural language question into a SPARQL query on a knowledge base. In contrast to previous work, the system treats entity identification in the question as an integral part of the problem and also considers efficiency aspects.
- A joint disambiguation of entities and relations mentioned in the question. This is achieved by applying learning-to-rank techniques to learn a pairwise comparator of query candidates. Previous approaches framed this ranking as a classification or parsing problem.

- An improvement over the state of the art (at the time of publication) on two established benchmarks in terms of quality and efficiency. The two benchmarks exhibit quite different challenges. Many previous systems were evaluated and work well on only one of the benchmarks.

### 3.4.2 Approach

In the following, we describe how to answer natural language questions from a knowledge base. We use Freebase for our examples and in our evaluation. However, our approach is not specific to Freebase, but works for any knowledge base with entities and possibly $n$-ary relations between them.

On a high level, the approach can be split into four steps, which we explain in the following. As a running example, assume we are trying to answer the following question (from the WebQuestions benchmark):

*what character does ellen play in finding nemo?*

**Entity identification**. We begin by identifying entities from the knowledge base that are mentioned in the question. In our example, *ellen* refers to the tv host *Ellen DeGeneres* and *finding nemo* refers to the movie *Finding Nemo*. However, these words are ambiguous, for example, *ellen* could also refer to the actor *Ellen Page* and *finding nemo* to a video game with the same name.

Often, entities are not mentioned with their full knowledge base name, but with a synonym or alias. To find these, we utilize CrossWikis [78], which was built by mining the anchor text of links to Wikipedia entities (articles) from various large web crawls. CrossWikis covers around 4 million Wikipedia entities and provides prior probability distributions $p(e|s)$, of entity $e$ being mentioned by text span $s$. We extend these distributions with information about Freebase entities (details are in our publication [17]). This way, we are able to recognize around 44 million entities with about 60 million aliases. In order to keep the number of matched entities as small as possible without affecting recall we apply part-of-speech (POS) tagging to the question and restrict matches to reasonable POS sequences (mainly nouns).

Note that the disambiguation above can be helped by facts from the knowledge base. In our example, the fact that *Ellen DeGeneres* actually performed in the movie *Finding Nemo* makes the joint mentioning of both entities more likely. Therefore, instead of fixing the mentioned entities at this point, we delay the decision and jointly disambiguate entities and relations in a later step. The result of this step is a set of entity mentions $p(e|s)$ with attached probabilities. Subsequent steps also make use of a popularity score of each entity.

**Template matching**. Next, we match a set of query templates to the question using the previously identified entities. Figure 6 shows our templates. Each template consists of entity and relation placeholders. A matched template has these placeholders filled and corresponds to a *query candidate*, which can be executed against the knowledge base to obtain an answer.

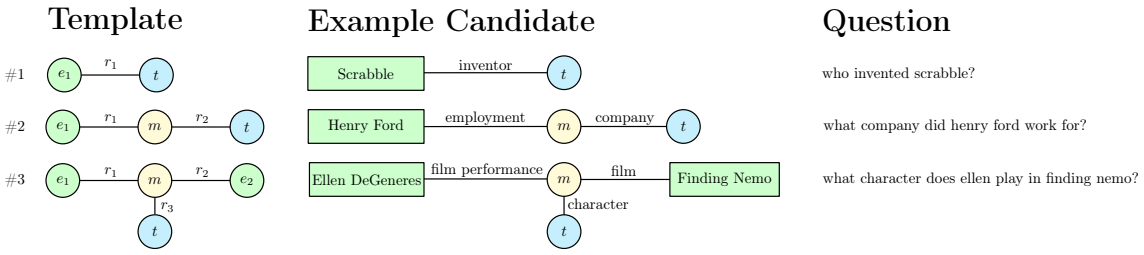


**Figure 6.** From our publication [17]: Query templates and example candidates with corresponding questions. A query template contains entity placeholders $e_i$, relation placeholders $r_i$, a mediator $m$ and an answer node $t$.

Conceptually, we match the templates as follows. Let $E$ be the set of all entities matched to a subsequence of the question in the previous step. Consequently, a word in the question can be part of several entity matches. For each template, fill the entity placeholder(s) with entities $e_i \in E$ for which the words in the question don't overlap. Then, look up relations $R_i$ for each entity $e_i$ in the knowledge base and create a query candidate for each relation. If multiple entities are part of the template, the relations must connect them in the knowledge base. This avoids generation of query candidates that have no answer.

For our example, we match template #3, besides others, as follows. We map $e_1$ and $e_2$ to *Ellen DeGeneres* and *Finding Nemo*, respectively. Then we look up relations $r_1$ as *film performance* and $r_2$ as *film*. These connect the two entities in the knowledge base because Ellen DeGeneres played in Finding Nemo. One of the possible relations for $r_3$ is *character*, which generates the correct candidate shown in Figure 6 (center of bottom row).

Technically, this matching process is more involved to perform efficiently. In particular, for matching template #3, we need to find $r_1$ and $r_2$ in connected triples $(e_1, r_1, m)$, $(m, r_2, e_2)$ given $e_1$ and $e_2$. Such queries are slow in current triple stores.[28] We, therefore, constructed a special inverted index for faster lookup. Details can be found in our publication [17].

In this step, we favor recall over precision and generate a lot of query candidates, most of them wrong. Wrong candidates can be identified in a later step, but a missing correct candidate will lead to a wrong final answer. For our example question, we generate 356 candidates, only one of which of is correct. The final result of this step is the set of all generated query candidates.

[28] In our experiments we use Virtuoso: `http://virtuoso.openlinksw.com`.

**Relation matching**. The query candidates still miss the fundamental information about which relations were actually mentioned in the question. We distinguish four ways of matching relations of a query candidate to the question text:

- *Relation name*: We match the name of the relation to words in the question. We distinguish between literal matches (e.g., the relation named *character* matches the word *character*), synonyms based on word embeddings and cosine distance [60] (e.g., *started* matches *founded*), and word derivations extracted from Wordnet [61] (e.g., *high* matches *elevation*).
- *Distant supervision* [62]: We learn indicator words for each relation using text from Wikipedia where entity mentions were identified. This allows deriving noisy training examples: A sentence expresses relation $r$ if it contains two co-occurring entities that are in relation $r$ according to a knowledge base. For each relation, we rank the words by their tf-idf to learn, for example, that *born* is a good indicator for the relation *place of birth*.
- *Supervised learning*: Our evaluation benchmarks come with a training set of questions and corresponding answer entities. We can use this to derive positive and negative training examples: Generate all query candidates for a question; if a query candidate answers the question it is a positive, otherwise, a negative example. As indicator features, we concatenate the n-grams of the question with the relation name. This way we can learn a logistic regression classifier.
- *Deep (supervised) learning*: Using the same supervised data, we can train a deep neural network instead of a logistic regression classifier. This is an extension of our work and not described in [17]. We provide technical details below (Section 3.4.4). In our experiments, we explicitly state when we use the neural network.

Each of the techniques provides a confidence score that we use for ranking candidates.

In our example, a word learned using distant supervision for the relations *film performance* and *film* (between an actor and the film she acted in), is *play*. Furthermore, the word *character* matches the relation with the same name.

**Ranking**. We now have a set of query candidates with information about which entities and relations match which parts of the question how well. In a final step, we rank these to find the best matching candidate. Note that performing ranking at this final step has the strong benefit of jointly disambiguating entities and relations. A candidate can have a weak match for an entity, but a strong match for a relation, and vice versa. We can identify these combinations as correct, even when one of the matches seems unlikely when considered separately.

To rank the candidates, we apply learning to rank [57]. The training data from the benchmarks allows sorting the generated candidates for each question by how well they answer it. Using the ranked list of candidates, we learn a comparator (a random forest [26]) that, given two candidates, decides which should be ranked first. We engineer a total of 23 features for this. These indicate how well entities and relations match but also include more general features like coverage (ratio of question words matched by an entity or relation) or result size. The exact features are listed in our publication [17].

For our example above, the candidate covering most words of the question is best. Matching *ellen* to *Ellen Page* no longer allows matching *Finding Nemo* because these aren't related in the knowledge base. On the other hand, the *character* relation matches the word *character* in the question, which is not the case for alternative relations like *performance type*. This leaves us with the correct interpretation of asking for *Ellen DeGeneres*' character in *Finding Nemo*.

### 3.4.3 Experiments and Results

To evaluate our system, we used two established benchmarks: *Free917* [30] and *WebQuestions* [20]. Each benchmark consists of a set of questions and their answers (one or more entities) from Freebase. As our knowledge base, we used the original Freebase data and not the curated version we described in Section 3.1. This has two reasons. First, we want to use all available facts in our answering process, and our curation simplifies $n$-ary relations (like the film performance or leadership relations above). Second, and more importantly, this allows a fair comparison with other approaches.

Each benchmark comes with a pre-defined set of training and test questions. The two benchmarks differ substantially in the types of questions and their complexity.

Free917 contains 917 manually generated natural language questions. The questions cover a wide range of domains. Two examples are:

| | |
|---|---|
| *who won the 1964 united states presidential election?* | answer: *Lyndon B. Johnson* |
| *how many languages has jrr tolkein created?*[29] | answer: *10* |

Questions are mostly grammatical and tend to be tailored to Freebase. The benchmark also provides an *entity lexicon*: a manually constructed mapping from text (the surface form) to the mentioned entity. This was used for identifying entities by all systems reporting results on the dataset so far. We only make use of this lexicon where explicitly stated. The established evaluation measure on this benchmark is *accuracy*, the fraction of queries answered with the exact gold answer.

[29] The typo in *tolkien* is indeed part of the dataset.

WebQuestions consists of 5,810 questions that were selected by crawling the Google suggest API. Contrary to Free917, questions are not necessarily grammatical and are more colloquial. For example:

*who brad pitt has dated?* answer: *Angelina Jolie, Jennifer Anniston, ...*
*who plays dwight in the office?* answer: *Rainn Wilson*

Due to the selection process, questions are biased towards topics that are frequently asked from Google. Furthermore, the structure of questions tends to be simpler. Answers to the questions were obtained by using crowdsourcing. This introduces additional noise. In particular, for some questions, only a subset of the correct answer is provided as gold answer. Therefore, what is usually reported is *average F1*: the F1 measure for each question (obtained by comparing the gold answer set and the system's answer set of entities) averaged over all questions.

Table 6 compares the quality of our system, Aqqu, to recent systems:

| | | Free917 | | WebQuestions |
|---|---|---|---|---|
| Method | Venue | Accuracy+ | Accuracy | Average F1 |
| Cai et al. [30] | ACL'13 | 59 % | – | – |
| Jacana [86] | ACL'14 | – | – | 35.4 % |
| Sempre [20] | EMNLP'13 | 62 % | 52 % | 35.7 % |
| Kwiat. et al. [53] | EMNLP'13 | 68 % | – | – |
| Bordes et al. [25] | EMNLP'14 | – | – | 39.2 % |
| ParaSempre [22] | ACL'14 | 68.5 % | 46 % | 39.9 % |
| **Aqqu** [17] | CIKM'15 | **76.4 %** | **65.9 %** | **49.4 %** |
| STAGG [88] | ACL'15 | – | – | 52.5 % |
| Berant et al. [21] | TACL'15 | – | – | 49.7 % |
| Xu et al. [83] | CoRR'16 | – | – | 53.3 % |
| Reddy et al. [73] | TACL'16 | 78.0 % | – | 50.3 % |
| QUINT [1] | WWW'17 | 72.8 % | – | 51.0 % |
| **Aqqu + NN** | – | **78.7 %** | **70.2 %** | **51.8 %** |

**Table 6.** Results on the Free917 (267 questions) and WebQuestions (2032 questions) test set. Results of our system, Aqqu, in bold. The bottom part of the table corresponds to work published after Aqqu [17]. For the results in the third column (Accuracy+), a manually crafted entity lexicon was used. Note that better performing systems fundamentally relied on external data (see text).

Most of the systems in Table 6 have only been evaluated on one of the two benchmarks. Our system, *Aqqu*, uses a single approach that considerably improved the state of the art on both benchmarks at the time of publication. The extension with a deep neural network for relation matching, *Aqqu + NN* (see Section 3.4.4), is also competitive for the current state of the art, in particular, since other approaches make considerably more use of external data than our rather shallow learning of indicator words via distant supervision (see Section 3.4.2). Xu et al. [83] issue queries against a full-text search engine on Wikipedia during the answering process. STAGG [88] uses large amounts of additional data (derived via distant supervision) to train their neural network. Without this external data, [83] report a drop of 6.2% and [88] a drop of 0.9% in average F1. Both techniques are likely to benefit our approach as well.

To inspect how many questions can be answered by our query templates we analysed oracle results: the score achievable when assuming perfect ranking. On Free917 *Aqqu* achieves an oracle accuracy of 85% (without manual entity identification) and on WebQuestions an oracle average F1 of 68%. Note that, as explained above, the WebQuestions dataset consists of frequent but very noisy questions that often have an incomplete ground truth. In a manual inspection we estimate that the best achievable score is around 80%. This shows that our patterns allow answering the majority of questions with high quality.

Since we focused on rather short and frequent questions our templates will not work well for arbitrarily complex questions. The work in [1] shows that templates can also be learned automatically in order to answer compositional questions such as *Which were the alma maters of the PR managers of Hillary Clinton?*. The described approach is orthogonal to ours and could be integrated on top of our current system. We consider this direction worthwhile future work.

On both benchmarks we also look at the top-k results of *Aqqu*. The best candidate is within the top two in 74% of questions for Free917 and 67% for WebQuestions. This demonstrates that our learned ranking is very strong. Indeed, in many cases, the top two candidates are hard to distinguish and often match the question very well. For example, *where is chris paul from?* can be answered with his place of birth or his nationality, but only one interpretation is considered correct in the ground truth.

Besides quality, we also evaluated efficiency of our system. On average, *Aqqu* answers a question within 217ms and 143ms on the test sets of Free917 and WebQuestions, respectively.[30] *Aqqu + NN* requires 193ms and 169ms, respectively. Berant et al. [21] report 291ms per question on average for WebQuestions. For other systems that provide code and for which we reproduced results, run times are (at least) several seconds per query.

In an error analysis, we found that there is no single large source of errors worth pointing out. Instead, each of the components (entity recognition, pattern and relation matching, ranking) fails about equally often for various reasons. The accompanying materials of our publication [17] provide a list of errors. Our publication also includes a more detailed evaluation and analyses of, for example, feature and component importances.

### 3.4.4 Matching Relations in Questions Using Deep Learning

We shortly describe the neural network used in *Aqqu + NN* above. This is an extension that is not part of our original publication [17]. The neural network computes a score that indicates how well a candidate query and its relations matches a question. Essentially, it is an additional, more sophisticated approach of the supervised relation matching classifier. The neural network score is used in ranking query candidates. Besides this additional ranking feature, all other components of the system are identical.

The architecture of the neural network is similar to that of [88]: a Siamese neural network [27] that learns a real-valued vector representation (embedding) of the question and knowledge base query. The cosine between both vector representations is used as a score on how well the question matches (translates to) the query. Ideally, the representation of the correct query is close to the representation of the question and the resulting cosine similarity is large.

To compute an embedding of the question, we first transform it into a sequence of 128-dimensional word vectors. Word vectors are learned using word2vec [60] on a corpus of 50 million sentences extracted from ClueWeb [34]. The vectors are kept fixed during training of the neural network, i.e., we perform no fine-tuning. On the sequence of vectors we apply a 1-d convolution followed by max-pooling as described in [50]. The convolution uses filters of size one, two, and three with 300 filters of each size. The output of the convolution layer is followed by a fully connected feed-forward network with 200 nodes.

[30] These numbers differ from our original publication [17] because we have improved the implementation in obvious ways, e.g., by avoiding unnecessary re-computations of features. Originally we reported 644ms and 900ms, respectively. New experiments were performed on a system with Intel i7-6700 CPUs, 60GB of RAM, and a Titan X GPU.

To compute an embedding of a knowledge base query, we split each involved relation into its three inherent parts separated by a dot. Each part is then represented as the average of its word vectors (some parts have several words like *place_of_birth* in *people.person.place_of_birth*). We use the same word vectors as above that are also fixed during training. Because a query can have up to three relations (see the templates in Section 3.4.2) this results in nine 128-dimensional vectors. Their concatenation is passed through two fully connected feed-forward layers with 200 nodes each.

All nodes of the network use exponential linear units [33] as activation function. The 200-dimensional outputs for the question and query representation are then compared using their cosine, which gives the final output of the network.

The training data is the same as for the supervised relation matching classifier, i.e., it is constructed from the training questions of the benchmark. We train the network using Adam [51] for a fixed duration of 30 epochs, determined after observing performance on a development set. As loss function, we use mean squared error.

We also experimented with many variations and extensions: using dropout for regularization, different activation functions, different loss functions, and optimizers. In our experiments, none gave performance improvements on the development set over the network described above.

## 3.5 Semantic Search Survey

In this section, we describe our extensive survey (156 pages) on semantic search on text and knowledge bases [16]. We start by illustrating our motivation for the survey followed by giving a short outline.

As we stated in the introduction, semantic search is not a single well-defined problem. Rather, it is understood by many different communities in different ways. As a result, researchers are often not aware of related work in other communities, though the addressed problems are similar. This is the main motivation behind the survey. It should give an extensive overview of addressed problems, their approaches, and state of the art in the different communities. Crucially, no such overview was available in the literature. Our survey categorizes the vast research field. It also explains basic techniques which underpin many of the described approaches and provides details on advanced techniques. This should make it useful to newcomers as well as seasoned researchers.

To categorize the research, we devised a classification scheme along two dimensions: the *type of data* that is searched on and the *search paradigm*. We consider this classification scheme a major contribution. It can serve as a guideline to navigate the field, categorize new approaches and techniques, and find related work in that area. Figure 7 shows the categories according to our classification scheme. We shortly describe each of the dimensions.

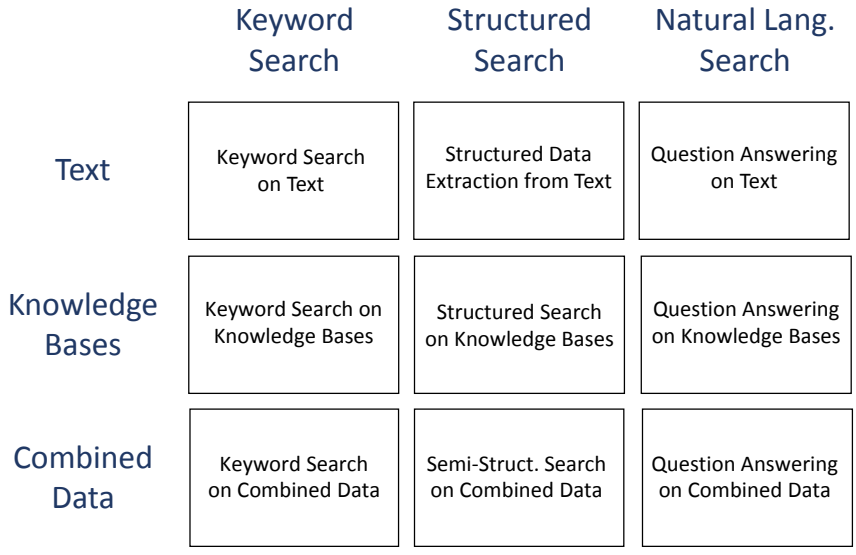


**Figure 7.** Adapted from our survey [16]: Our basic classification of research on semantic search into categories by underlying data (rows) and search paradigm (columns). Each category corresponds to a subsection in the survey.

Data in the form of *text* usually consists of a collection of documents that contain natural language. This is the most abundant kind of information available, but completely unstructured. A typical example is the Web, where each web page corresponds to a document. *Knowledge bases*, on the other hand, contain structured statements, often in the form of *subject predicate object* triples. An important trait is that identifiers of entities and relations are used consistently, that is, the same entity or relation should have the same identifier in all triples. A typical example of a knowledge base is Freebase. *Combined data* refers to a combination of the two previous types. Entities from a knowledge base can be linked to their mentions in text. This is the kind of data semantic full-text search (see Section 3.1) uses. Combined data also refers the case where several knowledge bases (with different naming schemes) are combined into one huge interlinked knowledge base. In this case, the same entity or relation can exist multiple times with different identifiers. A typical example is *linked open data*, the data behind the Semantic Web.

The classification into our three search paradigms is as follows. *Keyword search* accepts a list of (typically few) keywords. It places no restriction on the structure of the query. This is still the best known and most ubiquitous search paradigm. *Structured search* uses a query language, like SQL or SPARQL. Queries must adhere to the syntax defined by the language. This is the obvious choice when the data to be searched is structured. The language can also be extended to incorporate search in unstructured data, for example with keyword search in text, as in semantic full-text search (Section 3.1). *Natural language search* answers complete questions as a human would pose them. We presented an approach for question answering on a knowledge base in this document in Section 3.4.

Our classification leads to a total of nine categories. Each category corresponds to a subsection that contains: a profile of the corresponding line of research (including strengths and limitations) and a description of the basic techniques, important systems, benchmarks, and the state of the art. These nine categories form the core section of the survey.

In addition, our survey provides a lot of supplementary information, for example, we list and reference important datasets for each category. We also provide introductory and background information, for example, we dedicate a whole section to fundamental natural language processing techniques. In a final section on advanced techniques, we address ranking, indexing, ontology matching and merging, as well as inference.

Altogether, we believe the survey is a valuable contribution to the semantic search community that was missing and long overdue, especially since the field is so vast, hard to delineate, and difficult to get an overview of.

# 4 Future Work

The ultimate semantic search engine is one that has human-like understanding of the world. It is able to perfectly determine the intent of each query, whether given as keywords or complete question. And it can reason about the answer using whatever data it has at its disposal, much like a human. Such a level of understanding and where it might come from is still elusive. Nonetheless, we consider the following three directions of future research as promising steps (albeit comparably small) towards this ultimate goal.

First, extending the question answering approach presented in Section 3.4.2 to use full text in addition to a knowledge base will allow answering a wider range of questions. The obvious way to achieve this is to make use of the *occurs-with* relation from semantic full-text search (see Section 3.1). The main challenge will be to determine when to make use of the text in addition to the knowledge base. For example, for questions like *who is the ceo of apple?* results could always be kept up-to-date with a combined search in full text and a knowledge base. However, searching full text is not necessary and possibly detrimental in other cases, for example, for *who founded apple?*, which is answered perfectly from the knowledge base alone.

Second, using semantic completions may direct a user to formulate questions that are less ambiguous and better understood. For example, after typing "*who played rick deckard in*" the user can be presented suggestions for the different Blade Runner movies. If she selects one of the suggestions, no ambiguity arises compared to typing *blade runner*, which can mean the movie from 1982 or 2017. This requires that the question typed so far is sufficiently understood, for example, that the movie character Rick Deckard has been identified and that the list of suggestions contains movies in which he appears. Suggestions could be provided for all words of the question, not only for entities. In particular, words that describe relations between entities, such as *played* above, are a major source of ambiguity (see Section 3.4) and could be suggested as well. There is a large body of work on auto-completing keyword queries summarized in [29]. However, only a few works make use of entities or knowledge bases, and no work seems to address the problem for question answering. Preliminary experiments indicate that a class-based language model can learn meaningful suggestions. There is also recent work showing how knowledge base facts can be incorporated in a neural language model [2].

Finally, it will be interesting to apply recent advances in learning neural networks in an end-to-end fashion to answer questions [55, 67, 66]. This has the potential to enable search on multiple data sources by learning which data source is reliable for which kind of information. For example, it may be possible to learn when to query text in addition to a knowledge base as in the example above. However, to achieve good results, large amounts of training data may be required, and to answer a wide range of questions, it will

be necessary to transfer the learnings from one domain or dataset to others - which is still an open research problem.

The problems we have addressed in this thesis have a wide range of applications in semantic search. In particular, they can serve as building blocks in the suggestions outlined above. Together, this represents a promising direction of future work which may bring us a step closer to the ultimate semantic search engine.

# References

The following lists the references cited in this summary. Note that the list of references accumulated in all publications given in Section 2, especially the survey (see Section 3.5), is much larger. In the digital version of this dissertation, titles below are clickable and link to an electronic copy.


[1] A. Abujabal, M. Yahya, M. Riedewald, and G. Weikum. "Automated Template Generation for Question Answering over Knowledge Graphs". In: *WWW*, 2017, pp. 1191–1200.

[2] S. Ahn, H. Choi, T. Pärnamaa, and Y. Bengio. "A Neural Knowledge Language Model". In: *CoRR*, 2016.

[3] P. Bailey, R. W. White, H. Liu, and G. Kumaran. "Mining Historic Query Trails to Label Long and Rare Search Engine Queries". In: *TWEB*, 2010, 15:1–15:27.

[4] K. Balog, P. Serdyukov, and A. P. de Vries. "Overview of the TREC 2010 Entity Track". In: *TREC*, 2010.

[5] K. Balog, P. Serdyukov, and A. P. de Vries. "Overview of the TREC 2011 Entity Track". In: *TREC*, 2011.

[6] K. Balog, A. P. de Vries, P. Serdyukov, P. Thomas, and T. Westerveld. "Overview of the TREC 2009 Entity Track". In: *TREC*, 2009.

[7] M. Banko, M. J. Cafarella, S. Soderland, M. Broadhead, and O. Etzioni. "Open Information Extraction from the Web". In: *IJCAI*, 2007, pp. 2670–2676.

[8] H. Bast, A. Chitea, F. M. Suchanek, and I. Weber. "ESTER: Efficient Search on Text, Entities, and Relations". In: *SIGIR*, 2007, pp. 671–678.

[9] H. Bast, F. Bäurle, B. Buchhold, and E. Haußmann. "A Case for Semantic Full-Text Search". In: *SIGIR-JIWES*, 2012, 4:1–4:3.

[10] H. Bast, F. Bäurle, B. Buchhold, and E. Haußmann. "Broccoli: Semantic Full-Text Search at your Fingertips". In: *CoRR*, 2012.

[11] H. Bast, F. Bäurle, B. Buchhold, and E. Haußmann. "Easy Access to the Freebase Dataset". In: *WWW*, 2014, pp. 95–98.

[12] H. Bast, F. Bäurle, B. Buchhold, and E. Haußmann. "Semantic Full-Text Search with Broccoli". In: *SIGIR*, 2014, pp. 1265–1266.

[13] H. Bast and B. Buchhold. "An Index for Efficient Semantic Full-Text Search". In: *CIKM*, 2013, pp. 369–378.

[14] H. Bast, B. Buchhold, and E. Haussmann. "A Quality Evaluation of Combined Search on a Knowledge Base and Text". In: *KI*, 2018, pp. 19–26.

[15] H. Bast, B. Buchhold, and E. Haußmann. “Relevance Scores for Triples from Type-Like Relations”. In: *SIGIR*, 2015, pp. 243–252.

[16] H. Bast, B. Buchhold, and E. Haußmann. “Semantic Search on Text and Knowledge Bases”. In: *Foundations and Trends in Information Retrieval*, 2016, pp. 119–271.

[17] H. Bast and E. Haußmann. “More Accurate Question Answering on Freebase”. In: *CIKM*, 2015, pp. 1431–1440.

[18] H. Bast and E. Haußmann. “More Informative Open Information Extraction via Simple Inference”. In: *ECIR*, 2014, pp. 585–590.

[19] H. Bast and E. Haußmann. “Open Information Extraction via Contextual Sentence Decomposition”. In: *ICSC*, 2013, pp. 154–159.

[20] J. Berant, A. Chou, R. Frostig, and P. Liang. “Semantic Parsing on Freebase from Question-Answer Pairs”. In: *ACL*, 2013, pp. 1533–1544.

[21] J. Berant and P. Liang. “Imitation Learning of Agenda-based Semantic Parsers”. In: *TACL*, 2015, pp. 545–558.

[22] J. Berant and P. Liang. “Semantic Parsing via Paraphrasing”. In: *ACL*, 2014, pp. 1415–1425.

[23] R. Blanco, P. Mika, and S. Vigna. “Effective and Efficient Entity Search in RDF Data”. In: *ISWC*, 2011, pp. 83–97.

[24] K. D. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor. “Freebase: a Collaboratively Created Graph Database for Structuring Human Knowledge”. In: *SIGMOD*, 2008, pp. 1247–1250.

[25] A. Bordes, S. Chopra, and J. Weston. “Question Answering with Subgraph Embeddings”. In: *EMNLP*, 2014, pp. 615–620.

[26] L. Breiman. “Random Forests”. In: *Machine Learning*, 2001, pp. 5–32.

[27] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah. “Signature Verification Using a Siamese Time Delay Neural Network”. In: *NIPS*, 1993, pp. 737–744.

[28] M. Bron, K. Balog, and M. de Rijke. “Ranking Related Entities: Components and Analyses”. In: *CIKM*, 2010, pp. 1079–1088.

[29] F. Cai and M. de Rijke. “A Survey of Query Auto Completion in Information Retrieval”. In: *Foundations and Trends in Information Retrieval*, 2016, pp. 273–363.

[30] Q. Cai and A. Yates. “Large-Scale Semantic Parsing via Schema Matching and Lexicon Extension”. In: *ACL*, 2013, pp. 423–433.

[31] J. Chen, C. Xiong, and J. Callan. “An Empirical Study of Learning to Rank for Entity Search”. In: *SIGIR*, 2016, pp. 737–740.

[32] A. Chuklin, I. Markov, and M. de Rijke. “Click Models for Web Search”. Synthesis Lectures on Information Concepts, Retrieval, and Services. Morgan & Claypool Publishers, 2015.

[33] D. Clevert, T. Unterthiner, and S. Hochreiter. “Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs)”. In: *CoRR*, 2015.

[34] ClueWeb. The Lemur Project, http://lemurproject.org/clueweb09. 2009.

[35] R. Collobert et al. “Natural Language Processing (Almost) from Scratch”. In: *Journal of Machine Learning Research*, 2011, pp. 2493–2537.

[36] L. D. Corro, A. Abujabal, R. Gemulla, and G. Weikum. “FINET: Context-Aware Fine-Grained Named Entity Typing”. In: *EMNLP*, 2015, pp. 868–878.

[37] L. D. Corro and R. Gemulla. “ClausIE: Clause-Based Open Information Extraction”. In: *WWW*, 2013, pp. 355–366.

[38] X. L. Dong et al. “From Data Fusion to Knowledge Fusion”. In: *PVLDB*, 2014, pp. 881–892.

[39] S. Elbassuoni, M. Ramanath, R. Schenkel, M. Sydow, and G. Weikum. “Language-Model-Based Ranking for Queries on RDF-graphs”. In: *CIKM*, 2009, pp. 977–986.

[40] O. Etzioni et al. “Web-scale Information Extraction in KnowItAll: (Preliminary Results)”. In: *WWW*, 2004, pp. 100–110.

[41] A. Fader, S. Soderland, and O. Etzioni. “Identifying Relations for Open Information Extraction”. In: *ACL*, 2011, pp. 1535–1545.

[42] Y. Fang, L. Si, Z. Yu, Y. Xian, and Y. Xu. “Entity Retrieval with Hierarchical Relevance Model, Exploiting the Structure of Tables and Learning Homepage Classifiers”. In: *TREC*, 2009.

[43] P. Ferragina and U. Scaiella. “TAGME: On-the-Fly Annotation of Short Text Fragments (by Wikipedia Entities)”. In: *CIKM*, 2010, pp. 1625–1628.

[44] J. L. Fleiss. “Measuring Nominal Scale Agreement Among Many Raters”. In: *Psychological bulletin*, 1971, p. 378.

[45] E. Gabrilovich, M. Ringgaard, and A. Subramanya. “FACC1: Freebase Annotation of ClueWeb Corpora, Version 1”. (Release Date 2013-06-26, Format Version 1, Correction Level 0).

[46] M. Gupta and M. Bendersky. “Information Retrieval with Verbose Queries”. In: *Foundations and Trends in Information Retrieval*, 2015, pp. 91–208.

[47] H. Halpin et al. “Evaluating ad-hoc object retrieval”. In: *IWEST*, 2010.

[48] S. Heindorf, M. Potthast, H. Bast, B. Buchhold, and E. Haußmann. “WSDM Cup 2017: Vandalism Detection and Triple Scoring”. In: *WSDM*, 2017, pp. 827–828.

[49] J. Hoffart, D. Milchevski, and G. Weikum. “STICS: Searching with Strings, Things, and Cats”. In: *SIGIR*, 2014, pp. 1247–1248.

[50] Y. Kim. “Convolutional Neural Networks for Sentence Classification”. In: *ACL*, 2014, pp. 1746–1751.

[51] D. P. Kingma and J. Ba. “Adam: A Method for Stochastic Optimization”. In: *CoRR*, 2014.

[52] E. Kiperwasser and Y. Goldberg. “Simple and Accurate Dependency Parsing Using Bidirectional LSTM Feature Representations”. In: *TACL*, 2016, pp. 313–327.

[53] T. Kwiatkowski, E. Choi, Y. Artzi, and L. S. Zettlemoyer. “Scaling Semantic Parsers with On-the-Fly Ontology Matching”. In: *EMNLP*, 2013, pp. 1545–1556.

[54] X. Li, B. J. A. Schijvenaars, and M. de Rijke. “Investigating Queries and Search Failures in Academic Search”. In: *Information Processing and Management*, 2017, pp. 666–683.

[55] C. Liang, J. Berant, Q. Le, K. D. Forbus, and N. Lao. “Neural Symbolic Machines: Learning Semantic Parsers on Freebase with Weak Supervision”. In: *CoRR*, 2016.

[56] D. Lin and P. Pantel. “DIRT @SBT@Discovery of Inference Rules from Text”. In: *KDD*, 2001, pp. 323–328.

[57] T. Liu. “Learning to Rank for Information Retrieval”. In: *Foundations and Trends in Information Retrieval*, 2009, pp. 225–331.

[58] Mausam, M. Schmitz, S. Soderland, R. Bart, and O. Etzioni. “Open Language Learning for Information Extraction”. In: *EMNLP*, 2012, pp. 523–534.

[59] M. Melucci. “Contextual Search: A Computational Framework”. In: *Foundations and Trends in Information Retrieval*, 2012, pp. 257–405.

[60] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. “Distributed Representations of Words and Phrases and their Compositionality”. In: *NIPS*, 2013, pp. 3111–3119.

[61] G. A. Miller. “WordNet: A Lexical Database for English”. In: *Commun. ACM*, 1995, pp. 39–41.

[62] M. Mintz, S. Bills, R. Snow, and D. Jurafsky. “Distant Supervision for Relation Extraction Without Labeled Data”. In: *ACL*, 2009, pp. 1003–1011.

[63] T. M. Mitchell et al. “Never-Ending Learning”. In: *AAAI*, 2015, pp. 2302–2310.

[64] N. Nakashole, T. Tylenda, and G. Weikum. “Fine-grained Semantic Typing of Emerging Entities”. In: *ACL*, 2013, pp. 1488–1497.

[65] N. Nakashole, G. Weikum, and F. M. Suchanek. “PATTY: A Taxonomy of Relational Patterns with Semantic Types”. In: *EMNLP-CoNLL 2012, July 12-14, 2012, Jeju Island, Korea*, 2012, pp. 1135–1145.

[66] K. Narasimhan, A. Yala, and R. Barzilay. “Improving Information Extraction by Acquiring External Evidence with Reinforcement Learning”. In: *EMNLP*, 2016, pp. 2355–2365.

[67] A. Neelakantan, Q. V. Le, M. Abadi, A. McCallum, and D. Amodei. “Learning a Natural Language Interface with Neural Programmer”. In: *CoRR*, 2016.

[68] H. Paulheim and C. Bizer. “Improving the Quality of Linked Data Using Statistical Distributions”. In: *Int. J. Semantic Web Inf. Syst.* 2014, pp. 63–86.

[69] B. Popov et al. “KIM - Semantic Annotation Platform”. In: *ISWC*, 2003, pp. 834–849.

[70] J. Pound, P. Mika, and H. Zaragoza. “Ad-hoc Object Retrieval in the Web of Data”. In: *WWW*, 2010, pp. 771–780.

[71] D. Ramage, D. L. W. Hall, R. Nallapati, and C. D. Manning. “Labeled LDA: A Supervised Topic Model for Credit Attribution in Multi-Labeled Corpora”. In: *ACL*, 2009, pp. 248–256.

[72] L. Ratinov, D. Roth, D. Downey, and M. Anderson. “Local and Global Algorithms for Disambiguation to Wikipedia”. In: *ACL*, 2011, pp. 1375–1384.

[73] S. Reddy et al. “Transforming Dependency Structures to Logical Forms for Semantic Parsing”. In: *TACL*, 2016, pp. 127–140.

[74] M. Sanderson. “Test Collection Based Evaluation of Information Retrieval Systems”. In: *Foundations and Trends in Information Retrieval*, 2010, pp. 247–375.

[75] S. Sarawagi. “Information Extraction”. In: *Foundations and Trends in Databases*, 2008, pp. 261–377.

[76] M. Schuhmacher, L. Dietz, and S. P. Ponzetto. “Ranking Entities for Web Queries Through Text and Knowledge”. In: *CIKM*, 2015, pp. 1461–1470.

[77] F. Silvestri. “Mining Query Logs: Turning Search Usage Data into Knowledge”. In: *Foundations and Trends in Information Retrieval*, 2010, pp. 1–174.

[78] V. I. Spitkovsky and A. X. Chang. “A Cross-Lingual Dictionary for English Wikipedia Concepts”. In: *LREC*, 2012, pp. 3168–3175.

[79] F. M. Suchanek, G. Kasneci, and G. Weikum. “YAGO: a Core of Semantic Knowledge”. In: *WWW*, 2007, pp. 697–706.

[80] V. Tablan, K. Bontcheva, I. Roberts, and H. Cunningham. “Mímir: An Open-Source Semantic Search Framework for Interactive Information Seeking and Discovery”. In: *J. Web Sem.* 2015, pp. 52–68.

[81] T. Tran, P. Mika, H. Wang, and M. Grobelnik. “SemSearch’11: the 4th Semantic Search Workshop”. In: *WWW*, 2011, pp. 315–316.

[82] D. Vrandecic and M. Krötzsch. “Wikidata: a Free Collaborative Knowledgebase”. In: *CACM*, 2014, pp. 78–85.

[83] K. Xu, Y. Feng, S. Reddy, S. Huang, and D. Zhao. “Enhancing Freebase Question Answering Using Textual Evidence”. In: *CoRR*, 2016.

[84] B. Yang, W. Yih, X. He, J. Gao, and L. Deng. “Embedding Entities and Relations for Learning and Inference in Knowledge Bases”. In: *CoRR*, 2014.

[85] X. Yao, J. Berant, and B. V. Durme. “Freebase QA: Information Extraction or Semantic Parsing?” In: *ACL, Workshop on Semantic Parsing*, 2014.

[86] X. Yao and B. V. Durme. “Information Extraction over Structured Data: Question Answering with Freebase”. In: *ACL*, 2014, pp. 956–966.

[87] A. Yates et al. “TextRunner: Open Information Extraction on the Web”. In: *NAACL*, 2007, pp. 25–26.

[88] W. Yih, M. Chang, X. He, and J. Gao. “Semantic Parsing via Staged Query Graph Generation: Question Answering with Knowledge Base”. In: *ACL*, 2015, pp. 1321–1331.

[89] M. A. Yosef, J. Hoffart, I. Bordino, M. Spaniol, and G. Weikum. “AIDA: An Online Tool for Accurate Disambiguation of Named Entities in Text and Tables”. In: *PVLDB*, 2011, pp. 1450–1453.