Carnegie Mellon  Universität Karlsruhe (TH)

Deploying Semantic Resources for Open Domain Question Answering

Nico Schlaefer

Diploma Thesis

International Center for Advanced Communication Technologies

Carnegie Mellon University, Pittsburgh, USA
Universität Karlsruhe (TH), Germany

Advisors:

Dr. Petra Gieselmann
Prof. Dr. Eric Nyberg
Prof. Dr. Alex Waibel

January - June 2007

Hiermit erkläre ich, dass ich diese Diplomarbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Pittsburgh, den 28. Juni 2007

Nico Schlaefer

Abstract

This thesis investigates how semantic resources can be deployed to improve the accuracy of an open domain question answering (QA) system. In particular, two types of semantic resources have been utilized to answer factoid questions: (1) Semantic parsing techniques are applied to analyze questions for semantic structures and to find phrases in the knowledge source that match these structures. (2) Ontologies are used to extract terms from questions and corpus sentences and to enrich these terms with semantically similar concepts. These resources have been integrated in the Ephyra QA framework and were compared to previously developed syntactic answer extraction approaches.

A semantic extractor for factoid answers was devised that generates semantic representations of the question and phrases in the corpus and extracts answer candidates from phrases that are similar to the question. Different query generation techniques are used to retrieve relevant text passages from the corpus, ranging from simple keyword queries over compound terms expanded with synonyms to specific query strings built from predicate-argument structures. A fuzzy similarity metric compares semantic structures at the level of key terms by measuring their pairwise syntactic and semantic similarities and aggregates these term similarities into an overall similarity score. This mechanism is flexible and robust to parsing errors and it maximizes the recall of the semantic answer extractor. Score normalization and combination techniques allow merging answer candidates found with different semantic and syntactic extraction strategies.

Several ontologies are used to extract compound terms from questions and answer sentences and to expand terms with alternative representations. (1) A framework for domain-specific ontologies allows integrating expert knowledge on restricted domains. (2) WordNet is used as an open-domain resource of ontological knowledge. (3) A new approach for automatically learning semantic relation between entities and events in a textual corpus is introduced. Semantic structures are extracted from the corpus with a semantic parser and are subsequently transformed into a semantic network that reveals relations between the entities and events in the corpus.

These semantic query generation and answer extraction techniques were assessed on factoid questions from past TREC evaluations using the Web as a large open domain corpus, as well as a local domain-specific document collection. The evaluation results show that the semantic extraction approach has a higher precision than Ephyra's syntactic answer extractors and that a hybrid approach of semantic and syntactic answer extractors outperforms each individual technique. Furthermore, the query expansion techniques can be combined with existing syntactic extractors to boost their accuracy.

Acknowledgements

First of all, I would like to thank Petra Gieselmann, who first introduced me to the fascinating field of question answering about two years ago. Since then, she has continuously helped me with words and deeds and provided valuable input in numerous discussions.

I further wish to thank Eric Nyberg for accepting me in the JAVELIN research group at Carnegie Mellon University and for his profound advice on semantic question answering. Eric Nyberg shared his know-how with me in various meetings and has been of great support during my stay in Pittsburgh.

Special thanks go to Alex Waibel, who established the interACT center and the student exchange program between Universität Karlsruhe and Carnegie Mellon University. Alex Waibel gave me the opportunity to conduct research on question answering both in Pittsburgh and in his research group in Karlsruhe for the past two years. He always had a sympathetic ear for my concerns and supported me whenever he could.

Thanks also go to Eric Nyberg's students, especially to Andrew Schlaikjer for sharing his knowledge on semantic answer extraction, and to Matthew Bilotti for his input on structured retrieval and for providing me with versions of the AQUAINT and CNS corpora with semantic annotations.

Finally, I would like to appreciate the Landesstiftung Baden-Württemberg for sponsoring me with a Baden-Württemberg-Stipendium, an exchange scholarship that enabled me to visit Carnegie Mellon University and to compile this thesis in the first place.

Contents

List of Figures

List of Tables

1. Introduction

This chapter gives an introduction to question answering and motivates the application of semantic techniques to answer factoid questions in an open domain. It further outlines the scope of this thesis and reviews related work on semantic question answering.

1.1 Question Answering

The discipline of *question answering* (QA) is concerned with the retrieval of accurate answers to natural language questions from a textual corpus. QA systems differ from document retrieval systems in that the question may be posed in natural language and the answers are precise and to the point. Often, the Web is utilized as a large and redundant knowledge source. Local text corpora are used in expert systems for restricted domains and in evaluations to ensure the comparability of the results.

Common question types are *factoid questions* asking for concise answers, often named entities (e.g. "Which country is the largest by population?"), *list questions* seeking a list of such factoid answers (e.g. "Who are members of the European Parliament?"), and *definitional questions* requesting relevant information on a given topic (e.g. "What is Stonehenge?"). Future systems may also address *complex questions* that require composing an answer from multiple sources, for instance "Which countries build the A380 airplane and how do they contribute?".

The Text REtrieval Conference (TREC) [Voo02, DLK06] conducts an annual evaluation of various information retrieval tasks, including question answering, to foster research in the field. The first QA evaluations exclusively dealt with factoid questions, but in recent years, list questions and definitional questions have been adopted and additional challenges were introduced. The questions were grouped in series with a common topic and need to be answered in the context of this topic. Furthermore, questions may contain coreferences to the topic or preceding questions in the series and their answers. In past evaluations, systems were required to retrieve answers from the AQUAINT newswire corpus. Lately, this document collection has been replaced by a combination of the AQUAINT-2 corpus, which covers a more recent time frame, and a large collection of web logs.

The Cross-Language Evaluation Forum (CLEF) evaluates QA systems for various European languages including Bulgarian, Dutch, English, French, German, Italian, Portuguese and Spanish. In some subtasks, the knowledge base is in the same language as the questions (*monolingual QA*), other tasks require translating the question and retrieving answers from a corpus in a different language (*bilingual* or *cross-lingual QA*). The NTCIR workshop (NII Test Collection for IR Systems) is concerned with monolingual and cross-lingual QA in English, Japanese and Chinese; Korean is under consideration for future evaluations.

1.2 Motivation of Semantic Techniques

Factoid question answering systems often utilize simple syntactic techniques to identify relevant documents in the knowledge base and to extract instances of correct answers from these documents. A common approach determines the expected answer type of the question, transforms the question into a bag of keywords, searches the corpus for documents that contain these keywords and extracts answers of the expected type (e.g. dates for "when" questions) from these documents. Other syntactic approaches use textual patterns to extract answer candidates that occur in proximity with question terms.

These syntactic techniques proved to be fast, robust and surprisingly effective. They further have the advantage of being easy to implement and can be used to answer a wide range of factoid questions. On the other hand, pure syntactic approaches depend on the wording of the question and fail to recognize answers if different terms are used in the corpus. For instance, a question may ask for the "mean income of a lawyer" while the corpus terms it the "average earnings of an attorney". Furthermore, a phrase in the corpus may match the keywords of the question without preserving the semantic relations. For example, a keyword query formed from the question "Who killed Lee Harvey Oswald?" would retrieve phrases such as "Lee Harvey Oswald killed John F. Kennedy". Since the term "John F. Kennedy" is a person's name and it cooccurs with the question keywords, a simple extractor based on answer type analysis would falsely consider it as an answer candidate.

Furthermore, the answer may be given implicitly so that additional knowledge and logical reasoning is required to recognize it as correct. Consider the question "What nuclear technologies does Teheran possess?" and a knowledge base containing the (fictitious) statement "Iran recently acquired gas centrifuges from Pakistan". A human may intuitively recognize "gas centrifuges" as a correct answer. However, a question answering system would require general knowledge (the capital Teheran is used synonymously with Iran) as well as domain-specific expert knowledge (gas centrifuges are commonly used for uranium enrichment) to find the answer, and it would need to reason that acquiring implies possessing.

A pure syntactic approach requires a redundant knowledge source that contains multiple instances of an answer in varying formulations to compensate for these flaws. Semantic resources can be deployed to improve the recall on smaller corpora and to answer complex questions that require background knowledge and logical inference.

1.3 Scope and Contributions

This thesis primarily discusses two types of semantic resources: Semantic parsing techniques are utilized to identify phrases in the knowledge source that are similar to the question on a semantic level and ontologies represent world knowledge that is used to extract key terms from questions and corpus documents and to enrich them with semantically similar concepts.

A semantic parser analyzes the question and documents retrieved from the knowledge source for semantic structures. The semantic representations of phrases in the documents are compared to the question and answer candidates are extracted from similar structures. Answers obtained with this semantic approach are combined with candidates from syntactic extractors. Caching techniques are proposed to reduce the runtime of the most expensive processing steps.

Several ontologies have been combined in a modular framework for term extraction and expansion. The WordNet lexical database is utilized as a resource of general semantic knowledge, an interface for domain-specific ontologies allows integrating expert knowledge on restricted domains, and semantic networks built from a text corpus with semantic annotations are used to extract relations between entities and events in the corpus.

These resources have been integrated in the Ephyra framework for open-domain question answering to complement existing syntactic answer extraction techniques. Evaluation results show that the semantic techniques outperform the baseline system on questions that are amenable to a semantic analysis, and that a combination of syntactic and semantic techniques has a higher accuracy than any individual answer extraction approach.

1.4 Related Work

This section reviews related work on query expansion with ontologies (Section 1.4.2), semantic parsing approaches for question answering (Section 1.4.2), and score normalization and combination techniques for integrating answers retrieved with multiple extraction strategies (Section 1.4.3).

1.4.1 Query Expansions

[MN02] uses relations in the WordNet lexical database [Fel98] to automatically derive topically related terms and suggests their use to (1) expand query strings and increase the document retrieval recall and (2) recognize instances of correct answers that use different formulations than the question. Terms are considered semantically similar if they are linked through a lexical chain, a sequence of terms that are related in WordNet. For each of the WordNet relations, a weight has been determined empirically that reflects the degree of similarity between related words. The weight of a related term is calculated as the product of the weights of all relations on the lexical chain that links it to the original word. This thesis adopts the idea of lexical chains and reuses the proposed weights as confidence scores for related words.

The eXtended WordNet [MM01] provides additional semantic relations for query expansion. WordNet groups words with similar meanings in so-called synsets and

complements these groups with definitional glosses to clarify their meanings. In the eXtended WordNet, the terms in the glosses were semantically disambiguated and two additional relations were introduced: (1) A gloss link from a synset S_1 to a synset S_2 indicates that a word from S_2 occurs in the gloss for S_1 and (2) a reverse gloss link from S_1 to S_2 means that a word from S_1 appears in the gloss for S_2. [MN02] and [SJT+05] exploit these links as additional relations to derive related words for query expansion.

1.4.2 Semantic QA Systems

The JAVELIN open-domain question answering system [NFM+05] used semantic parsing techniques to answer relational questions in the TREC 14 evaluation. The questions were annotated with multiple layers of semantic information such as named entity types and predicate-argument structures, and they were expanded with weighted alternative hypotheses. Answer sentences were extracted from the document collection and enriched with the same layers of semantic annotations. The target structures in the questions were matched with those in candidate sentences and the sentences were ranked according to their similarity to the question. Answers were composed from all candidate sentences with similarity scores that met a minimum threshold.

[NMF+05] describes how JAVELIN has been extended with domain semantics to answer questions in a restricted domain, using a corpus from the Center for Nonproliferation Studies (CNS) as a domain-specific knowledge resource. An ontology of frequent concepts in this domain and English expressions with domain-specific meanings was generated manually. The ontology was deployed to expand semantic structures in the question to improve the recall of the system on the relatively small document collection. The CNS ontology is reused in this thesis as one of multiple resources for term expansion.

The factoid QA system described in [SHTT06] uses the semantic role labeling tool ASSERT to extract predicate-argument structures from questions and relevant sentences found in the corpus. Factoid answers are extracted from corpus sentences by identifying the arguments in the semantic structures of the sentences that are missing in the question. The answer extraction approach presented in this thesis also makes use of a semantic role labeling system to extract semantic structures from the corpus that are similar to the question. In addition, answer type information is exploited to extract answer candidates more reliably and ontological knowledge from various semantic resources is used to expand question terms and improve the recall of the document retrieval and answer extraction stages.

The QA system from the National University of Singapore [SJT+05] applied ASSERT to answer factoid and list questions in TREC 14. The target of each question series was used to fetch relevant Web documents from Answers.com, or if the search failed, from Google. The predicate-argument structures in the question and the sentences from the Web documents were compared, using a predicate similarity metric composed of the similarity of the predicate verbs and the similarity of the arguments. WordNet and eXtended WordNet were utilized to identify related verbs and to calculate similarity scores, adopting the approach introduced in [MN02]. Argument similarity scores were computed by forming the Jaccard coefficient of the sets of terms extracted from the arguments of the two predicates. The semantic roles

of the arguments were not matched for two reasons: (1) The authors found that ASSERT often fails to label the arguments correctly and (2) the semantic roles may be different for the verb in the question and related verbs. The similarity measure for predicates has been further refined in this thesis to provide a higher flexibility and improve the recall of the answer extraction. Multiple ontologies are combined to expand the terms in the arguments, and the semantic similarity of the arguments is measured in addition to their syntactic similarity.

1.4.3 Score Normalization and Combination

Various score normalization and combination schemes have been proposed by the document retrieval community and are used by metasearch engines to make document relevance scores from different retrieval approaches comparable and to combine search results from multiple systems. [MA01] suggests simple score normalization techniques that rescale the relevance scores from each approach to a fixed interval. [FS93] describes a family of score combination schemes denoted *Comb{MIN, MAX, SUM, ...}*. *CombSUM*, for instance, aggregates confidence scores by forming the sum of the scores from all systems that found a document. Some of these techniques are applied in this thesis to merge answer candidates found with semantic and syntactic extraction approaches.

Question answering systems frequently use statistical models to estimate comparable confidence scores for answer candidates. [IFR01] and [MLS+07] trained maximum entropy classifiers using features such as the document rank from the IR system, the similarity of semantic structures in the question and the answer sentence, the number of question terms that cooccur with the answer and the proximity of these terms to the answer. This thesis also makes use of a maximum entropy classifier to normalize confidence scores and compares it to other learning techniques such as adaptive boosting and decision trees.

[KSN07] proposes an approach for selecting answer candidates from multiple extractors that is not solely based on confidence scores but that also takes syntactic and semantic properties of the answers into account. A probabilistic framework estimates confidence scores for answer candidates using two types of features:

- *Answer validation features* are used to verify answer candidates. They deploy external semantic resources and are either knowledge-driven (e.g. gazetteers) or data-driven (e.g. Wikipedia).
- *Answer similarity features* exploit redundancy among the answer candidates, using both syntactic and semantic similarity measures. Examples for syntactic similarity features are the Levenshtein distance or the Jaccard coefficient. WordNet synsets can be used to identify semantically similar answers.

1.5 Outline

The thesis is organized as follows:

- Chapter 1 introduced the discipline of question answering. It motivated the use of semantic techniques, reviewed related work on semantic question answering and pointed out contributions of this thesis.

- Chapter 2 discusses a common pipeline architecture for question answering and describes how it is realized in the Ephyra system, which is used as a framework for the experiments in this thesis. Furthermore, the chapter gives an introduction to semantic role labeling and reviews notions and algorithms from the fields of statistical modeling and graph theory.

- Chapter 3 presents an approach for factoid questions that makes use of a semantic parser to identify relevant semantic structures in the corpus and to extract factoid answers from these phrases. It concludes with a discussion of relevance score normalization and combination techniques that allow integrating answers obtained with different extractors.

- Chapter 4 describes how ontologies are used to extract key terms and to enrich these terms with alternative syntactic representations. WordNet, a framework for domain-specific ontologies and semantic networks that represent relations between entities and events in a corpus are discussed in detail.

- Chapter 5 gives evaluation results for these semantic question analysis and answer extraction techniques and compares them to a baseline system with syntactic extractors.

- Chapter 6 concludes the thesis with a summary and directions for future research on semantic questions answering.

2. Fundamentals

This chapter describes a common architecture for question answering that has been implemented in many recent systems and illustrates it at the example of the Ephyra QA system, which served as a framework for the experiments presented in this thesis. Furthermore, it gives an introduction to semantic role labeling, a form of shallow semantic parsing that is utilized in Chapter 3 to extract semantic structures from questions and corpus documents and to identify phrases in the corpus that are semantically similar to the question. I also review statistical modeling techniques that have successfully been applied in QA and that can be used to normalize confidence scores of answer candidates (Section 3.5.2). The last section gives definitions of graph-theoretical concepts that provide the basis for Section 4.4 on semantic networks for term expansion.

2.1 QA Pipeline

A question answering system can be implemented as a pipeline of components for question analysis, query generation, search, answer extraction and answer selection. Most recent systems in principle follow this architecture, although the individual stages are often not clearly separated. For instance, the JAVELIN multilingual QA system [NFM+05, MLS+07] combines the question analysis and query generation stages in a single component, while Ephyra, described in the next section, uses a common pipeline for answer extraction and selection.

The *question analysis* component extracts syntactic and semantic information from the question, using techniques such as syntactic parsing, answer type analysis and named entity recognition. In the *query generation* stage, this information is transformed into a set of queries, which are passed to the *information retrieval* component. Depending on the type of the question, the IR component searches one or more resources that are appropriate for that type and aggregates the results. The search results are then passed to the *answer extraction* component, which extracts answer candidates of the desired granularity (usually factoid answers or definitional phrases). Finally, the *answer selection* component filters, combines and rearranges the candidates and returns a ranked list of answers. This architecture is illustrated in Figure 2.1.

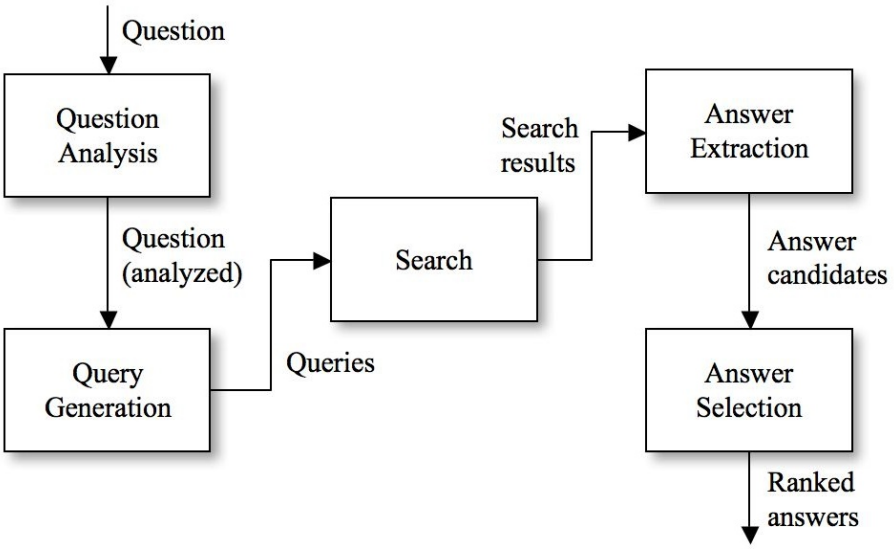

Figure 2.1: Common architecture of QA systems.

QA systems commonly rely on existing information retrieval engines to browse a document collection or the Web for relevant documents. Thus the question analysis and query generation components can be understood as preprocessing stages that transform user questions into queries that are supported by the underlying IR system. The answer extraction and selection components focus on the extraction of precise, to-the-point answers from potentially relevant documents.

2.2 Ephyra QA System

The Ephyra question answering system is a flexible and extensible framework for open-domain QA. Ephyra integrates several techniques for question analysis, query generation and answer extraction, and it can extract answers from both structured and semi-structured knowledge sources. Individual approaches often suffer from a low precision or they are limited in the types of questions they can answer. Thus the design goal was to combine different, complementary techniques to improve the performance of the overall system.

This section gives an overview of the Ephyra QA system, which serves as a framework and as well as a baseline for the experiments conducted in this thesis. An in-depth discussion of the underlying architecture and extraction strategies can be found in [Sch05, SGSW06, SGS06].

2.2.1 Framework

Ephyra is organized as a pipeline of reusable components for query generation, search, and answer extraction and selection. The components can be combined arbitrarily to adapt the framework to varying requirements. In this way, one can evaluate different setups and combine multiple QA techniques and knowledge sources that best fit the requirements. The system can be extended with additional techniques for question analysis, answer extraction and selection or a new knowledge source by plugging in additional components.

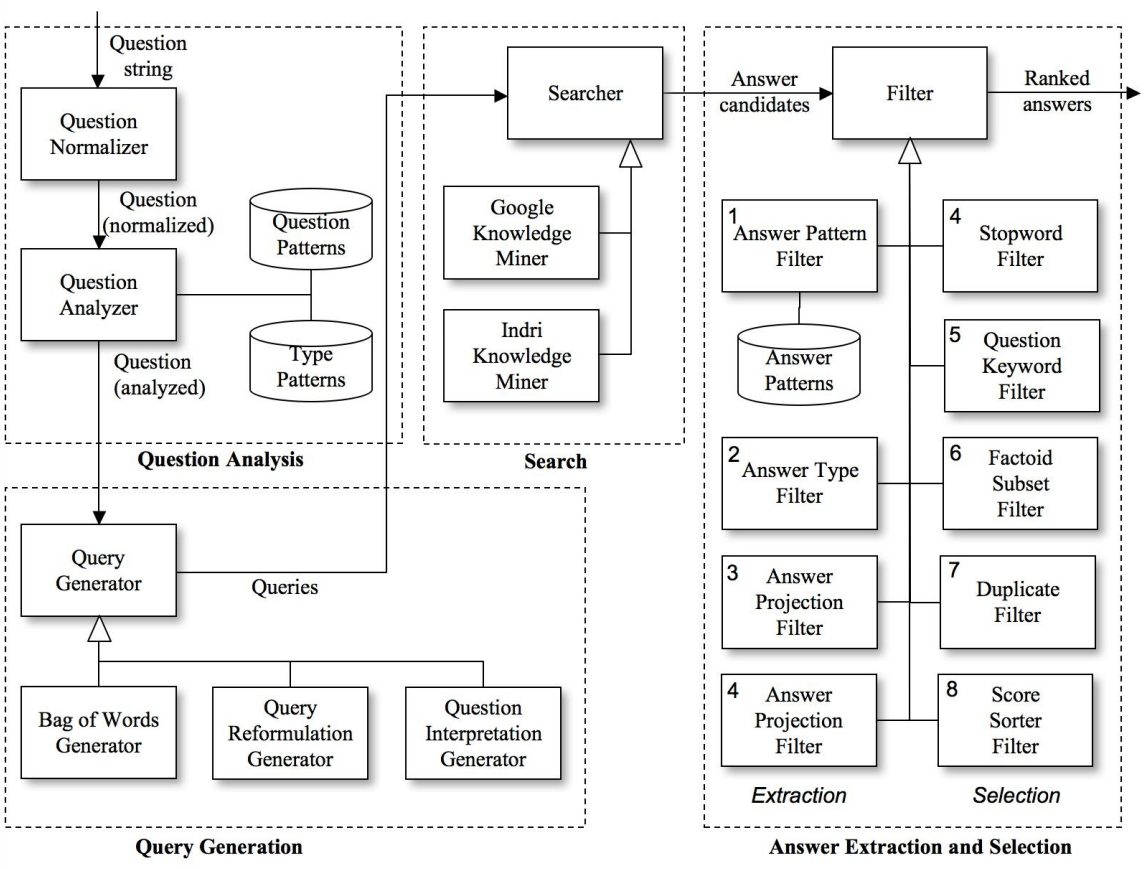

Figure 2.2: Pipeline for factoid and list questions.

Figure 2.2 shows the configuration of Ephyra that was used as a baseline for experiments with semantic techniques. The configuration is similar to our setup for the TREC 15 evaluation, but I have improved the capability of the answer selection pipeline to detect similar answers and non-responsive answers. In this thesis, I have extended the baseline system with semantic question analysis, query formation and answer extraction techniques. Note that the figure shows the setup for factoid and list questions. Ephyra is also capable of answering definitional questions, but this thesis deals with factoid questions only.

At first the question string is normalized, e.g. tenses are modified and verbs and nouns are replaced by their lemmas. The question analyzer then parses the question string, it determines the expected answer type, extracts named entities and other terms and creates a concise interpretation of the question consisting of its key phrases and a more specific answer type. Given this information, the query generators build several queries for document retrieval. The baseline system uses ”bag of words” queries, which are simple sets of keywords, as well as more specific queries such as reformulations of the question string and key phrases.

Ephyra supports two classes of search component to integrate external knowledge sources. *Knowledge miners* incorporate unstructured resources using an underlying document retrieval system, such as a Web search engine. *Knowledge annotators* extract information from semi-structured resources such as an online encyclopedia or a web service for geographical information.

NE type	Type pattern
Date	when
Date → Season	(what|which) (.*)?(season|time of (the)?year)
Date → Weekday	(what|which) (.*)?(day of (the)?week|weekday)
Location	where
Location → City	(what|which|name) (.*)?(city|metropolis|town|village)
Location → Country	(what|which|name) (.*)?(colony|country|nation)
Size	how (big|large)
Size → Area	(how large in|how many) (acre|square (foot|.*meter))
Size → Length	how (deep|far|high|long|tall|wide)
Size → Length	(how large in|how many) (foot|inch|.*meter|mile|yard)
Size → Volume	(how large in|how many) (gallon|liter|ounce)

Table 2.1: NE hierarchy and type patterns.

In the answer extraction and selection component, the search results are passed through a pipeline of filters to produce the final list of ranked answers. A filter can drop unresponsive answers (function words, duplicate answers, answers that merely repeat part of the question etc.), it can form new answers from existing ones (e.g. by extracting factoid answers from sentence-level answer candidates), and it can rearrange the answers according to features such as confidence scores from the answer extractors or hit positions from the document retrieval systems.

2.2.2 Answer Extraction Techniques

For factoid and list questions, Ephyra combines a simple extraction approach based on answer type classification with a pattern learning and matching approach. The first technique, described in Section 2.2.2.1, determines the expected answer type of a question by mapping it to a hierarchy of named entity (NE) types and extracts entities of the expected type as answer candidates, using an appropriate NE tagger. This approach has a high precision, but it can only be applied to a question if the answer type appears in the NE hierarchy and a suitable NE tagger is available. Thus it has been complemented with a second approach that uses textual patterns to analyze and categorize questions and to extract factoid answer candidates from text passages without necessarily knowing their type. This approach is discussed in Section 2.2.2.2.

2.2.2.1 Answer Type Analysis

This answer extractor is based on a hierarchy of frequent NE types which resulted from an analysis of past TREC questions. The hierarchy currently comprises about 120 types such as *Date*, *Food*, *Location*, *Number*, *Person* or *Size* on the top level and *City*, *Country* or *State* as subtypes of *Location*. Each of the types is associated with one or more type patterns, i.e. regular expressions that match questions asking for entities of that type. Table 2.1 shows an extract of the NE hierarchy and associated type patterns.

The question analysis component compares the question string against each of the type patterns to determine potential answer types. A number of tie-braking rules are used to select an answer type for questions that match more than one pattern.

For instance, longer patterns are preferred over shorter ones since they are more specific and a false match is less likely.

In the answer extraction phase, the Answer Type Filter (cf. Figure 2.2) uses NE taggers for the expected type to extract answer candidates from text snippets retrieved from the knowledge base. NEs of type *Person*, *Organization* and *Location* are extracted with the OpenNLP and Stanford NE recognizers ([ONL] and [FGM05]). For all other types, Ephyra uses proprietary taggers which are either rule-based (e.g. for the types *Number*, *Size*) or list-based (e.g. for *Food*, *Country*). The extractor first looks for entities of the most specific answer type and moves upwards in the type hierarchy if no entities could be tagged. For instance, if the question asks for a country name, it first applies the tagger for the answer type *Country*, followed by the tagger for *Location* if the first tagger did not find any entities. The answer candidates are normalized (i.e. tokenized and stemmed using a Porter stemmer [Por80]) and similar answers are merged. The confidence score of an answer is the number of answer candidates with the same normalized formed.

2.2.2.2 Pattern Learning and Matching

The pattern matching approach uses two types of textual patterns to (1) analyze and interpret questions and (2) extract factoid answer candidates from text passages retrieved from the knowledge source. The interpretation of a question is a concise representation that abstracts from the original question string while preserving its meaning. It can be transformed into a largely formulation-independent query for document retrieval. While the patterns for question analysis have been defined manually, the answer extraction patterns can be learned automatically from sample questions and answers.

The approach is based on the assumption that a question is fully specified by the following three components: A question asks for a *property* of a *target* in a specific *context*. For instance, the question "What is the job of Mel Gibson in Conspiracy Theory?" asks for the *profession* (property) of *Mel Gibson* (target) in the movie *Conspiracy Theory* (context) [1]. During question analysis, a set of handcrafted question patterns are applied to the question string to extract the target and context and to identify the property the question is asking for. The Question Interpretation Generator (cf. Figure 2.2) transforms the interpretation into a query string, consisting of the target and context objects, and passes the query to the search component to retrieve relevant text passages from the Web or a local text corpus.

In the answer extraction phase, a second set of patterns is used to extract answer candidates from text passages that contain both the target of the question and the desired property, in the above example the person's name *Mel Gibson* and the profession *taxi driver*. For instance, an answer pattern for the property *profession* may look as follows:

```
<Target> is employed as a <Property>.
```

Ephyra automatically learns the answer patterns using question-answer pairs as training data, which can be obtained from past TREC evaluations. Each training

[1] Example taken from [SGS06].

Original pattern:

```
<Target> , who died in Salzburg in <Property> and
```

Generic pattern:

```
<Target> [^<]*died [^<]*(<Location> )?[^<]*<Property_Date>
```

Illustration:

We are looking for a date. The string in between the person's name and that date must include the keyword "died", and it may optionally contain a named entity of type location. In addition, it may comprise any other tokens but no further named entities.

Figure 2.3: Answer pattern for the property *day of death*, its generalization and an interpretation in natural language.

question is interpreted and a query string consisting of the interpretation and a correct answer is formed. The query string is used to fetch text snippets from the Web that contain both the target of the question and the answer, using the Google and Yahoo search engines. From these text snippets, Ephyra extracts answer patterns that relate a target to a specific property, in the above example patterns that connect working people and their jobs. An answer pattern consists of tags indicating the target and its property, an arbitrary string in between the two tags plus one token preceding or following the property tag to indicate where the property starts or ends.

The answer patterns returned by the learning algorithm can be very long and they can contain named entities, thus they are often too specific. Therefore, to improve the recall of the answer extraction, the patterns are transformed into more generic patterns. Fist of all, named entities in the patterns are replaced by their types. Whenever possible, the property is also assigned a NE type for the following two reasons: (1) If the type of the property is known, the tokens adjacent to the property are not required as delimiters, thus they can be dropped to obtain a less specific pattern. (2) The pattern can be restricted to only extract entities of the desired type. Furthermore, tokens which are not NEs are substituted by placeholders that match any sequence of tokens other than NEs. The only exception are key tokens that are important for the relationship between the target and the answer, which are maintained. A token is considered important if it appears in a question pattern for the respective property, regardless of its grammatical form (e.g. the verb "employed" in patterns for the property *profession*). Figure 2.3 illustrates how a generic pattern is derived from a specific pattern and describes its meaning in natural language. The pattern extracts the *day of death* of a person.

The generic answer patterns are evaluated on a separate test set and patterns with a low precision or recall are dropped, the remaining patterns are assigned confidence scores that reflect their precision. The scores of the extracted answer candidates are calculated by cumulating the confidence measures of the answer patterns used to extract them (see [SGSW06]).

	Ephyra	Minimum	Median	Top
Unsupported (U)	18	18	11	N/A
Inexact (X)	26	26	23	N/A
Locally correct (L)	2	2	2	N/A
Factoid accuracy	0.196	0.196	0.196	0.186
List F_1	0.092	0.096	0.097	0.087
Other F_3	0.143	0.150	0.145	0.125
Average per-series score	0.139	0.143	0.141	0.134

Table 2.2: TREC15 results.

2.2.3 Evaluation Results

The Text REtrieval Conference (TREC) conducts an annual evaluation of QA systems [Voo02, DLK06]. We participated in the TREC evaluation in 2006 for the first time, deploying the framework and the answer extraction techniques described in the preceding sections. Table 2.2 shows the evaluation results for our best run, along with the minimum, median and top scores among all 27 participants. The configuration of Ephyra used in the TREC evaluation served as a baseline for the semantic question analysis and answer extraction experiments presented in this thesis.

2.3 Semantic Role Labeling

Semantic role labeling (SRL) is a form of shallow semantic parsing that deals with the extraction of predicate-argument structures from natural language sentences. Question answering systems can apply SRL to identify semantic structures in the knowledge base that are similar to structures in the question and therefore potentially provide answers to the question (see Chapter 3).

The semantic structures extracted by an SRL system are of the form *who* does *what* to *whom*, *when*, *where*, *why*, *how* etc. The predicate (*what*) is a verb, usually describing an event, while the arguments (*who*, *when*, *where* etc.) are the entities involved in that event. Each of the arguments is associated with one of the semantic roles listed in Table 2.3. There are arguments with verb-specific roles (ARG0 to ARG5) as well as modifiers with fixed meanings (starting with ARGM). For example, in the predicate *send* ARG0 refers to the *sender* and ARG2 to the *recipient* whereas in the predicate *fall* ARG0 is the *thing falling* and ARG1 the *start point*. The modifier ARGM-LOC, in contrast, always refers to the location of an event.

The Proposition Bank (in short PropBank) corpus is a collection of newswire articles that has been enriched with predicate-argument structures [KP02, KP03]. It was created at the University of Pennsylvania and comprises the Wall Street Journal part of the Penn TreeBank corpus. The goal was to provide the research community with a substantial amount of training data that can be used for the development of semantic role labeling tools.

In 2005, the Conference on Computational Natural Language Learning (CoNLL) conducted an evaluation of statistical SRL systems [CM05]. PropBank was used as a common development set by all participants to ensure fairness. The 19 participants used a wide range of learning techniques and features. However, the four best systems had in common that they combined several individual SRL systems, each relying on

Semantic Roles	Interpretations	Examples
ARG0 - ARG5	verb-specific roles (often ARG0 agent, ARG1 patient/theme)	it, her
ARGM-ADV	sentence-level adverb, or general purpose	unfortunately
ARGM-CAU	cause	because of him
ARGM-DIR	direction	down
ARGM-DIS	discourse connective	furthermore, on the other hand
ARGM-EXT	extent	much, by 1 million
ARGM-LOC	location	in Pittsburgh
ARGM-MNR	manner	quickly
ARGM-MOD	modal verb	will, might
ARGM-NEG	negation	not, n't
ARGM-PNC	purpose	for money
ARGM-TMP	time	tomorrow

Table 2.3: Semantic roles, their meanings and illustrating examples.

different syntactic structures, to improve the robustness and coverage. The top system [KPRY05] achieved an F1 score of about 79%. In real applications such as QA, the performance can be expected to be lower as documents differ in their style and domain from Wall Street Journal articles. This shows that semantic role labeling is still an error-prone task. SRL systems often fail to correctly recognize argument boundaries and, more frequently, do not assign the correct semantic roles to the arguments. A question answering system that utilizes SRL for answer extraction has to be robust to these types of errors.

For the experiments presented in this thesis, I used ASSERT [PWH+04], a state-of-the-art semantic role labeling tool that has been deployed in several QA systems. ASSERT is one of the most accurate systems that are publicly available, but it has the disadvantage of not revealing any intermediate results such as the word sense assigned to a predicate. An open architecture for semantic parsing is proposed in [EP06]. The parser is organized as a modular toolchain, consisting of independent modules for syntactic analysis, word sense disambiguation and semantic role assignment. Individual modules can be replaced, and the results from each module can be reused. However, the performance of this toolkit is not yet on a par with ASSERT, and therefore I refrained from using it for my experiments.

2.4 Statistical Modeling

Statistical modeling deals with the prediction of the behavior of a random process. Given sample input and output from the process, the goal is to create a representation of the process that can be used to predict its behavior on unseen data. The modeling problem can be composed into two subproblems: (1) extract relevant facts from the sample data and (2) create a model that adheres to these facts.

In question answering, one can find many instances of this problem. For example, the prediction of the answer type of a factoid question can be viewed as a random

process that, given a set of features such as the parse tree of the question and focus terms, outputs the expected answer type. Statistical modeling can also be used to score answer candidates, or to normalize the confidence scores of answer candidates obtained with different extraction techniques to make them comparable. The latter will be dealt with in Section 3.5.2.

In this section, I give an overview of two statistical modeling techniques that have been applied successfully in question answering: maximum entropy (Section 2.4.1) and adaptive boosting (Section 2.4.2).

2.4.1 Maximum Entropy

Maximum entropy (in short MaxEnt) models have been used in several question answering systems to estimate confidence scores for answer candidates [IFR01, MLS+07]. They form a class of distributions within the family of exponential models that adhere to the *maximum entropy principle*. This principle states that among all distributions that conform to a set of features extracted from training samples, one should choose the least biased distribution. It dates as far back as 1956, when E.T. Jaynes initiated the maximum entropy thermodynamics. An introduction to maximum entropy modeling with an emphasis on natural language processing (NLP) is given in Berger's tutorial [Ber96]. This section is meant as a brief motivation of the MaxEnt principle.

Given a random process that produces output values $y \in Y$ from input $x \in X$, the task is to model the *conditional probability distribution* $p(y|x)$. The behavior of the process is described by a set of *training samples* $(x_1, y_1), ..., (x_n, y_n)$ that define the *empirical probability distribution*

$$\tilde{p}(x, y) := \frac{1}{n} \times \text{ number occurrences of } (x, y) \text{ in the sample.}$$

A *feature* is a binary function $f(x, y)$ that assumes the value 1 for some combinations of input and output values. The *expected value* of a feature f is defined with respect to the empirical probability distribution $\tilde{p}(x, y)$ (2.1) and with respect to the model $p(y|x)$ (2.2):

$$\tilde{p}(f) := \sum_{x,y} f(x, y)\tilde{p}(x, y) \tag{2.1}$$

$$p(f) := \sum_{x,y} f(x, y)\tilde{p}(x)p(y|x) \tag{2.2}$$

In general, features should be selected to (1) cover the training sample as closely as possible but (2) a feature f should only be used if its expected value $\tilde{p}(f)$ can be estimated reliably from the sample.

Now one can impose a *constraint* on the set of allowable models by requiring the two expected values of a feature f to be the same:

$$\tilde{p}(f) \stackrel{!}{=} p(f)$$

Given a set of constraints, the maximum entropy principle provides a guideline for choosing a model $p(y|x)$. It suggests the following strategy for model selection:

1. We only consider models that satisfy all constraints.
2. Among these models we select the model that is most uniform.

The most uniform model is the one that maximizes the *conditional entropy*

$$H(p) := -\sum_{x,y} \tilde{p}(x)p(y|x)\log p(y|x).$$

This model is well-defined, i.e. there is a unique model p^* such that

$$p^* = \operatorname*{argmax}_p H(p).$$

The maximum entropy principle is plausible since it suggests that we should model all that we know from the training data (1.) but that we should not make any assumptions about what we do not know (2.). More formally, it can be shown that the model with maximum entropy is also the model that maximizes the likelihood of the training sample (the dual problem, cf. [Ber96]), which provides additional justification for the MaxEnt principle.

2.4.2 Adaptive Boosting

Adaptive boosting (in short AdaBoost) is an iterative algorithm for improving the accuracy of a weak classifier that was formulated in 1995 by Freud and Schapire [FS96]. An introduction to AdaBoost and sample applications are given in a later paper from Freud and Schapire [FS99].

The AdaBoost algorithm calls the *weak* or *base learning algorithm* in a series of rounds. It repeatedly uses the same training data, but each training sample is associated with a weight which is updated after each round. Initially all samples are assigned equal weights. After each iteration, the hypothesis of the weak classifier is tested on the training set, and the weights of incorrectly classified samples are increased while the weights of the remaining examples are decreased. In this way, the weak learner focuses more and more on the hard samples which it could not classify correctly previously. The final hypothesis is a weighted majority vote of the hypotheses from all rounds. The weight of a hypothesis depends on its error and is the larger the more examples have been classified correctly. Figure 2.4 shows pseudocode for the AdaBoost algorithm.

In [FS96] it is shown that the error of the final hypothesis on the training data has an upper bound of

$$\exp\left(-2\sum_{r=1}^{R}\left(\frac{1}{2}-\epsilon_r\right)^2\right)$$

where R is the number of rounds and ϵ_r is the error of the hypothesis in round r. Consequently, assuming that each hypothesis is better than random guessing (i.e. $\epsilon_r \leq \epsilon$ for some $\epsilon < \frac{1}{2}$), the training error can be made arbitrarily small by increasing R, and it decreases exponentially fast. In contrast to earlier boosting approaches,

Input: Training samples $(x_1, y_1), ..., (x_n, y_n)$ where $y_1, ...y_n \in \{-1, 1\}$,
Number of rounds R.

Algorithm:

- For $i = 1, ..., n$: Initialize weights $D_1(i) := \frac{1}{n}$.
- For $r = 1, ..., R$:
  - Train weak learner using weights D_r.
  - Get weak hypothesis h_r and compute error $\epsilon_r := \sum_{i:h_r(x_i) \neq y_i} D_r(i)$.
  - Compute importance of hypothesis $\alpha_r := \frac{1}{2} \ln \left(\frac{1-\epsilon_r}{\epsilon_r} \right)$.
  - Update weights

$$D_{r+1}(i) := \frac{D_r(i)}{Z_r} \times \begin{cases} e^{-\alpha_r} & \text{if } h_r(x_i) = y_i \\ e^{\alpha_r} & \text{if } h_r(x_i) \neq y_i \end{cases}$$

where Z_r is a normalization factor such that D_{r+1} is a distribution:

$$\sum_{i=1}^{n} D_{r+1}(i) \stackrel{!}{=} 1.$$

Output: Combined hypothesis

$$H(x) := \text{sign} \left(\sum_{r=1}^{R} \alpha_r h_r(x) \right).$$

Figure 2.4: AdaBoost algorithm in pseudocode.

the *adaptive* boosting algorithm does not need to know the threshold ϵ in advance, but it adapts to the error rates of the hypotheses.

The algorithm given in Figure 2.4 returns binary predictions but it can be modified to provide real-valued confidences. In Section 3.5.2 on score normalization, a generalization of AdaBoost is used that is based on confidence-rated predictions [SS99]. The weak learner deployed in the implementation is a decision tree with a maximum depth of 5.

AdaBoost has the advantages of being easy to implement and fast. It does not require any knowledge about the base learner and it can be combined with any weak learner as long as this learner is better than random guessing. In general, AdaBoost is also little susceptible to overfitting. In the application presented in Section 3.5.2, however, it does overfit, which is likely due to an adaptation to outliers. AdaBoost assigns increasingly large weights to outliers which can result in an over-adaptation if R is too large.

2.5 Graph Theory

This section introduces graph-theoretical concepts and algorithms that form the basis of Section 4.4 on semantic networks. A semantic network is an undirected graph that can be built from semantic annotations of a textual corpus and that represents relations between entities and events in the corpus.

2.5.1 Definitions

A *directed graph* G is an ordered pair (V, E) where V is a set of vertices and E is a set of directed edges between the vertices, i.e. $\forall e \in E : e = (u, v) \in V \times V$. In an *undirected graph*, the edges are undirected and can be denoted as binary sets $\{u, v\}$ of vertices $u, v \in V$.

A *path* $p = (u, v)$ from vertex u to vertex v in a directed graph $G = (V, E)$ is a sequence of vertices $v_0, v_1, ..., v_k \in V$ such that $u = v_0$, $v = v_k$ and $(v_{i-1}, v_i) \in E$ for $i = 1, 2, ..., k$. In an undirected graph, a path is a sequence of vertices that are connected by undirected edges $\{v_{i-1}, v_i\}$. A path is *simple* if its vertices are pairwise distinct. The *length* of a path is the number of edges k. A path $p = v_0, v_1, ..., v_k$ is a *cycle* if $k \geq 1$, $v_0 = v_k$ and the prefix $v_0, v_1, ..., v_{k-1}$ is simple. If $v_0 \neq v_k$ then p is an *open* path, v_0 is its *head* and v_k its *tail*.

An undirected graph $G = (V, E)$ is *connected* if there is a path between each pair of vertices $(u, v) \in V \times V$. A directed graph is *strongly connected* if for any two vertices $u, v \in V$ there is a path $p_1 = (u, v)$ from u to v and a path $p_2 = (v, u)$ from v to u. In this thesis, I refer to an undirected graph as strongly connected if there are two distinct, simple paths between any two vertices $u, v \in V$. This implies that all vertices are on cycles.

A *tree* is a connected graph $G = (V, E)$ that does not contain any cycles, i.e. there is exactly one path between any two vertices $u, v \in V$. In a *forest*, any two vertices are connected by at most one path, i.e. a forest is cycle-free but in general not connected. The *degree* of a vertex is the number of adjacent edges. A *leaf* is a vertex of degree 1.

2.5.2 Cycles

There are two classes of algorithms for finding all cycles in a graph. The first class of algorithms exploit the observation that the cycles in an arbitrary graph $G = (V, E)$ induce a vector space over the field $GF(2)$. Every cycle can be represented as a bit-vector of length $|E|$, where each element indicates whether a particular edge is part of the cycle. The addition $\oplus$ and multiplication $\otimes$ in the cycle vector space are induced by the respective operations in $GF(2)$, e.g. the sum of two cycles is the set of all edges that are part of exactly one of the cycles. The dimension of the cycle vector space is $|E| - |V| + C(G)$ where $C(G)$ is the number of connected components of G. A cycle basis can be constructed in polynomial time, e.g. with the well-known algorithm of Horton [Hor87], which has a runtime of $O(|E|^3|V|)$. All cycles can be obtained as linear combination of the cycles in a cycle basis.

The second class subsumes search algorithms, which are usually more efficient since they only consider a fraction of all possible subsets of E. [LW06] introduces a simple search algorithm that enumerates all cycles in a directed graph. It maintains a queue

Input: Directed graph $G = (V, E)$.

Algorithm:

- Put all vertices $v_1, ...v_n \in V$ into queue Q, create empty set of cycles C.
- While Q is not empty:
  - Remove open path P from queue Q.
  - Set $k := length(P)$, $v_h := head(P)$, $v_t := tail(P)$.
  - If $(v_t, v_h) \in E$:
    - Add open path P to set of cycles C.
    - Continue with next open path.
  - $\forall\ (v_t, v_x) \in E$: $x > h$ and $v_x \notin P$:
    - Put new open path $P + v_x$ into queue Q.

Output: Set of all cycles C.

Figure 2.5: Simple algorithm that finds all cycles in a directed graph.

of open paths, which initially contains all vertices in the graph. The algorithm repeatedly removes an open path from the queue and checks if its head and tail vertices are connected by an edge. If it finds a cycle, the open path is discarded and the cycle is added to the output set. Otherwise, the algorithm extends the open path by adding vertices that are adjacent to its tail and appends the newly constructed paths to the queue. To avoid duplicates, an order is imposed on the vertices and a vertex is added to a path only if its order is larger than the order of the head vertex. Figure 2.5 shows pseudocode for the algorithm.

This search algorithm is applicable to directed graphs and, after minor modifications (see Section 4.4.2), also to undirected graphs. It is easy to implement and it can find all cycles of up to a given length without taking any longer paths into account. On the other hand, its time and space requirements are exponential, while the fastest known algorithms have a runtime that is polynomial in the size of the graph and the number of cycles. Johnson's algorithm [Joh75], for instance, uses backtracking and pruning techniques to reduce the time complexity to $O((|V| + |E|)(c + 1))$ and the space complexity to $O(|V| + |E|)$ where c is the number of cycles in the graph. However, Section 4.4 only deals with short cycles of length 2 or 3 in large but relatively sparse networks, and the simple algorithm presented in this section is sufficiently fast for that purpose.

2.5.3 Strongly Connected Components

An algorithm that efficiently finds all strongly connected components in a directed graph $G = (V, E)$ was introduced by S.R. Kosaraju in an unpublished paper in 1978. The algorithm uses two consecutive depth-first searches (DFS): At first it performs a DFS on the original graph G, then it reverses the order of all edges in E and performs a second DFS on the reversed graph. The sequence of completion of the recursive calls in the first DFS induces an order on the vertices. The second

Input: Directed graph $G = (V, E)$.

Algorithm:

- Perform a DFS on G and number the vertices according to the order of completion of the recursive calls.
- Construct a new graph $G_r = (V, E_r)$ where $E_r = \{(u, v)|(v, u) \in E\}$, i.e. the direction of the edges is reversed.
- Perform a DFS on G_r, starting from the vertex with the largest order according to the first step. If the DFS does not reach all vertices then resume the search in the vertex with the highest order that has not been visited yet.
- Add each tree in the resulting forest to the set of strongly connected components S.

Output: Set S of strongly connected components in G.

Figure 2.6: Kosaraju's algorithm for finding all strongly connected components in a directed graph.

DFS is started in the last vertex according to that order and, if the search does not reach all vertices, it is resumed in the last remaining vertex. The trees that result from the second DFS are the strongly connected components of G. Pseudocode for Kosaraju's algorithm is given in Figure 2.6. Section 4.4.2 shows how the algorithm can be extended for undirected graphs.

3. Semantic Parsing for Factoid Questions

This chapter introduces an approach for question analysis and answer extraction that is based on semantic role labeling (cf. Section 2.3). The approach has been implemented within the Ephyra framework (see Section 2.2) and is used in conjunction with the existing syntactic answer extraction techniques.

Sections 3.1 to 3.4 give an outline of the question analysis, search, and answer extraction and selection stages of this semantic approach. At first, a question is analyzed for predicate-argument structures and transformed into a semantic representation. Ephyra forms search engine queries from this representation and retrieves relevant documents from either a static text corpus or the Web. In the answer extraction phase, sentences are extracted from the documents and undergo the same semantic analysis. The semantic representations of the sentences are compared to the question to identify candidates that contain similar semantic structures. Ephyra extracts factoid answers from these sentences either by looking for arguments with a specific semantic role or by searching for entities of the expected answer type. Finally, unresponsive answers are dropped and similar answers are merged in a final list of ranked answers. Ephyra has been extended to cache intermediate results in order to save runtime on the most expensive processing steps.

Section 3.5 further describes how answer candidates retrieved with the semantic approach can be combined with candidates from the previously described syntactic answer extractors (cf. Section 2.2.2). As the different extraction techniques use distinct scoring mechanisms that are incomparable, it is necessary to normalize scores before combining them in a single confidence measure.

3.1 Question Analysis

This approach is only applicable to questions with predicate-argument structures that can be recognized by the semantic role labeling (SRL) system. I used the semantic parser ASSERT (cf. Section 2.3) for my experiments, which is unable to label the predicates *to be*, *to do* and *to have* because their meaning depends on the

arguments and semantic knowledge is required to disambiguate them. Thus only questions that contain predicates other than those three can be analyzed. In certain cases it may be possible to transform these predicates into parsable predicates, but it is unfeasible in the general case. For example, the question "Who is the governor of California?" can be rephrased into "Who governs California?", but the question "Who is the current Pope?" does not suggest a simple transformation.

Moreover, the semantic parsing of questions is more error-prone than the parsing of declarative sentences because semantic parsers are hardly trained on questions. Therefore I chose to transform questions into statements first and apply the semantic parser to the statements. The interrogative pronoun is replaced with a placeholder argument that represents a possible answer to the question. The placeholder must be of the proper type to ensure that it is assigned the correct role by the semantic parser. This is particularly important for factoid questions as knowledge of the semantic role of the missing argument can be applied in the answer extraction stage.

I have derived a number of transformation rules from the questions of the TREC 15 evaluation [DLK06]. The rules use the following features to rephrase a question into a statement:

1. the interrogative pronoun
2. the type of adjacent phrases
3. the expected answer type

The phrase chunker from the OpenNLP toolkit [ONL] is used to extract noun, verb and prepositional phrases from the question. The expected answer type is determined by a classifier that maps the question to a hierarchy of frequent answer types described in Section 2.2.2.1.

Table 3.1 lists sample transformation rules that illustrate how the above features are applied. *PP* refers to a prepositional phrase, *NP* to a noun phrase and *JJ* to an adjective. Phrases denoted in square brackets are optional. For instance, by applying the second rule, the question

> *"In what year was the Carnegie Mellon campus at the west coast established?"*

can be transformed into the statement "*The Carnegie Mellon campus at the west coast was established on 1/1/1111*". This question will serve as a running example throughout the remainder of this chapter.

Note that these rules are not applied to the original question but to a normalized question string. Verb constructions with auxiliary verbs that are specific to questions are replaced by forms that are used in declarative sentences. For example, in the question "When did Shakespeare write Hamlet?" the verbs *did* and *write* are replaced by *wrote*. This normalization step is described in detail in the section on question normalization in [Sch05].

The semantic parser is applied to the resulting statement to extract the predicate verbs and arguments. The placeholder argument is dropped and the corresponding

Interrogative + Adjacent Phrases	**Answer Type**	**Transformation Rule**
when	ignore	drop interrogative pronoun, append placeholder "on 1/1/1111"
[PP +] what + NP	Date, Time or subtypes	drop [PP +] interrog. pronoun + NP, append placeholder "on 1/1/1111"
where	ignore	drop interrogative pronoun, append placeholder "in America"
[PP +] what + NP	Location or subtypes	drop [PP +] interrog. pronoun + NP, append placeholder "in America"
why	ignore	drop interrogative pronoun, append placeholder "for purpose"
how	ignore	drop interrogative pronoun, append placeholder "with manner"
[PP +] how + JJ	Duration	drop [PP +] interrog. pronoun + JJ, append placeholder "for one hour"
{name, give, tell, list} + NP + interrogative pronoun	ignore	drop {name, give, tell, list}, drop interrogative pronoun

Table 3.1: Question transformation rules based on interrogative pronouns, adjacent phrases and expected answer types.

semantic role is marked as "missing", indicating that this is the information the question is seeking. A question may contain multiple predicates, but in general only one with a missing argument. Questions without missing arguments cannot be handled and are left to one of the syntactic extraction techniques.

Finally, terms are extracted from the question and are associated with the predicate verb or argument they occur in. Terms are units of meaning and can consist of one or more tokens (e.g. "Carnegie Mellon", "west coast"). The terms are enriched with related concepts derived from different semantic resources as described in Chapter 4. For instance, "CMU" could be used as a short form for "Carnegie Mellon" and "founded" as a synonym of "established". Each of these concepts is assigned a confidence value reflecting its similarity to the original term. Terms provide a more fine-grained representation of the phrases in a predicate and are used for query generation and expansion (see Section 3.2) and to measure the similarity between question and answer predicates (cf. Section 3.3.2).

The semantic representation of the question in the running example ("In what year was the Carnegie Mellon campus at the west coast established?") is illustrated in Figure 3.1.

3.2 Query Generation and Search

The semantic representations need to be transformed into query strings that conform with the query languages of the underlying document retrieval systems. I used the Indri search engine from the Lemur toolkit [Lem] for my experiments on locally available text corpora and Google to search for web sites.

- TARGET: established
  - TERM: established (POS: VBN, NE Types: -)
    Aliases: founded (Weight 0.8), launched (Weight 0.7)
- ARG1: the Carnegie Mellon campus
  - TERM: Carnegie Mellon (POS: Compound, NE Types: Organization)
    Aliases: CMU (Weight 0.9)
  - TERM: campus (POS: NN, NE Types: -)
    Aliases: -
- ARGM_LOC: at the west coast
  - TERM: west coast (POS: Compound, NE Types: Location)
    Aliases: -
- ARGM_TMP: *missing*

Figure 3.1: Semantic representation of the question "In what year was the Carnegie Mellon campus at the west coast established?".

Indri provides a highly flexible query language that supports complex, structured queries [OC03, BOCN07]. An index can be build from a document collection in conjunction with offset annotations of syntactic or semantic structures, such as parse trees or named entities. For this thesis I had access to a text corpus from the Center for Nonproliferation Studies (further referred to as CNS corpus, see Section 5.2 for details) that had been annotated with ASSERT in the JAVELIN project [NMF+05]. An Indri index built from this corpus and annotations supports specific queries for semantic structures, e.g. one can search for all predicates that satisfy certain constrains on their arguments and semantic roles. For instance, the Indri query

```
#combine[sentence](
    #max(#combine[target](established
        #max(#combine[./ARG1](Carnegie Mellon campus))
        #max(#combine[./ARGM-LOC](west coast))
    ))
)
```

can be read as:

```
Find sentences that contain the predicate "established" with the
argument "Carnegie Mellon campus" and the location "west coast".
```

Structured queries can be used to conveniently search for sentences that match a given semantic pattern, or even elements within these sentences. This greatly facilitates the answer extraction stage of the QA pipeline (cf. Section 2.1) and improves the runtime performance, but it requires a locally available corpus that can be annotated before indexing.

In contrast, Web search engines such as Google and Yahoo only allow for simple queries that are boolean combinations of keywords or phrases. To emulate the

behavior of the above Indri query with a boolean query, one would have to search for documents that contain the verb and arguments, extract sentences from these documents, parse the sentences and identify those that contain a predicate with the specified arguments and roles. This is by far more time consuming since it requires parsing the sentences at runtime. On the other hand, the Web has the obvious advantage of being the largest and most redundant knowledge source available, and it is updated continuously.

Since the aim of this thesis is to propose semantic techniques for open domain question answering that are applicable to any text-based data resource including the Web, I decided to restrain from using structured queries in my experiments. While this limits the flexibility in the query generation phase, there remains a degree of freedom in choosing the granularity of the expressions in the query string. I implemented and evaluated the following types of queries:

- **Keyword queries** simply list all the keywords in a question. Function words and duplicates are dropped. I apply an implementation of the Porter stemmer [Por80] to normalize the keywords before searching for duplicates.

  Bags of keywords are very general and usually yield a high recall, but often do not preserve the semantics of the question.

  Example: *Carnegie Mellon campus west coast established*

- A **term query** is obtained by concatenating all terms that occur in the question. Terms can be single-token or multi-token expression (compound nouns and named entities). They are tokenized and compared at the level of keywords to identify duplicates.

  Term queries require a document to contain units of meaning as a whole. Therefore, they are more specific than keyword queries and in general result in a lower recall while being more precise. The terms can be enriched with alternative representations (see Chapter 4) to improve the recall.

  Examples:
  - *"Carnegie Mellon" campus "west coast" established*
  - *("Carnegie Mellon" OR CMU) campus "west coast" (established OR founded OR launched)*

- **Predicate queries** are concatenations of the predicate verbs and arguments.

  These queries require an exact match for each of the arguments and thus yield the highest precision but the lowest recall. They can be expanded by expanding the individual terms in the arguments and forming all possible combinations of alternative representations of terms.

  Examples:
  - *"the Carnegie Mellon campus" "at the west coast" established*
  - *("the Carnegie Mellon campus" OR "the CMU campus") "at the west coast" (established OR founded OR launched)*

  In these examples, "established" is the predicate verb and "the Carnegie Mellon campus" and "at the west coast" the arguments with the semantic roles *ARG1* and *ARGM-LOC*.

- Furthermore, I implemented **combinations** of the above schemes, for instance queries consisting of the terms and, in addition, the individual keywords of multi-token terms.

  Combined queries have the advantage of yielding results even if there is no exact match for a compound expression in the corpus. This can be beneficial when searching a small corpus with little redundancy. Web search engines, on the other hand, usually require a document to match each of the expression in the query string and thus the combination with fine-grained expressions has no effect on their results.

  Examples:
  - *"Carnegie Mellon" campus "west coast" established Carnegie Mellon west coast*
  - *("Carnegie Mellon" OR CMU) campus "west coast" (established OR founded OR launched) Carnegie Mellon west coast*

I evaluated these query generation techniques as well as combinations of them using both factoid questions from previous TREC evaluations (see Section 5.1) and a small test collection of scenario-based questions that has been developed for the CNS corpus (cf. Section 5.2).

For my experiments with recent TREC questions, I first resolved coreferences within the questions using the algorithm described in [SGS06] and then formed query strings from both the question and the target of the question series to ensure that the context is preserved. I relied on the Web as the only resource and did not project the answers onto the AQUAINT corpus, which was used in the TREC evaluations.

For the experiments on the CNS corpus, I also expanded queries at the level of keywords. These queries require nested boolean combinations, which are supported by Indri. Web search engines such as Google and Yahoo, however, do not allow nested operators, and thus I could not use such queries for answering TREC questions. An example of an expanded keyword query is given below:

> *((Carnegie AND Mellon) OR CMU) campus west coast (established OR founded OR launched)*

The Java API for the Google search engine was used to fetch text snippets from the Web. As Google snippets often consist of incomplete sentence fragments which are not parsable, I downloaded the whole Web documents that contained the snippets. In particular, I fetched the first 100 distinct documents referenced in the search results for each type of query. Documents that did not contain regular HTML code such as PDF and PS files were ignored. The HTMLParser [Par] was used to convert the Web pages to plain text.

3.3 Factoid Answer Extraction and Selection

Given the semantic analysis of the question and the Web documents, I first identify sentences in the documents that potentially contain similar semantic structures. These sentences are further analyzed and their semantic representations are compared against the question. Factoid answers are extracted from sentences that match the semantic structure of the question. Finally, I drop malformatted and unresponsive answer candidates, and I merge and boost the scores of similar answers.

3.3.1 Extraction of Relevant Sentences

The Web documents are parsed into sentences, using the sentence detector from LingPipe [Lin], a Java API for the linguistic analysis of natural language. Semantic parsing is a time-intensive task, and thus it is not feasible to parse all the sentences in the documents, but it is necessary to first narrow down the number of candidate sentences before applying the parser.

At first, the length of each sentence is checked against an upper and lower threshold and a sentence is dropped if its length does not fall within these boundaries. Web documents often contain sentence fragments, e.g. as part of menu bars or advertisements, that can be ignored. In addition, the segmenter may fail to correctly recognize sentence boundaries and return long phrases of multiple sentences that cannot be parsed.

Furthermore, I require a sentence to contain a predicate from the question that has a semantic role marked as "missing". A POS tagger is used to identify all verbs in a sentence and their infinitives are compared against the lemmas of the predicate verbs in the question. WordNet is used to determine the infinitive form of a verb.

If the answer type of the question is known, each sentence is further required to contain a named entity of the expected type that is different from the entities in the question. For instance, a candidate sentence for the question "Who is the wife of Bill Clinton?" would have to contain a person's name other than "Bill Clinton" to be considered relevant.

Finally, each sentence has to contain a term that is similar to a term in one of the arguments of a predicate in the question. Otherwise, the sentence cannot contain a predicate that is similar to a question predicate. The concepts of term similarity and predicate similarity are discussed in the next section.

3.3.2 Predicate Matching

All sentences that satisfy the above constraints are parsed with ASSERT and predicates are extracted. These answer predicates are compared to the question predicates that have a missing argument and a predicate similarity score is computed for each pair of predicates. The confidence score assigned to an answer predicate is the maximum of the similarity scores over all question predicates with missing arguments. In the following I describe how similarity scores are calculated.

Term similarities are calculated by comparing the keywords that occur within two similar terms. Both terms are replaced by their lemmas in WordNet. The lemmas are tokenized and function words are dropped. The term similarity of two terms t_1 and t_2 is defined as the *Jaccard coefficient* of the sets of content words W_1 and W_2 extracted from the terms:

$$Sim_{Term}(t_1, t_2) := J(W_1, W_2) = \frac{|W_1 \cap W_2|}{|W_1 \cup W_2|}$$

As mentioned earlier, a question term t is expanded with related concepts $R = \{r_1, ..., r_n\}$ with weights $w(r_1), ..., w(r_n)$ (see Chapter 4 for details). These alternative representations of a question term and their weights are also taken into account

when comparing it to a term in an answer sentence. This leads to the more general definition of the **expanded term similarity** between an answer term t_a and a question term t_q:

$$Sim_{ExpTerm}(t_a, t_q) := \max_{t \in \{t_q\} \cup R} (w(t) \times Sim_{Term}(t_a, t))$$
$$\text{where} \quad w(t_q) := 1.$$

The **verb similarity** of an answer predicate p_a and a question predicate p_q is simply the expanded term similarity of the terms t_{verb_a} and t_{verb_q} that represent the verbs of the two predicates:

$$Sim_{Verb}(p_a, p_q) := Sim_{ExpTerm}(t_{verb_a}, t_{verb_q})$$

The **argument similarity** of an answer predicate p_a and a question predicate p_q is determined by comparing the sets of terms within the arguments of the predicates, denoted T_a and T_q. I have extended the concept of the Jaccard coefficient to take the similarity of terms into account. The Jaccard coefficient only distinguishes between common elements and elements that appear exclusively in one set. The argument similarity also depends on similar terms, even if there is no per-token match:

$$Sim_{Args}(p_a, p_q) := \frac{\sum_{t_a \in T_a} \left(\max_{t_q \in T_q} \left(Sim_{ExpTerm}(t_a, t_q)\right)\right)}{|T_q| + \left|\left\{ t_a \in T_a \;\middle|\; \max_{t_q \in T_q} \left(Sim_{ExpTerm}(t_a, t_q)\right) = 0 \right\}\right|}$$

Each term in T_a is compared to all terms in T_q and the maximum of the similarity scores is computed. If the maximum is larger than 0, then the term is assumed to be covered by both predicates and the numerator of the coefficient is incremented by this score, else the denominator is incremented by 1.

The **predicate similarity** of an answer predicate p_a and a question predicate p_q is defined as the product of their verb and argument similarity scores:

$$Sim_{Predicate}(p_a, p_q) = Sim_{Verb}(p_a, p_q) \times Sim_{Args}(p_a, p_q)$$

The scoring mechanism has been designed to be flexible and robust to parsing errors in order to maximize the recall of the answer extraction. The idea of using a Jaccard coefficient to measure the similarity of all arguments as a whole was introduced in [SJT+05]. It takes into account that semantic role labeling systems often fail to assign the correct semantic roles to the arguments, which makes a per-argument comparison infeasible. I have extended this idea to perform a fuzzy matching not only for arguments but also at the level of terms.

In the following, these similarity measures are illustrated using the example of the question "In what year was the Carnegie Mellon campus at the west coast established?" and the answer sentence "The CMU campus at the US west coast was founded in the year 2002." The sentence satisfies all the constraints discussed in Section 3.3.1: The question predicate "established" is covered by the related concept

- TARGET: founded
  TERM: founded (POS: VBN, NE Types: -)
- ARG1: The CMU campus
  TERM: CMU (POS: NN, NE Types: Organization)
  TERM: campus (POS: NN, NE Types: -)
- ARGM_LOC: at the US west coast
  TERM: US west coast (POS: Compound, NE Types: Location)
- ARGM_TMP: in the year 2002
  TERM: year (POS: NN, NE Types: -)
  TERM: 2002 (POS: NN, NE Types: Date, Year)

Figure 3.2: Semantic representation of the statement "The CMU campus at the US west coast was founded in the year 2002.".

"founded" and the argument terms "Carnegie Mellon", "campus" and "West coast" have similar terms in the answer sentence. Furthermore, the sentence contains an entity of type *Year*, which is the expected answer type of the question. Therefore, the sentence is selected for a semantic analysis, illustrated in Figure 3.2, which is subsequently compared to the semantic representation of the question, shown in Figure 3.1.

The verb similarity is the expanded similarity of the verb terms (f: founded, e: established, l: launched):

$$\begin{aligned} Sim_{Verb} &= Sim_{ExpTerm}(f, e) \\ &= \max\{w(e) \times Sim_{Term}(f, e), w(f) \times Sim_{Term}(f, f), w(l) \times Sim_{Term}(f, l)\} \\ &= \max\{1 \times 0, 0.8 \times 1, 0.7 \times 0\} = 0.8 \end{aligned}$$

The terms in the arguments of the question and answer predicates are compared pairwise. The answer term "campus" also appears in the question, while the term "CMU" is most similar to the question term "Carnegie Mellon" and the closest match for "US west coast" is the term "west coast". The terms "year" and "2002" cannot be associated with a term in the question. The argument similarity score is calculated as follows (c_1: CMU, c_2: Carnegie Mellon, c: campus, w_1: US west coast, w_2: west coast):

$$\begin{aligned} Sim_{Args} &= \frac{Sim_{ExpTerm}(c_1, c_2) + Sim_{ExpTerm}(c, c) + Sim_{ExpTerm}(w_1, w_2)}{5} \\ &= \frac{0.9 \times Sim_{Term}(c_1, c_1) + 1 \times Sim_{Term}(c, c) + 1 \times Sim_{Term}(w_1, w_2)}{5} \\ &= \frac{0.9 + 1 + \frac{2}{3}}{5} \approx 0.513 \end{aligned}$$

Finally, the predicate similarity is the product of the verb similarity and argument similarity scores:

$$Sim_{Predicate} = Sim_{Verb} \times Sim_{Args} \approx 0.8 \times 0.513 \approx 0.411$$

3.3.3 Extraction of Factoid Answers

Predicates with confidence scores larger than 0 are semantically similar to the question and are therefore considered as candidates for factoid answer extraction. The extraction strategy depends on whether the answer type of the question could be determined during the question analysis phase:

- If the answer type of the question is unknown, the answer extractor checks if the answer predicate has an argument with a semantic role that is marked as missing in one of the question predicates. This argument is extracted as an answer candidate.
- If the answer type is available, entities of the expected type are extracted from the arguments. A natural approach would be to extract entities only from arguments with semantic roles that are missing in the question. However, as mentioned earlier, ASSERT often assigns incorrect semantic roles to arguments and thus the recall of the answer extraction would suffer from this restriction. Thus I decided to relax this constraint and extract answer candidates from all arguments of the answer predicate. Both variants have been evaluated (see Section 5.1) and the results show that the second variant in fact yields the higher accuracy.

Due to redundancy, the same factoid answer is usually extracted from more than one predicate. The confidence score of an answer candidate is the sum of the confidence scores of all the predicates it was extracted from.

3.3.4 Answer Selection

The semantic parser often fails to correctly recognize argument boundaries. Therefore, answer candidates that have not been extracted with a named entity tagger but merely by selecting an argument with a semantic role that is missing in the question sometimes need to be truncated. In particular, the following prefixes and suffixes are cut off:

- Whitespaces and non-word characters except symbols for units (such as currencies, percentage)
- Articles, ”and”, ”or”
- Prepositions (e.g. ”in” preceding a location)

In addition, there are often unresponsive answers among the arguments that can be filtered out. The following types of answers are dropped:

- Function words and concatenations of function words (e.g. ”he”, ”not yet”)

- Answers that contain an interrogative pronoun (e.g. "what city")
- Obviously malformatted answers, e.g. answers that contain a single bracket
- Adverbs (e.g. "recently")

Furthermore, answer candidates that repeat information which is already provided in the question are dropped. This is particularly important if the question contains a named entity of the expected answer type. For instance, the question "Who killed John F. Kennedy?" already contains an entity of the answer type *Person*, which would certainly be among the answer candidates as it appears in every snippet retrieved with the question keywords. This approach works well for most TREC-style questions, but it causes difficulties if a correct answer repeats part of the question (e.g. "What are the members of the Kennedy clan?").

The remaining answers are compared pairwise and syntactically similar answers are merged, their scores summed up. Answer candidates are considered similar if their normalized forms, obtained by tokenization and stemming with a Porter stemmer [Por80], are identical.

Subsequently, the set of answer candidates is checked for subset relations. If the (normalized) keywords within one factoid answer form a subset of the keywords of another candidate, then the former is dropped and its score is transferred to the latter. In this way, longer and more specific answers are preferred over shorter ones. For instance, if there are two answer candidates "John Kennedy" and "John F. Kennedy", the second answer is retained and the first, more ambiguous answer is removed. However, malformatted answers that contain additional tokens should not repress shorter answers that have been truncated correctly (e.g. "1879 but" should not be preferred over "1879"). For this reason, answers are only merged if the longer answer has been extracted with a named entity tagger and thus is known to be properly formatted.

3.4 Runtime and Caching

Some of the above question analysis and answer extraction techniques can be fairly time-intensive. Caching is a useful technique to improve the runtime performance and to speed up evaluations on large test sets such as the TREC 15 collection. However, it is important to carefully choose the granularity of the elements to be cached, which is usually a trade-off between the reduction in runtime and the flexibility of the cache. For instance, one could cache the search results for each question, which would reduce the runtime of the search component on repeated questions to zero, but the cache would need to be reset whenever the component is modified.

A runtime analysis revealed that the by far most time consuming tasks are the semantic parsing and the retrieval of the web documents referenced in the search engine snippets. Thus, by caching the retrieved web sites and the output of the semantic parser for each input sentence, the runtime can be reduced significantly. A cache of this granularity only needs to be reset if the semantic parser itself is modified or when the web sites in the cache are outdated.

I have implemented a simple cache that stores the retrieved web sites and output from ASSERT in files, using the MD5 checksum of the URL or input sentence as the

filename. Ephyra always searches the cache first and only processes the remaining URLs and sentences that do not have an entry in the cache. In this way, repeated evaluations that use the same test set but different system setups can be greatly accelerated.

3.5 Combination with Syntactic Techniques

This section motivates the combination of the above semantic approach with existing syntactic answer extraction techniques. It further describes how confidence scores from the different extractors can be made comparable by means of normalization, and how these normalized scores can be merged into a single, improved score.

3.5.1 Motivation

I have analyzed the questions of the TREC 15 test set to validate the generality of the semantic approach. The test set is composed as follows:

- Ca. 60% of the questions contain a parsable predicate and the answer is an argument of this predicate.

  For example, "In what year was Moon born?" (Question 141.3) contains the parsable predicate *born(ARG1: Moon, ARGM-TMP: *missing*)*.

- Ca. 35% of the questions only contain one of the ambiguous predicates *to be*, *to do* and *to have*.

  For example, "How many events are part of the LPGA tour?" (Question 142.3) only contains the ambiguous predicate *are*.

- Ca. 5% of the questions contain a parsable predicate, but the answer is an argument of another, ambiguous predicate.

  For example, "What color was the dress that she wore at her birthday lunch?" (Question 165.2) contains the parsable predicate *wore(ARG0: she, ARG1: the dress, ARGM-LOC: at her birthday lunch)*, but the answer is an argument of the ambiguous predicate *was*.

The SRL-based approach for question analysis and answer extraction is limited to the first type of questions, which constitute about 60% of the test set. Some of the remaining questions could be rephrased into questions of the first category as described in Section 3.1, but there still remains a significant portion of questions that are not covered by this approach.

On the other hand, the semantic answer extraction is more precise as it requires an answer sentence to preserve the semantic structure of the question. Syntactic techniques, in contrast, can be misled by sentences that contain the same keywords as the question but that do not retain its meaning. For instance, a query for the keywords in the question "Who killed Lee Harvey Oswald?" would likely result in statements such as "Lee Harvey Oswald killed John F. Kennedy." Since "John F. Kennedy" is of the expected answer type and in proximity with the question keywords, a simple extractor based on answer type analysis would falsely extract it as

an answer candidate. However, a semantic analysis shows that the semantic roles are switched: We are looking for a predicate in which "Lee Harvey Oswald" is the patient and not the agent. Furthermore, the semantic approach can answer a question even if the type of the answer is unknown or no suitable named entity tagger is available.

Therefore, the combination of the semantic answer extractor with syntactic answer extraction techniques appears reasonable, and experiments (see Section 5.1) show that a combined approach indeed yields the highest accuracy.

What remains is the question of how different extraction techniques can be combined to produce one ranked list of answer candidates. One can simply prefer answers from one extractor and use the other extractors as a fallback if the preferred approach fails to retrieve any answers. This strategy is reasonable if there are just two or three extractors, and one has a significantly higher precision than the others. In the experiments discussed in Section 5.1.2, I combined the answer candidates from the semantic extractor and Ephyra's two syntactic extractors in this way, always preferring the semantic approach over answer type analysis and using the pattern learning approach as a third choice.

However, different extractors use different evidence to identify relevant answers, which is particularly true when combining syntactic and semantic techniques. An answer candidate found by more than one extractor is more likely to be correct than an answer retrieved with a single technique. It is therefore desirable to consider the results from all extractors simultaneously when compiling the final list of ranked answers.

A combined confidence score for an answer candidate could be calculated as a weighted sum of the scores of all extractors that found the answer. However, the underlying scoring mechanisms can be very different and may produce incomparable results. Scores can depend on many factors such as the type of the question, the expected answer type and the redundancy of the corpus. The answer extractors in Ephyra calculate confidence measures by summing up the scores of multiple occurrences of an answer candidate in the corpus. The final score of a candidate is unbounded and depends on the number of instances found in the corpus.

Therefore, it is necessary to normalize the scores from the different extractors do make them comparable (Section 3.5.2). These normalized scores can then be combined into a single score that reflects the confidence of all extractors (Section 3.5.3).

3.5.2 Score Normalization

Score normalization techniques can be distinguished into scaling schemes that transform the scores of the answers for each question independently (Section 3.5.2.1) and supervised training approaches that learn from scores and relevance judgements of results for many questions (Section 3.5.2.2).

3.5.2.1 Local Scaling

A number of normalization techniques have been proposed by the document retrieval community. Score normalization is deployed in metasearch engines that combine search results from multiple systems. Most of the techniques used for document

retrieval have in common that they treat the algorithms that generated the scores as black boxes. Furthermore, they only consider the results to a single query at a time, trying to rescale their confidence scores while preserving the order suggested by the original scores.

A simple and frequently used approach (see for example [Lee97]) is the linear transformation of a set of confidence scores S to the interval $[0, 1]$:

$$Norm_{Standard}(s) = \frac{s - \min(S)}{\max(S) - \min(S)} \quad \forall s \in S.$$

Montague and Aslam [MA01] introduced the following three desirable qualities of a score normalization scheme:

- **Shift invariance.** Let S be a set of scores, and S_a be the scores obtained by adding a constant $a \in \mathbb{R}$ to each score: $S_c = \{s_c \mid s_c = s + a,\ s \in S\}$. A normalization scheme $Norm$ is shift invariant if $\forall s \in S : Norm(s_c) = Norm(s)$.
- **Scale invariance.** Let S be a set of scores, and S_m be the scores obtained by multiplying each score with a constant $m \in \mathbb{R}$: $S_m = \{s_m \mid s_m = s * m,\ s \in S\}$. A normalization scheme $Norm$ is shift invariant if $\forall s \in S : Norm(s_m) = Norm(s)$.
- **Outlier insensitivity.** A single outlier with an exceptionally large or small score should not corrupt the normalized scores of the remaining results.

The standard normalization scheme is shift invariant and scale invariant, but it it highly sensitive to outliers since they affect $\min(S)$ or $\max(S)$. To avoid outlier sensitivity, [MA01] suggests two other normalization schemes:

- Shift the minimum score to 0 and scale the sum to 1:

$$Norm_{Sum}(s) = \frac{s - \min(S)}{\sum_{s' \in S} s' - \min(S)} \quad \forall s \in S.$$

- Shift the mean to 0 and scale the variance to 1 (Zero-Mean, Unit-Variance):

$$Norm_{ZMUV}(s) = \frac{s - \mu}{\sigma} \quad \forall s \in S;$$
$$\mu = \frac{1}{|S|} \sum_{s \in S} s \text{ (mean)};$$
$$\sigma = \sqrt{\sigma^2} = \sqrt{\frac{1}{|S|} \sum_{s \in S} (s - \mu)^2} \text{ (standard deviation)}.$$

Both techniques are shift invariant and scale invariant. $Norm_{Sum}$ is only sensitive to the minimum score, which in practice is often bounded. In Ephyra, for instance,

factoid answers are guaranteed to have a positive score. $Norm_{ZMUV}$ is insensitive to small and large outliers, but the normalized scores are unbounded in both directions and, since the mean is scaled to 0, there are answers with negative scores. This has to be taken into consideration when assigning scores to results that have not been retrieved with a particular approach. Often, unretrieved documents are simply assigned a score of 0, but this is not possible with $Norm_{ZMUV}$ as results with negative scores would be penalized although they are actually supported. Montague and Aslam suggest to assign a negative score of -2 to all results that have not been found with a particular retrieval approach.

Local scaling techniques have been widely adopted in document retrieval systems. They are easy to implement, fast and do not require any training. On the other hand, they only take the scores of the results for a single query into account. This may be sufficient for document retrieval systems that provide hundreds of results per query, but it is of limited use for factoid question answering.

The answer extractors in Ephyra sometimes retrieve only a handful of answer candidates, in which case the normalized scores are rather arbitrary. For instance, if an extractor suggests just a single answer, the normalized score is undefined and can only be set to a pre-defined constant. If there are two candidates, then their normalized scores do not reflect the original scores. $ZMUV$, for example would always assign the higher ranked answer a score of 1 and the lower ranked one the score -1. This makes it impossible to tell from the normalized scores whether the extractor was confident about the answers.

In addition, these normalization techniques do not consider the characteristics and underlying scoring mechanisms of the answer extractors. Without additional knowledge, there is no way of telling what a score of, say, 100 implies about the confidence of the extractor in the answer, and it is not possible to compare it to a score of 100 from another extractor. In past TREC evaluations, some of the factoid questions did not have an answer in the document collection. For such questions, systems were required to return the answer "NIL", indicating that the question is not answerable. Therefore, the answer selection component needs reliable and comparable confidence measures.

3.5.2.2 Global Training

For these reasons, the local scaling techniques cannot readily be used to normalize relevance scores of answer candidates in QA. Better results can be obtained with a global training approach that takes the answers to many questions with different characteristics into account.

Several binary classifiers have been applied to categorize answer candidates into correct and incorrect ones. The classifiers were trained on the output of the three answer extractors in Ephyra (answer type analysis, pattern learning and semantic parsing) on TREC questions from past years. The training was supervised, i.e. the answers were judged with the TREC answer keys and each answer was labeled as correct or incorrect. Subsequently, the classifiers were utilized for score normalization: the probability of the positive class, which is the likelihood of an answer candidate being correct, was used as the normalized score of the answer.

Ittycheriah et al. [IFR01] use a maximum entropy classifier to estimate the probability that an answer candidate is correct. This approach was adopted in the JAVELIN

cross-lingual QA system [MLS+07] to score Chinese and Japanese answer candidates. In both systems, the classifier is part of the extraction component and makes use of features that are specific to the extractor, such as the proximity of key terms and the semantic similarity of the answer to the question.

The approach discussed in this section is different in that the normalization is performed in an independent step and for multiple answer extractors at once. Therefore, I have focused on features that are available for all extraction techniques, which makes it easy to incorporate additional extractors and retrain the system without modifying the feature set. The following features were considered as indicators for the correctness of an answer candidate:

- *Score.* The original score of the candidate from the answer extractor.
- *Extractor.* The answer extractor that found the candidate.
- *ATypes.* The predicted answer type(s) of the question.
- *Num.* The number of answer candidates from the same extractor.
- *Mean.* The mean score over all these candidates.
- *Max.* The maximum score over all these candidates.
- *Min.* The minimum score over all these candidates.

Minorthird, a Java toolkit of machine learning methods [Coh04], was used to train classifiers with different combinations of the above features. Minorthird implements various learning algorithms, among which I evaluated and compared the following:

- Adaptive boosting (*AdaBoost*) of a decision tree, cf. Section 2.4
- Logistic regression version of adaptive boosting (*AdaBoost L*)
- *Balanced Winnow*
- *Decision Tree* with a maximum depth of 5
- 5-Nearest Neighbor (*5NN*)
- *Margin Perceptron*
- Maximum Entropy (*MaxEnt*), cf. Section 2.4
- *Naive Bayes*
- *Negative Binomial*
- *Voted Perceptron*

Features	**Model**		
	AdaBoost	Decision Tree	MaxEnt
Score	0,072	0,081	0,055
Score, Extractor	0,076	0,060	0,055
Score, Extractor, ATypes	0,096	0,060	0,058
Score, Extractor, ATypes Mean, Max, Min	0,164	0,060	0,057
Score, Extractor, ATypes Num, Max, Min	0,169	0,060	0,060
Score, Extractor, ATypes Num, Mean, Max, Min	0,140	0,060	0,073

Table 3.2: Selected models and feature combinations for score normalization.

Different combinations of these models and features were evaluated on past TREC questions. For each combination, a 3-fold cross validation was performed on the raw output of all three answer extractors on the factoid and list questions from the TREC 13 to 15 evaluations. The precision and recall in recognizing correct answers was measured and combined in an F_1 score (equally weighted precision and recall):

$$\text{Precision} = \frac{\text{Number of correct answers classified as correct}}{\text{Number of answers classified as correct}},$$

$$\text{Recall} = \frac{\text{Number of correct answers classified as correct}}{\text{Number of correct answers}},$$

$$F_1 = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}.$$

Table 3.2 shows the F_1 scores of MaxEnt models, decision trees and AdaBoost classifiers (using decision trees as week learning algorithms) trained on different feature combinations. The scores for all evaluated classifiers and feature combinations are listed in Appendix A. The results indicate that adaptive boosting performs best, in particular it outperforms the maximum entropy classifier used in previous approaches. Furthermore, the results suggest to use all features except the mean score, which appears to be little relevant to the correctness of an answer candidate.

The AdaBoost classifier used in these experiments called the weak learner 10 times. The performance of the classifier can be further improved by increasing the number of rounds. Figure 3.3 illustrates the performance of adaptive boosting for different numbers of rounds and the three most effective feature combinations. The precise numeric values are given in Appendix A. The highest F_1 scores are achieved with about 50 to 80 rounds of boosting. The results also indicate a tendency to overfitting if the number of rounds is further increased.

The classifiers do not necessarily preserve the order of the answers suggested by the original confidence scores from the answer extractors. This is unfavorable because the original scores are usually more reliable indicators for the relevance of an answer

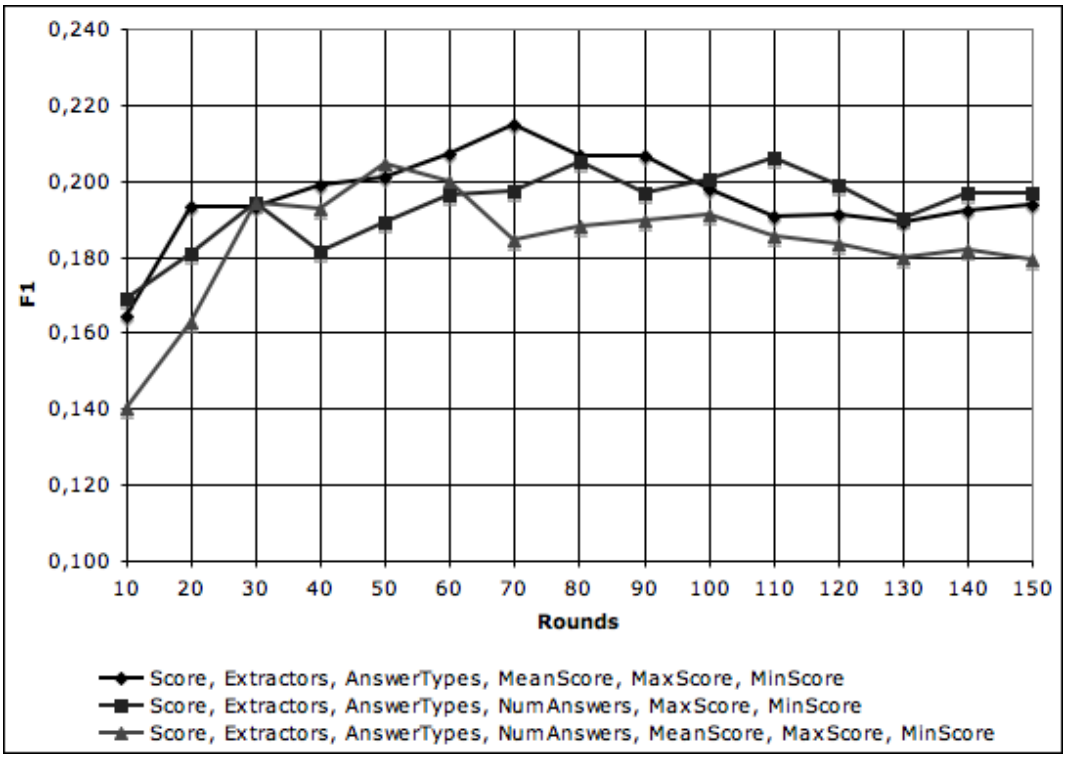

Figure 3.3: Evaluation of AdaBoost classifiers with different numbers of rounds and combinations of features.

candidate compared to other candidates from the same extractor. Therefore, different techniques have been devised to restore the original order of the results for each extraction technique by adjusting the normalized confidence scores:

- *Averaging.* For each answer, the normalization factor f_{norm} is computed:

$$f_{norm} = \frac{\text{Normalized score}}{\text{Original score}}.$$

  New normalized scores are calculated by multiplying the original scores with the average normalization factor over all answers from the same extractor.

- *Rescaling.* f_{norm} is computed only for the top answer from each extractor according to the original scores. The normalized scores are recalculated with this normalization factor.

- *Resorting.* For each extraction technique, the answers are sorted by their original scores and the normalized scores are reassigned to the answers in descending order.

For all further experiments, I used an AdaBoost classifier with 70 rounds of boosting and the features Score, Extractor, ATypes, Num, Max and Min. The classifier was trained on the output of the three answer extractors for all factoid and list questions from the TREC 8 to 10 and TREC 12 to 15 evaluations. TREC 11 was left untouched to be used as a test set.

3.5.3 Score Combination

Given the normalized scores from the AdaBoost classifier, the final task is the combination of these scores in a joint score that reflects the confidence of all extractors. Various score combination techniques have been proposed by the document retrieval community [FS93]. I have applied the following techniques to combine the normalized scores from different extractors:

- *CombMIN*: Minimum of the confidence scores from all extractors.
- *CombMED*: Median of the confidence scores from all extractors.
- *CombMAX*: Maximum of the confidence scores from all extractors.
- *CombSUM*: Sum of the confidence scores from all extractors.
- *CombANZ*: CombSUM divided by the number of extractors that found the answer.
- *CombMNZ*: CombSUM multiplied by the number of extractors that found the answer.

In addition, I have evaluated a score combination scheme that is based on complementary probabilities. Let $s_1, ...s_n$ be the normalized confidence scores for an answer candidate found with n extractors. The combined score is calculated as follows:

$$CombCP = 1 - \prod_{i=1}^{n} (1 - s_i)$$

The normalized confidence scores from the different extractors are interpreted as estimations of the probability of the answer candidate. Assuming that these probabilities are statistically independent, *CombCP* represents the overall probability of the answer candidate being correct.

These techniques have been combined with the different approaches for adjusting the normalized scores discussed in the previous section. Evaluation results are shown in Section 5.1.4.

4. Ontologies

In this thesis, ontologies are used to (1) extract terms from questions and answer sentences and to (2) expand these terms with semantically similar concepts. A term is a phrase consisting of one or more tokens that form a unit of meaning, e.g. a person's first and last name. It is desirable to expand terms as a whole rather than individual tokens to ensure that their sense is preserved.

The purpose of term extraction and expansion is twofold: (1) Terms and alternative representations are used at the query generation stage to form queries that are independent of the formulations used in the question and thus yield a higher recall in document retrieval (cf. Section 3.2) and (2) the similarity measure for predicates described in Section 3.3.2 is based on terms rather than arguments or other high-level semantic structures to allow a greater flexibility and improve the robustness to parsing errors.

4.1 Term Extraction and Expansion

Ephyra provides a framework for term extraction and expansion that is based on *named entity taggers*, *dictionaries* and *ontologies*. Dictionaries of multi-token terms and named entity taggers are used to extract terms from sentences. The semantic relations in ontologies are deployed to enrich the extracted terms with related concepts. External semantic resources can be integrated in this framework by wrapping them as either dictionaries or ontologies. In Ephyra, each ontology is also a dictionary, i.e. it can be used to look up multi-token terms, but not vice versa. By reusing the ontologies as dictionaries, it can be ensured that the extracted terms occur in one of the ontologies and can subsequently be expanded.

4.1.1 Term Extraction

At first, Ephyra applies all available named entity taggers to a sentence and compiles the results into a dictionary of named entities. A sliding window of tokens is then used to extract multi-token expressions from the sentence. The size of the window is first set to a predefined maximum value and is gradually decreased to extract shorter phrases and eventually single tokens. The windows are shifted over the

sentence to extract phrases of the respective length. The maximum window size has been restricted to 4 as terms rarely consist of more than 4 tokens.

The multi-token phrases are looked up in the precompiled list of named entities and the dictionaries to identify terms. A multi-token term can be a named entity, a compound verb (e.g. "to pass away", "to slow down") or a compound noun (such as "computer science", "weapons of mass destruction"). Dictionary lookups can potentially be expensive, depending on the underlying semantic resource, and thus the number of lookups should be kept to a minimum. To avoid unnecessary dictionary lookups, only verb phrases and noun phrases are considered as multi-token terms. Duplicates of phrases that have already been checked and phrases that do not contain any content words are ignored. In addition, phrases that can be truncated with the algorithm described in 3.3.4 are skipped (e.g. phrases that start with prepositions). Furthermore, terms must not overlap with any previously extracted longer terms. Thus, if a term is contained in another term (e.g. "mass destruction" as part of "weapons of mass destruction"), only the longer term is extracted and the shorter term is discarded.

4.1.2 Term Expansion

The framework for term expansion supports three types of concepts:

- *Events* describe relations between objects.
- *Entities* are the objects that participate in events.
- *Modifiers* further qualify events.

Not all ontologies can expand all three types (e.g. the semantic networks introduced in Section 4.4 are currently only applicable to events) and different relations may be used for different concept types (e.g. causal relations are not defined for modifiers).

At first, a term is replaced by its lemmas in WordNet. The lemma is expanded with all available ontologies, whereby the type of the term determines the relations that are used to expand it. Each ontology returns a set of related concepts associated with weights that reflect their similarity to the original term. The results are merged and duplicates are dropped except for the instance with the maximum weight. If the term is a verb, the results are converted to the original verb form (infinitive, gerund, third person singular present tense, simple past or past participle), using grammatical rules for regular verbs and lists of irregular verbs to derive the correct forms. The original verb form is restored to improve the document retrieval results when related verbs are used for query expansion.

Two sets of thresholds are used to restrain the number of alternative representations and their minimum weights for query expansion (Section 3.2) and for predicate similarity scoring (Section 3.3.2). Initial experiments showed that queries should only be expanded with synonyms whereas for predicate matching, all available relations can be used. Web search engines do not support weighted expressions, thus all terms in a query are considered equally important and the precision suffers from extensive disjunctions of terms. The predicate similarity measure, on the other hand, takes

Relations	Examples	Weights
alias	buy → purchase	1.0
is-a	export → give ship → give	0.9
subtype	give → export give → ship	0.9
sibling	export → ship	0.9
reflexive	collaborate → collaborate	1.0
inverse	buy → sell	1.0
implies	construct → possess	0.8
implied-by	possess → construct	0.8

Table 4.1: Relations in the CNS ontology used for term expansion.

the weights of alternative representations into account and remotely related concepts only carry weight if no closer matches are found.

Ephyra makes use of three complementary semantic resources, which are discussed in the remainder of this chapter. Section 4.2 introduces a framework for domain-specific ontologies, Section 4.3 deals with WordNet relations and Section 4.4 describes how relations between events can be derived from a semantic network built from a text corpus with semantic annotations.

4.2 Domain-Specific Ontologies

Domain-specific ontologies are supported through a framework that has been adapted from the JAVELIN system [NMF+05]. For my experiments, I could reuse a sample ontology built from a corpus created by the Center for Nonproliferation Studies (CNS). The corpus comprises documents about the proliferation of weapons of mass destruction, thus covers a relatively narrow domain. The ontology represents knowledge about weapon categories, geo-political entities and events that describe the transfer of weapons between these entities.

The CNS ontology has been handcrafted from lists of frequent verbs and nouns in the corpus. Verbs correspond to events in the term expansion framework, nouns correspond to entities. The ontology only contains the most frequent concepts, but yet it covers a large portion of all instances of nouns and verbs in the corpus. Table 4.1 gives an overview of the relations in the CNS ontology.

The framework for domain-specific ontologies is not only applicable to terms but it can also expand predicate-argument structures extracted with a semantic role labeling system. When applied to predicate verbs, some of the relations require permutations of the semantic roles of the arguments. Table 4.2 shows how the arguments are altered.

An ontology for a restricted domain can provide expert knowledge that is not covered by open-domain resources such as WordNet. Furthermore, the disambiguation of word senses is facilitated as not all possible senses of a word are common in a specific domain. For instance, in the CNS domain, the verb ”to buy” can be assumed to have the meaning ”to purchase” rather than ”to accept as true”. On the other hand, creating an ontology is a time-consuming task that recurs for each domain

Relations	Permutations	Examples
reflexive	ARG0 → ARG1, ARG1 → ARG0	[ARG0: A] collaborates with [ARG1: B] ⇓ [ARG0: B] collaborates with [ARG1: A]
inverse	ARG0 → ARG2, ARG2 → ARG0	[ARG0: A] buys X from [ARG2: B] ⇓ [ARG0: B] sells X to [ARG2: A]

Table 4.2: Relations in the CNS ontology requiring permutations of semantic roles.

to be covered. Section 4.4 proposes a technique for automatically deriving semantic relations from a text corpus.

4.3 WordNet

WordNet has been widely used as a semantic resource for question answering. It has been deployed for a variety of tasks such as answer type classification, factoid answer recognition, answer validation, lemmatization, key term extraction and term expansion. The latter is dealt with in this section, following a similar approach as [SJT+05], a question answering system from the National University of Singapore that has successfully participated in past TREC evaluations (cf. Section 1.4.2).

WordNet [Fel98] is a lexical database for the English language developed at Princeton University. It comprises nouns, verbs, adjectives and adverbs, grouped together in sets of words with similar meanings called *synsets*. Many English words, in particular verbs, have multiple senses and thus belong to more than one synset in WordNet. For instance, WordNet lists 27 distinct meanings of the verb ”to work” such as ”to exercise”, ”to function” or ”to shape”.

In addition, WordNet provides various types of relations between words, or more precisely, word senses represented by synsets. WordNet can therefore be regarded as a network or ontology that semantically links concepts from the English language as well as proper names such as famous people and organizations. The relations in WordNet can be exploited to find terms that are semantically related to a given concept.

Table 4.3 presents an overview of the WordNet relations used for term expansion, along with examples to illustrate their meanings. For each relation, it also lists the parts of speech that are covered. The last column assigns weights to the relations as suggested in [MN02] (see Section 1.4.1 for details). WordNet provides a few more relations which I decided not to use because they are either too vague or they only cover a small fraction of the synsets. The *see also* relation, for example, points to concepts that are related in some unspecific way, while the *pertainym* relation (e.g. the adjective ”musical” pertains to the noun ”music”) hardly has instances in the database.

As relations in WordNet are defined on synsets and not on terms, one needs to select a synset that represents the appropriate word sense of the term to be expanded. I followed a simple approach for word sense disambiguation that yields reasonably good results: Among all possible synsets that contain a given term, I choose the

Relations	Examples	Parts of Speech	Weights
Synonym	author → writer	noun	0.9
	buy → purchase	verb	0.9
	large → big	adjective	0.9
	recently → lately	adverb	0.9
Hypernym	car → vehicle	noun	0.8
	buy → acquire	verb	0.8
Hyponym	vehicle → car	noun	0.7
	acquire → buy	verb	0.7
Entailment	buy → pay	verb	0.7
Cause-to	kill → die	verb	0.5
Member-of Holonym	college → university	noun	0.5
Substance-of Holonym	flour → bread	noun	0.5
Part-of Holonym	accelerator → car	noun	0.5
Has-Member Meronym	university → college	noun	0.5
Has-Substance Meronym	bread → flour	noun	0.5
Has-Part Meronym	car → accelerator	noun	0.5

Table 4.3: WordNet relations used for term expansion.

first synset returned by WordNet, which represents the most frequent meaning of the term.

WordNet can be used to expand all three types of concepts supported by the framework introduced in Section 4.1: entities, events and modifiers. Entities correspond to WordNet nouns, events correspond to verbs and modifiers are either adjectives or adverbs. For each concept type, all the relations that are applicable to that type are used (for adjectives and adverbs, only synonyms can be derived). The confidence score assigned to a related concept is the weight of the relation that links its synset and the synset of the original term. If the path between the two synsets consists of multiple relations, then the confidence score is the product of the weights of all relations on the path.

A breath-first search (BFS) is performed on the WordNet graph to identify related terms. Ephyra first searches for concepts that are directly related to the given term and subsequently follows longer paths until a predefined search depth is exceeded. Whenever the BFS reaches a synset that has been visited before, the weight of the new path is compared to the weight of the shortest known path. If a cheaper path has been found, the weight of the synset is updated and the search is resumed in the synset, otherwise the synset is not further expanded. Initial experiments showed that paths of length larger than 2 rarely lead to concepts with meaningful relations to the original term. Therefore, the maximum search depth has been set to 2.

4.4 Semantic Networks

WordNet provides a wide range of relations for nouns, but fewer relations that can be used to expand verbs, and their coverage is comparatively low. Furthermore, verbs often have dozens of distinct meanings in WordNet, and thus the disambiguation of verbs in an open domain is particularly difficult. A handcrafted ontology for a restricted domain, on the other hand, is a more reliable resource for the expansion

of verbs as it only covers word senses that are common in the domain. However, substantial manual effort is required to analyze a corpus for domain-specific concepts and to arrange these concepts in an ontology.

This section introduces a novel approach for automatically learning ontological knowledge from a textual corpus. At first, the corpus is annotated with a semantic parser and the annotations are transformed into a semantic network of entities and events in the corpus (Section 4.4.1). Secondly, this network is searched for cycles that indicate alternative representations of events (Section 4.4.2). Finally, the cycles are analyzed for frequent relations and are deployed to expand verbs or semantic structures as a whole (Section 4.4.3).

4.4.1 Building a Semantic Network

A semantic network is a partially connected graph and can be built from an arbitrary text corpus that has been annotated with a semantic parser. The corpus can be domain-independent or it may cover a specific domain. The network can even be built from the same resource that is subsequently used as a knowledge base for question answering. The latter ensures that fewer word senses are actively used and disambiguation errors become less frequent.

Experiments have been performed on two corpora with semantic annotations: (1) The PropBank corpus that has been manually annotated with predicate-argument structures (cf. Section 2.3), and (2) a domain-specific corpus from the Center for Nonproliferation Studies (CNS) on the proliferation of weapons of mass destruction (cf. Section 5.2) that has been annotated with the ASSERT semantic role labeling system. The PropBank corpus proved to be too small to reliably learn ontological knowledge and the most promising results were obtained with the CNS corpus. Semantic networks can also be generated from larger document collections such as the AQUAINT newswire corpus, but further enhancements of the algorithms described in the next section will be required to improve the memory efficiency.

Semantic networks are built from the entities and events that occur in the semantic annotations of the corpus. The experiments in this thesis were performed on PropBank-style predicate-argument structures. The arguments were considered as entities, whereas the predicate verbs were interpreted as events that semantically link their arguments. However, the approach is not limited to predicate-argument structures but it is also applicable to other types of semantic structures. In FrameNet annotations, the targets of the semantic frames would represent the events and the frame elements the entities.

At first, the semantic annotations are preprocessed to reduce the network size and to increase the density and eventually the recall of the network. Predicate verbs are replaced by their lemmas in WordNet and arguments are transformed into a simple normal form, i.e. they are converted to lower case and tokenized with a rule-based approach. Semantic networks exploit the redundancy in a corpus and thus it is important to recognize similar verbs and arguments. In the future, coreference resolution techniques and WordNet relations such as synonyms and hyponyms could be utilized to add additional links between similar entities and events in order to create a yet denser network.

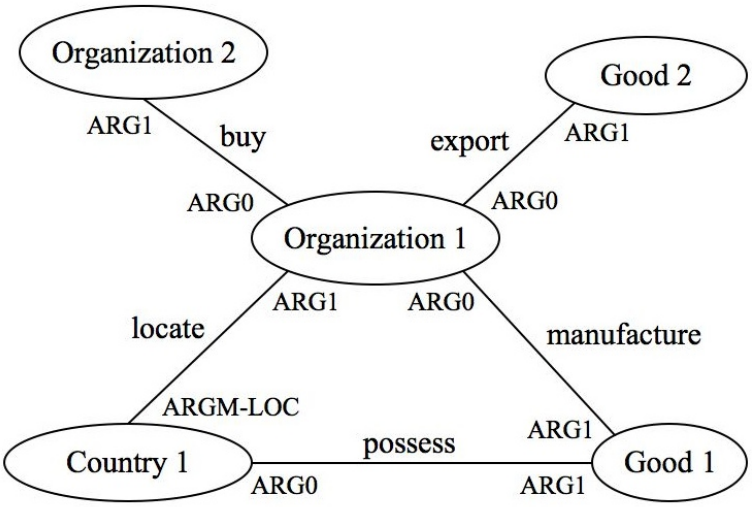

Figure 4.1: Sample semantic network with cycles.

The entities and events are subsequently loaded into a lightweight semantic network. The primary design goal for the underlying data structures was memory efficiency, thus any unnecessary overhead has been avoided. An entity consists of a unique identifier, its degree, and the events it is involved in (i.e. a list of adjacent events). An event comprises an identifier, a degree, and the entities that are part of the event along with their semantic roles (a labeled adjacency list).

Further steps have been taken to reduce the size of the network by dropping entities and events with small or excessively large degrees. The algorithm begins loading the entities and events from the semantic annotations into the network until it runs short of memory. Once the memory is exhausted, no further nodes are added but the remainder of the semantic annotations is parsed and the degrees of already loaded nodes are increased to correctly reflect the degrees in the complete network. The partial network is then reduced in an iterative process:

1. Entities that are involved in more than a predefined number of events are dropped to avoid common entities such as personal pronouns. These entities are often ambiguous and do not indicate reliable semantic relations.

2. Entities that are involved in only one event are dropped as the purpose of building a semantic network is to extract cycles and leaf nodes can never lie on a cycle.

3. Events that involve less than two entities are dropped for the same reason.

4. Steps 2 and 3 are repeated until the network cannot be further reduced.

Subsequently, the algorithm continues loading additional entities and events until the network needs to be reduced again or until finally the entire network is present in the main memory.

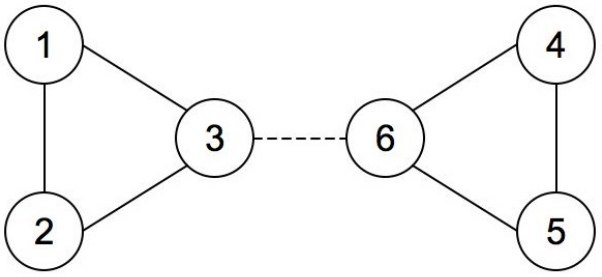

Figure 4.2: Connected but not strongly connected component.

4.4.2 Searching a Semantic Network for Cycles

The semantic network is searched for cycles that indicate semantically related events. Figure 4.1 shows a small sample network that illustrates how cycles reveal semantic relations. The network contains a cycle involving the entities *Country 1*, *Good 1* and *Organization 1*. From this cycle, one can derive the following general relation: If some organization or person X is located within a geographical entity Y and entity X manufactures a good Z, then X comes in possession of Z. Thus, the predicate-argument structure *possess(ARG0: Country 1, ARG1: X)* can be transformed into the logical form *locate(ARG1: Y, ARGM-LOC: Country 1) AND manufacture(ARG0: Y, ARG1: X)*.

The network built from the CNS corpus comprises more than 2 million nodes that need to be searched efficiently for cycles. It is hardly feasible to search the network as a whole, but it can be decomposed into subgraphs that can be searched independently. Intuitively, the graph can be split into connected components and each component can be searched separately since cycles always occur within one component. Furthermore, the search can even be restricted to strongly connected components: Since there are two distinct, simple paths between each two vertices that lie on a cycle, a cycle always forms a subset of a strongly connected component. For instance, Figure 4.2 shows a graph that is connected but not strongly connected. The graph can be divided into the two strongly connected components $\{1, 2, 3\}$ and $\{4, 5, 6\}$ by removing the edge between the vertices 1 and 2 and the two components can be searched for cycles independently.

Kosaraju's algorithm (see Section 2.5.3, Figure 2.6) is used to find all strongly connected components in a semantic network. Since the algorithm has originally been devised for directed graphs, a few changes were required to make it applicable to undirected semantic networks. Figure 4.3 shows a modified version of Kosaraju's algorithm that extracts all strongly connected components[1] from an undirected graph. The algorithm blocks all edges that have been traversed during the first DFS in the second DFS.

The connected components are searched for cycles with the algorithm from Liu and Wang (cf. Section 2.5.2, Figure 2.5). There are more efficient algorithms for finding all cycles in a graph (such as [Joh75] and [Tie70]), but this approach is sufficiently fast for finding short cycles that express meaningful semantic relations between events. In initial experiments, the algorithm was applied to an adjacency matrix built for

[1] A definition of strongly connected components in undirected graphs is given in Section 2.5.1.

Input: Undirected graph $G = (V, E)$.

Algorithm:

- Perform a DFS on G and number the vertices according to the order of completion of the recursive calls. Add each edge that is traversed to a set *Blocked*.
- Construct a new graph $G' = (V, E')$ where $E' = E \setminus Blocked$, i.e. the blocked edges are dropped.
- Perform a DFS on G', starting from the vertex with the largest order according to the first step. If the DFS does not reach all vertices then resume the search in the vertex with the highest order that has not been visited yet.
- Add each tree in the resulting forest to the set of strongly connected components S.

Output: Set S of strongly connected components in G.

Figure 4.3: Kosaraju's algorithm modified for undirected graphs.

each strongly connected component. However, the size of a component can exceed a 100,000 nodes for the CNS corpus, thus resulting in a huge but sparse matrix. Therefore, the algorithm was modified to work on the adjacency lists that are associated with each entity and event in the network, which saves memory at the expense of a slightly increased runtime of the search. Furthermore, two modifications of the algorithm from Liu and Wang were required: (1) As an undirected cycle can be traversed in two directions, the algorithm returned each cycle twice and had to be extended to ignore duplicates. (2) The FIFO queue that is used to keep track of open paths was replaced by a stack to improve the memory efficiency.

A cycle is a sequence of entities and interleaving events, each linking the two neighboring entities and the last event connecting the first and the last entity in the sequence. The entities and events are pairwise distinct. These specific instances of cycles are transformed into generic representations that comprise the events and the semantic roles of the entities that are involved, but abstract from the actual entities. Thus, a generic cycle is a sequence of events interleaved by pairs of semantic roles and can be denoted as

$$Event_1(Role_{1,n}, Role_{1,1})\ Event_2(Role_{2,1}, Role_{2,2})\ ...\ Event_n(Role_{n,n-1}, Role_{n,n})$$

where n is the number of events and entities and $Role_{i,j}$ denotes the semantic role of $Entity_j$ in $Event_i$. Several distinct cycles may be mapped to the same generic representation if they only differ in the entities but involve the same events and semantic roles.

A generic cycle can be regarded as a class of concrete cycles that express the same semantic relation between events. The frequency of each generic cycle is counted, with a higher frequency usually indicating a more reliable semantic relation. To be

able to recognize cycles that belong to the same class, a unique generic representation needs to be found for all these cycles. However, a cycle with $n > 2$ nodes has $2n$ possible generic representations: The cycle can be traversed in two directions, starting in any of the n entities. To ensure that the same generic representation is used consistently, the representation with the smallest lexicographic order according to the above notation is selected.

For each event and pair of semantic roles on a generic cycle, two additional figures are maintained: (1) the number of occurrences of this combination of event and semantic roles in the entire network and (2) the number of occurrences that lie on a cycle with the given generic representation. These figures are used in the next section to estimate confidence scores for expanded events.

4.4.3 Analysis of Cycles

A cycle in a semantic network indicates alternative representations of events and can be used directly to expand predicate-argument structures extracted from questions: For each two entities on a cycle, there are two chains of events that link the entities and that express a similar relation between the entities. Thus, if a question predicate with a pair of semantic roles can be found on a generic cycle, it can be replaced by the remaining predicates on the cycle. Consider the question "Who bought the mobile phone business from Siemens?", containing the predicate

*bought(ARG0: *missing*, ARG1: mobile phone business, ARG2: Siemens).*

The generic cycle $buy(ARG0, ARG2)$ $sell(ARG0, ARG2)$ (using the notation introduced in the previous section) suggests that the question predicate can be transformed into the alternative representation

*sold(ARG0: Siemens, ARG1 mobile phone business, ARG2: *missing*).*

In addition, the generic cycles can be analyzed for common relations between events. For instance, cycles of the form

$$Event_1(ARG0, ARG1)\ Event_1(ARG0, ARG1)$$

indicate that $Event_1$ is commutative (e.g. "collaborate"), whereas the generic cycle

$$Event_1(ARG0, ARG2)\ Event_2(ARG0, ARG2)$$

suggests that $Event_1$ is the inverse of $Event_2$ (e.g. "buy" and "sell").

The frequency of a generic cycle is used as an indicator for its reliability. If the frequency falls below a preset threshold, the cycle is not used to expand semantic structures or to derive relations between events. Confidence scores for the expanded events are estimated on the basis of the two figures introduced at the end of Section 4.4.2: The weight of a related event extracted from a generic cycle is the number of occurrences of this event and its pair of semantic roles on cycles with the given generic representation over the total number of occurrences in the network. This score reflects the likelihood that an instance of the event in the corpus has the desired meaning.

5. Experiments and Results

The previously described query generation and answer extraction techniques have been evaluated on past TREC questions. The Web was utilized as a large open-domain knowledge source and answer keys were used to judge and compare the results (cf. Section 5.1). In addition, a qualitative comparison of different resources for query expansion has been made on the CNS corpus, a static document collection on a specific domain (see Section 5.2).

5.1 Experiments on TREC Questions

The questions from the TREC 11 question answering track [Voo02] were used to evaluate the semantic question analysis and answer extraction techniques introduced in Chapter 3. Section 5.1.1 describes the test set and motivates the choice. Section 5.1.2 gives evaluation results and compares the semantic approach to the syntactic answer extractors in Ephyra. Section 5.1.3 measures the impact of term queries and expansion techniques on the search results. Finally, Section 5.1.4 compares different score combination approaches that merge answer candidates from the semantic and syntactic extractors.

5.1.1 Test Set

The TREC 11 test set comprises 500 factoid questions along with answer keys that cover correct answers found in the AQUAINT corpus by human assessors and systems that participated in the evaluation. The answer keys have been extended to cover additional correct answers found in the Web with the semantic and syntactic extractors. However, new answers were only accepted as correct if one of the following conditions was satisfied: (1) The answer was more precise than the answer keys, (2) the answer was more up to date, or (3) no answer keys were available for the question and the answer could be verified as correct. Still for the 53 questions listed in Appendix B no correct answer was found and no answer keys were available.

TREC 11 was chosen as the test set because it contains only factoid questions and the questions are independent and self-contained. More recent TREC evaluations

Technique	Questions Answered	Questions Correct	Precision	Recall
Answer type analysis	361	173	0.479	0.387
Pattern learning	293	104	0.355	0.233
Semantic parsing	154	90	0.584	0.201

Table 5.1: Precision and recall on TREC 11 questions with correct answers.

also include list questions and definitional questions, and the questions are organized in series of questions that refer to the same topic. Furthermore, the questions contain coreferences and need to be answered in the context of the topic and preceding questions. The semantic parsing approach is also applicable to recent TREC questions and initial experiments have been performed on the TREC 15 test set. However, the additional sources of error resulted in a lower accuracy of both the baseline system and the semantic approach and made comparisons more difficult.

5.1.2 Comparison of Extraction Techniques

The semantic approach for question analysis and answer extraction has been compared against Ephyra's two syntactic answer extraction approaches: answer type analysis (Section 2.2.2.1) and pattern learning and matching (Section 2.2.2.2). The configuration of Ephyra was similar to the setup used in the TREC 15 evaluation, but it included recent enhancements, particularly of the answer selection pipeline, the answer type classification component and the named entity recognizers. The test set used for the following experiments only comprised the 447 questions with known correct answers.

Table 5.1 shows evaluation results for all three approaches. *Questions Answered* refers to the number of questions the respective technique returned at least one answer for, while *Questions Correct* counts the number of questions that were answered correctly by the top-ranked result returned by the extractor. The precision and recall have been calculated as follows:

$$\text{Precision} = \frac{\text{Questions Correct}}{\text{Questions Answered}}$$

$$\text{Recall} = \frac{\text{Questions Correct}}{\text{Questions Total } (\hat{=} \ 447)}$$

The results show that the semantic parsing approach has the highest precision, but it suffers from a low recall. The higher precision is not surprising as the semantic extractor ensures that only answers which match the syntactic pattern of the question are returned, while the answer type approach merely extracts entities of the expected type that cooccur with question terms.

Furthermore, the semantic approach is capable of answering questions with unknown answer types, such as:

- How did Molly Shannon's mother die? – car accident (Question 1495)
- What does CPR stand for? – cardiopulmonary resuscitation (Question 1516)

Technique	NIL Precision	NIL Recall
Answer type analysis	0.175	0.340
Pattern learning	0.186	0.660
Semantic parsing	0.115	0.717

Table 5.2: NIL precision and recall on TREC 11 questions.

Technique	Questions Correct	Accuracy
Answer type analysis	76	0.494
Pattern learning	38	0.247
Semantic parsing	90	0.584

Table 5.3: Accuracy on TREC 11 questions covered by semantic parsing approach.

- What did Charles Babbage invent? - computer (Question 1552)
- What do grasshoppers eat? – plants (Question 1590)
- What does "E Pluribus Unum" mean? - one out of many (Question 1661)

On the other hand, the semantic approach is only applicable to questions that contain parsable predicates (cf. Section 3.1) and it follows a more conservative answer extraction strategy, which explains the lower recall. A large number of questions could not be parsed because they only contained one of the highly ambiguous verbs "to be", "to do" or "to have". One question ("How many electoral college votes in Tennessee?", Question 1537) did not contain any verb. Another common source of error were false POS labels, which corrupted the transformation of the question into a statement. In addition, ASSERT frequently failed to correctly recognize and label the arguments of the predicates.

The precision and recall in recognizing questions with no correct answers (NIL precision and recall [Voo02]) is shown in Table 5.2. Naturally, the semantic approach has the highest recall but the lowest precision because it left the most questions unanswered.

One could argue that the semantic parsing approach focused on easier questions and therefore yielded a higher precision. To allow a fair comparison, the accuracy of the three approaches has been compared on the subset of the TREC 11 questions answered by the semantic approach, listed in Appendix C (154 questions). The results are given in Table 5.3.

This comparison shows that the semantic parsing approach clearly outperforms the syntactic techniques on a subset of the TREC 11 questions. On the other hand, the simple syntactic approach based on answer type analysis has a higher coverage and the imprecise answer extraction is partially compensated for by the redundancy of the Web. It is therefore reasonable to combine the different techniques by using the semantic extractor as the primary approach and the syntactic extractors as a fallback if the semantic approach fails to return any answers. Table 5.4 shows evaluation results for a combination of the semantic parsing and answer type approaches, as well as all three approaches. The results indicate that the combination of multiple extractors indeed yields a higher recall than any individual technique.

Combination	Questions Answered	Questions Correct	Precision	Recall
Semantic parsing, Answer type analysis	388	187	0.482	0.418
Semantic parsing, Answer type analysis, Pattern learning	432	192	0.444	0.430

Table 5.4: Precision and recall of combined approaches on TREC 11 questions with correct answers.

Types of Queries	Accuracy
Terms	0.358
Keywords	0.376
Keywords + Terms	0.382
Reformulations	0.266
Reformulations + Terms	0.370
Keywords + Reformulations	0.388
Keywords + Reformulations + Terms	0.392

Table 5.5: Evaluation of query generation techniques on TREC 11 questions.

5.1.3 Impact of Term Extraction and Expansion on Search

Further experiments have been performed to find an optimal combination of query types and to evaluate the impact of the query expansion techniques on the relevance of the search results. Google was used to search the Web and the first 100 snippets were retrieved for each query. Answer candidates were always extracted with the answer type approach to obtain comparable results for the different query types. The test set consisted of all TREC 11 questions, including the questions without answer keys, thus the performance scores are slightly lower than in the previous section.

Table 5.5 shows the accuracy of the answer type based extraction approach when applied to Google snippets fetched with keyword queries, term queries, reformulation queries and combinations of different query types. The accuracy is the ratio of the number of question that were answered correctly by the top-ranked result over the total number of questions.

The keyword queries consisted of the content words in the question, function words and duplicates were dropped (see Section 3.2). Term queries were formed by extracting terms from the question and expanding them with related concepts (cf. Section 3.2 and Chapter 4). Compound terms were looked up in WordNet or extracted with Ephyra's named entity recognizers. WordNet was also used to expand the terms with synonyms. Reformulation queries were obtained by rephrasing the question into a statement (cf. Section 4.1.3 in [Sch05]). For instance, the question "When did Bob Marley die?" (Question 1143) was transformed into "Bob Marley died in" and "Bob Marley died on", strings that frequently appear in sentences that answer the question.

The following conclusions can be drawn from the evaluation results:

1. The term queries do not outperform simple keyword queries. However, the accuracy of keywords, reformulations as well as their combination can be improved by adding term queries. This improvement does not result from an increased number of snippets. The accuracy actually decreases if the number of search results per query is further increased. For instance, if the number of search results for keyword queries is increased from 100 to 300, the accuracy drops from 0.376 to 0.368.

2. There is no optimal query type that works equally well for all questions. Keyword queries are flexible but often do not preserve the semantics of the question. Term extraction and expansion, on the other hand, is more error-prone as it involves the detection of named entities and the disambiguation of word senses. Reformulations of the question rely on redundancy and are only effective for short questions on topics that are widely covered in the knowledge source.

3. A combination of different, complementary types of queries yields the highest accuracy. As long as one query returns relevant results, the noise caused by additional inappropriate queries has little effect on the performance of the answer extractor.

5.1.4 Score Normalization and Combination

The answer candidates retrieved with the semantic and syntactic extraction strategies were combined in two steps:

1. The confidence scores were normalized with an AdaBoost classifier, using a decision tree as the underlying weak learning algorithm. Different reordering techniques (*Averaging*, *Rescaling* and *Resorting*) were used to ensure that the classifier does not alter the order of the results, cf. Section 3.5.2.2.

2. The normalized scores of answer candidates found with multiple extractors were combined with common approaches that have successfully been applied in document retrieval systems (*CombMIN*, *CombMED*, *CombMAX*, *CombSUM*, *CombANZ* and *CombMNZ*) and a score combination scheme that is based on complementary probabilities (*CombCP*), cf. Section 3.5.3.

Table 5.6 shows the accuracy of the combined semantic and syntactic answer extractors when used with different reordering and score combination techniques. The test set comprised all TREC 11 questions with answer keys. For each question, all extractors that retrieved at least one answer were considered. If a particular answer candidate was not found by one of these extractors, the confidence score for this extractor was set to 0.

The results show that the combination of Rescaling and CombMNZ yields the highest accuracy (48.1%). This setup clearly outperforms the simple combination approach discussed in Section 5.1.2 that applies the answer extractors in a fixed order according to their precision (43.0% accuracy). However, all these combination techniques have

	MIN	MED	MAX	SUM	ANZ	MNZ	CP
Averaging	0.438	0.456	0.421	0.450	0.416	0.454	0.306
Rescaling	0,425	0,474	0,450	0,477	0,436	0,481	0,470
Resorting	0,398	0,385	0,324	0,374	0,295	0,416	0,349

Table 5.6: Accuracy of reordering and score normalization techniques on TREC 11 questions with correct answers.

in common that they only take the confidence scores of the answers into account and that the combined score does not depend on the actual answers. By identifying similar answers in the set of answer candidates, and by validating candidates with external semantic resources, the reliability of the combined scores could be further improved. This is one possible area for future research (cf. Section 6.2).

5.2 Experiments on CNS Corpus

The resources for query expansion described in Chapter 4 have been tested on a document collection created by the Center for Nonproliferation Studies (CNS). The corpus comprises about 200 MB of documents and deals with the proliferation of weapons of mass destruction. The Indri search engine from the Lemur toolkit [Lem] was utilized to build an index and to retrieve documents from the corpus.

A collection of 199 scenario-based questions [BN06] that has been created specifically for the CNS corpus was used to test the domain-specific CNS ontology (cf. Section 4.2), WordNet (see Section 4.3) and a semantic network that has been built from the CNS corpus (cf. Section 4.4).

Many of the questions in the collection are complex questions that cannot be answered with Ephyra and the test set only provides document-level relevance judgements, thus only a qualitative comparison of the query expansion techniques could be performed. The following sample questions illustrate how the three semantic resources expand term queries with related concepts:

(1) *What companies are competing to export nuclear power technologies to Indonesia?*

- Terms expanded with CNS ontology:
  companies competing (export OR ship OR transfer OR send OR supply OR sell OR offer OR provide OR deliver) nuclear power technologies Indonesia
- Terms expanded with WordNet:
  companies (competing OR vieing OR contending) export ("nuclear power" OR "atomic power") (technologies OR engineering) (Indonesia OR "Republic of Indonesia" OR "Dutch East Indies")
- Terms expanded with semantic network:
  companies competing (export OR ship OR transfer OR supply OR manufacture OR produce OR sell OR provide OR deliver) nuclear power technologies Indonesia

(2) *Does Iraq possess yellowcake?*

- Terms expanded with CNS ontology:
  Iraq (possess OR keep OR retain OR own OR provide OR complete OR obtain OR construct OR offer OR hold OR transfer OR procure OR send OR develop OR stockpile OR assemble OR ship OR build OR manufacture OR maintain OR supply OR produce OR use OR deploy OR export) yellowcake
- Terms expanded with WordNet:
  (Iraq OR "Republic of Iraq" OR Al-Iraq OR Irak) possess yellowcake
- Terms expanded with semantic network:
  Iraq (possess OR acquire OR manufacture OR use OR own) yellowcake

(3) *Why is Syria acquiring ballistic missiles?*

- Terms expanded with CNS ontology:
  Syria (acquiring OR building OR producing OR preparing OR procuring OR completing OR taking OR buying OR establishing OR developing OR purchasing OR obtaining OR assembling OR accepting OR constructing OR rebuilding OR manufacturing OR receiving OR creating OR implementing) "ballistic missiles"
- Terms expanded with WordNet:
  (Syria OR "Syrian Arab Republic") (acquiring OR getting) "ballistic missiles"
- Terms expanded with semantic network:
  Syria (acquiring OR receiving OR retaining OR buying OR purchasing OR manufacturing OR developing OR possessing OR using OR producing) "ballistic missiles"

6. Conclusion and Outlook

6.1 Summary

Two types of semantic resources have been integrated in the Ephyra framework for open-domain question answering to improve its accuracy on factoid questions: (1) A semantic parser is used to analyze questions and to extract answer candidates from semantically similar phrases in the knowledge source and (2) dictionaries and ontologies are deployed to extract terms from questions and answer sentences and to expand them with related concepts.

Semantic structures are extracted from both the question and the corpus and are subsequently compared to identify phrases that are semantically similar to the question. Relevant documents are retrieved from the corpus using different types of queries ranging from simple bags of keywords over compound terms and synonyms to specific queries built from semantic representations of the question. A new similarity metric for semantic structures has been proposed that breaks down high-level structures into terms, enriches the terms with alternative representations, compares the individual terms and aggregates their similarities into an overall similarity score. The selection of candidate sentences based on semantic similarities is flexible and robust to parsing errors. A careful selection of candidate sentences and caching techniques for intermediate results have been necessary to make it computationally feasible.

Multiple ontologies have been used to extract and expand terms that represent meaningful semantic units. WordNet is deployed as an open-domain resource, while a framework for domain-specific ontologies allows integrating semantic resources that provide expert knowledge on specific topics. Furthermore, a new technique is proposed for automatically extracting ontological knowledge from a textual corpus. The corpus is annotated with predicate-argument structures, which are subsequently transformed into a semantic network that reveals relations between the entities and events in the corpus.

Evaluation results have shown that these semantic techniques outperform Ephyra's syntactic answer extractors on a large portion of the questions from past TREC

evaluations. However, not all questions contain parsable semantic structures and current semantic parsers are still prone to errors, thus the overall recall of the syntactic extractors is higher. It has further been shown that a combined approach of semantic and syntactic techniques yields the highest accuracy and outperforms any individual approach.

Different techniques have been suggested that allow merging answer candidates from multiple extractors. As different extraction approaches use incompatible scoring mechanisms, the confidence scores of answer candidates need to be normalized before answers can be combined. Several statistical models and feature combinations for score normalization have been evaluated, including adaptive boosting and maximum entropy models. The normalized scores are combined with established techniques that have successfully been applied in document retrieval systems.

6.2 Future Work

A major bottleneck of the previously described semantic question analysis and answer extraction approach is the coverage and reliability of the semantic parser. By integrating multiple semantic role labeling (SRL) systems, the robustness can be improved significantly. A combination of different systems is particularly beneficial if the systems use complementary syntactic structures, such as full and partial syntax. This has been one of the key findings in the CoNLL-2005 evaluation of SRL systems [CM05] and has been verified in [MSCT05]. [PRY05] could achieve a significant improvement in recall by combining two SRL systems, one based on Charniak's parser, the other deploying Collins' parser.

The coverage of a semantic parser that is based on predicate-argument structures can be further improved by transforming some of the questions with ambiguous predicates (such as "to be" or "to have") into questions with parsable predicates (cf. Section 3.1). For instance, the question "Who was the winner of the competition?", which only contains the ambiguous predicate "was", can be rephrased to "Who won the competition?".

Yet there remains a significant portion of questions and answer sentences with semantic structures that do not fit into the schema of predicate verbs and arguments. For instance, SRL systems are incapable of recognizing nominalized predicates such as in the sentence "The current US president is the son of a former president.", where the noun "son" describes a semantic relation between two entities. It would therefore be desirable to cover a wider range of semantic structures. The FrameNet lexical database [REP+] introduces semantic frames that are not limited to verbs and thus provide a higher coverage. However, no parsers are available yet that extract frame structures with a reliability comparable to the best parsers for PropBank-style annotations.

In addition, semantic structures can be derived from compound noun phrases. For instance, the phrase "former seven-time Formula One champion Michael Schumacher" contains various semantic relations: It states that Michael Schumacher has been a Formula One pilot and that he has won the championship seven times in his career.

Furthermore, the answer extraction approach presented in this thesis does not take the context of the extracted semantic structures into account. Statements in the

knowledge source do not necessarily represent common knowledge, but they may also express the personal opinion of the author or reproduce a statement of a third person. The source of a statement may even be entirely fictitious, e.g. a review of a novel. Therefore, the context established by the source document should be considered. In particular, a statement may appear in indirect speech ("The spokesman denied that ..."), or it may be qualified by surrounding statements ("This is a common misconception."). However, I believe that the performance of the currently available semantic parsers hardly permits a profound context analysis. On the other hand, a redundant knowledge source that contains multiple representations of an answer can compensate for many of these errors.

In Section 3.2, I emphasized the benefits of structured retrieval techniques on text corpora with semantic annotations. In addition to improving the runtime performance, structured queries could be used to find answers by combining evidence from multiple documents (cross-document answer extraction). For instance, one could search for all organizations X in the document collection that satisfy the constraints imposed by the predicates *based(ARG1: X, ARGM-LOC: Japan)* and *manufacture(ARG0: X, ARG1: SUV)* to obtain a list of Japanese car makers that offer SUVs.

Semantic parsing techniques can also be utilized in an interactive question answering system to react dynamically to user questions. When asking an ambiguous question, the user can be queried for additional information that is missing in the semantic representation of the question. For instance, the question "What famous person was born in Salzburg?" does not specify a date for the predicate "born" and is therefore highly ambiguous. The user could be asked for this missing argument to narrow down the number of answer candidates.

The score combination techniques presented in Section 3.5.3 have been designed for merging extensive lists of documents, where a syntactic or semantic analysis of the individual results is infeasible. Factoid question answering, however, deals with comparatively small numbers of precise answers, which allows for answer validation and combination techniques that are not solely based on confidence scores but also take syntactic and semantic features of the answers into account. The JAVELIN system applies a generic approach for answer selection that is based on a probabilistic framework [KSN07]. Two types of features are combined to estimate confidence scores for the answer candidates: Answer validation features deploy external semantic resources to verify answer candidates and answer similarity features exploit redundancy among the candidates (see also Section 1.4.3). We are currently integrating this probabilistic framework in Ephyra and we expect to further improve the accuracy of the combined approach of syntactic and semantic answer extraction techniques (cf. Section 5.1.4).

The current approach for term expansion oversimplifies the word sense disambiguation problem. In WordNet, the most frequent synset of a term is chosen, whereas the other domain-specific resources assume that a word has a unique sense in each domain. This practical approach is rather effective, but it can result in false assumptions if a word is used in a rare sense. Disambiguation errors frequently occur for verbs and are particularly harmful at the query generation stage, where they may result in the retrieval of irrelevant documents. One could consider the word

senses assigned to predicate verbs by the semantic role labeling system to obtain more reliable results.

Semantic networks are a promising approach for automatically learning semantic relations from unstructured corpora and different directions for future research are possible: (1) The recall of semantic networks currently suffers from their sparsity. Syntactic similarity measures, WordNet synonyms and coreference resolution techniques would allow identifying nodes that refer to the same entity. These notes can be merged to create a smaller, yet denser network. (2) Semantic networks are not limited to event semantics but also allow conclusions about relations between entities. For instance, a network can be analyzed for entities that occur in the same or similar events. (3) Further performance optimizations will be required to build semantic networks from larger corpora. A network for the AQUAINT corpus or the newly released AQUAINT-2 corpus could be evaluated on TREC test sets.

Finally, the ontologies that have been used in this thesis can be combined with additional resources for query expansion. A wide range of domain-specific ontologies are readily available on the Web and could be integrated in the existing framework. An appropriate ontology for term expansion could be selected on the basis of the terms that occur in a question or the context established by previous questions from the same user.

A. Models for Score Normalization

Rounds	**Features**		
	Score, Extractor, ATypes, Mean, Max, Min	Score, Extractor, ATypes, Num, Max, Min	Score, Extractor, ATypes, Num, Mean, Max, Min
10	0.164	0.169	0.140
20	0.193	0.181	0.163
30	0.193	0.194	0.194
40	0.199	0.181	0.193
50	0.201	0.189	0.205
60	0.207	0.197	0.200
70	0.215	0.197	0.184
80	0.207	0.205	0.188
90	0.207	0.197	0.190
100	0.198	0.201	0.191
110	0.190	0.206	0.185
120	0.191	0.199	0.184
130	0.189	0.190	0.180
140	0.192	0.197	0.182
150	0.194	0,197	0.180

Table A.1: F_1 measures for an AdaBoost learner with different numbers of rounds and different feature combinations.

Features	**Model**				
	AdaBoost	AdaBoost L	Balanced Winnow	Decision Tree	5NN
Score	0,072	0,065	zero recall	0,081	N/A
Score, Extractor	0,076	0,056	zero recall	0,060	N/A
Score, Extractor, ATypes	0,096	0,076	0,005	0,060	N/A
Score, Extractor, ATypes, Num, Mean	0,149	0,109	0,061	0,060	N/A
Score, ATypes, Num, Mean, Max, Min	0,138	0,083	0,035	0,076	N/A
Score, Extractor, Num, Mean, Max, Min	0,149	0,081	0,066	0,064	N/A
Score, Extractor, ATypes, Mean, Max, Min	0,164	0,104	0,038	0,060	N/A
Score, Extractor, ATypes, Num, Max, Min	0,169	0,478	0,074	0,060	N/A
Score, Extractor, ATypes, Num, Mean, Min	0,132	0,097	0,065	0,060	N/A
Score, Extractor, ATypes, Num, Mean, Max	0,164	0,120	0,056	0,060	N/A
Score, Extractor, ATypes, Num, Mean, Max, Min	0,140	0,126	0,077	0,060	0,102
	Margin Perceptron	MaxEnt	Naive Bayes	Negative Binomial	Voted Perceptron
Score	0,094	0,055	zero recall	zero recall	0,139
Score, Extractor	zero recall	0,055	zero recall	0,096	0,156
Score, Extractor, ATypes	0,088	0,058	0,001	0,139	0,150
Score, Extractor, ATypes, Num, Mean	zero recall	0,058	0,001	0,009	0,074
Score, ATypes, Num, Mean, Max, Min	0,026	0,053	zero recall	0,009	0,077
Score, Extractor, Num, Mean, Max, Min	0,027	0,046	zero recall	0,011	0,077
Score, Extractor, ATypes, Mean, Max, Min	0,102	0,057	0,001	0,008	0,008
Score, Extractor, ATypes, Num, Max, Min	0,026	0,060	0,001	0,011	0,078
Score, Extractor, ATypes, Num, Mean, Min	zero recall	0,058	0,001	0,009	0,070
Score, Extractor, ATypes, Num, Mean, Max	0,053	0,051	0,001	0,008	0,077
Score, Extractor, ATypes, Num, Mean, Max, Min	0,028	0,073	0,001	0,015	0,077

Table A.2: F_1 measures for different models and feature combinations for score normalization.

B. TREC 11 Questions - No Answers

Table B.1: TREC 11 questions without correct answers in the document collection.

ID	Question	Correct
1420	How high is Mount Kinabalu?	yes
1422	What two European countries are connected by the St. Gotthard Tunnel	yes
1430	How old do you have to be to get married in South Carolina?	yes
1437	How many states were still united after the southern states seceded?	no
1445	When is Snoop Dog's birthday?	yes
1448	What is the fear of lightning called?	yes
1461	What country's leader was awarded the 2000 Nobel Peace Prize?	no
1468	If something has petrified, what has it turned into?	yes
1485	What is slang for a "five dollar bill"?	no
1486	Where did Roger Williams, pianist, grow up?	no
1511	Who said "Music hath charm to soothe the savage beast"?	yes
1543	How tall was Judy Garland?	yes
1548	What was the first name of the Gehrig who played for the New York Yankees?	yes
1558	How much does it cost to register a car in New Hampshire?	yes
1571	How much copper is in a penny?	yes
1577	What Burns poem does "the best laid plans of mice and men" come from?	yes
1581	What anniversary is the 20th?	yes
1587	What did Sherlock Holmes call the street gang that helped him crack cases?	yes
1594	Which long Lewis Carroll poem was turned into a musical on	no

Continued on next page.

Table B.1 – Continued from previous page.

ID	Question	Correct
	the London stage?	
1602	Who holds the record as the highest paid child performer?	no
1608	What is the appropriate gift for a 10th anniversary?	yes
1611	What is the lifespan of a house fly?	yes
1632	How much protein is in tofu?	yes
1642	What do you call a baby sloth?	yes
1644	How far away from the sun is Saturn?	yes
1648	What does TB stand for in baseball stats?	yes
1656	How fast does an Iguana travel (mph)?	yes
1696	When was the pencil first invented?	no
1670	Who was the Union general in the Battle of Winchester?	yes
1684	What card game uses only 48 cards?	no
1703	What award did Sterling North's book "Rascal" win in 1963?	yes
1707	What is the ranger's name in Yogi Bear cartoons?	yes
1709	What is the August birthstone?	yes
1725	Where did David Ogden Stiers get his undergraduate degree?	yes
1738	Where was the F Troop stationed?	no
1743	Which state has the longest coastline on the Atlantic Ocean?	yes
1761	How many black keys are on the piano?	yes
1765	What airport is LCY?	yes
1786	What were the most points Michael Jordan scored in a game?	no
1792	How far is it from Buffalo, New York to Syracuse, New York?	yes
1816	How old must you be to become President of the United States?	yes
1817	Who was the first president to speak on the radio?	yes
1832	What did Walter Cronkite say at the end of every show?	no
1841	What's the final line in the Edgar Allen Poe poem "The Raven?"	yes
1842	How much did a quart of milk cost in the 1930's?	yes
1854	What was William Shakespeare's occupation before he began to write plays?	yes
1860	What Broadway musical is the song "The Story is Me" from?	yes
1868	What capital is on the Susquehanna River?	yes
1869	In the Bible, who was Jacob's mother?	yes
1875	What county is St. Paul, Minnesota in?	yes
1877	What is the name of a newborn swan?	yes
1887	What was the name of the first enclosed baseball park?	yes
1893	What American general is buried in Salzburg?	yes

C. TREC 11 Questions - Covered

Table C.1: TREC 11 questions covered by the semantic question analysis and answer extraction approach.

ID	Question	Correct
1394	In what country did the game of croquet originate?	no
1398	What year was Alaska purchased?	yes
1400	When was the telegraph invented?	yes
1403	When was the internal combustion engine invented?	no
1406	When did the story of Romeo and Juliet take place?	no
1407	When did the shootings at Columbine happen?	yes
1409	Which vintage rock and roll singer was known as "The Killer"?	no
1411	What Spanish explorer discovered the Mississippi River?	yes
1415	Where does the vice president live when in office?	no
1416	When was Wendy's founded?	yes
1424	Who won the Oscar for best actor in 1970?	no
1428	Who won the Nobel Peace Prize in 1992?	no
1431	Who starred in "The Poseidon Adventure"?	yes
1434	What site did Lindbergh begin his flight from in 1927?	no
1443	When did Bob Marley die?	yes
1444	What female leader succeeded Ferdinand Marcos as president of the Philippines?	no
1446	How did Mahatma Gandhi die?	no
1449	What college did Magic Johnson attend?	yes
1451	Where was the first McDonalds built?	no
1453	Where was the first J.C. Penney store opened?	no
1455	The Hindenburg disaster took place in 1937 in which New Jersey town?	no
1457	Who succeeded Ferdinand Marcos?	yes
1465	What company makes Bentley cars?	yes
1469	When did Alexandra Graham Bell invent the telephone?	yes
1470	When did president Herbert Hoover die?	yes

Continued on next page.

Table C.1 – Continued from previous page.

ID	Question	Correct
1472	How do you say "house" in Spanish?	no
1473	When was Lyndon B. Johnson born?	yes
1477	What are wavelengths measured in?	no
1479	Who composed "The Messiah"?	yes
1493	When was Davy Crockett born?	yes
1495	How did Molly Shannon's mother die?	yes
1502	What year was President Kennedy killed?	yes
1505	What is the currency used in China?	yes
1516	What does CPR stand for?	yes
1519	Where was Hans Christian Anderson born?	yes
1522	What are the headpieces called that the Saudi Arabians wear?	no
1527	When did the 6-day war begin?	yes
1528	Where did Kublai Khan live?	yes
1531	What does NASDAQ stand for?	no
1534	The sun is mostly made up of what two gasses?	no
1545	What is a female rabbit called?	no
1546	What year was the movie "Ole Yeller" made?	no
1549	What year was the PC invented?	no
1551	What does DNA stand for?	yes
1552	What did Charles Babbage invent?	yes
1553	Who makes Magic Chef refrigerators?	no
1561	When was the first patent filed on the ice cream cone?	yes
1562	Where did the U.S. Civil War begin?	no
1563	Who started the Protestant reformation?	yes
1564	When did Led Zeppelin appear on BBC?	no
1566	What famous Spanish poet died in Spain's Civil War?	no
1567	When did the Black Panther party start in California?	yes
1569	When did the Vietnam War end?	yes
1572	For whom was the state of Pennsylvania named?	yes
1574	When did "The Simpsons" first appear on television?	no
1580	What name is given to the science of map-making?	no
1583	What is the text of an opera called?	no
1584	When did George W. Bush get elected as the governor of Texas?	yes
1586	When did the Golden Gate Bridge get finished?	yes
1588	When was Apollo 11 launched?	yes
1590	What do grasshoppers eat?	yes
1591	What percentage of the population is left handed?	no
1592	When was the Reichstag burned down?	yes
1593	What percent of Egypt's population lives in Cairo?	no
1596	What year did Mussolini seize power in Italy?	yes
1601	When did Einstein die?	yes
1610	Who signed the Declaration of Independence from Vermont?	yes
1617	When did the Klondike gold rush occur?	no
1618	Where is bile produced?	no
1619	Which baseball star stole 130 bases in 1982?	yes

Continued on next page.

Table C.1 – Continued from previous page.

ID	Question	Correct
1620	What year was the first Macy's Thanksgiving Day Parade held?	yes
1624	What year did "New Coke" come out?	no
1627	What is written on the U.S. tomb of the unknown soldier?	no
1629	Where is Mae West buried?	no
1630	When was the dog domesticated?	no
1636	When was the battle of Chancellorsville fought?	no
1640	Who founded Taoism?	no
1643	Who founded Rhode Island?	yes
1645	How much is the international space stations expected to cost?	no
1646	When was the first atomic bomb dropped?	yes
1649	What year did the shuttle Challenger explode?	yes
1652	When did the United States enter World War II?	yes
1655	Where was Abraham Lincoln born?	yes
1658	What year was Robert Frost born?	yes
1659	Where was Bob Dylan born?	yes
1661	What does "E Pluribus Unum" mean?	yes
1662	When was Jerusalem invaded by the general Titus?	no
1663	What are the people who make fireworks called?	no
1676	When was water found on Mars?	no
1681	When did Houdini die?	yes
1682	When was Martin Luther King Jr. born?	yes
1683	What name is horror actor William Henry Pratt better known by?	yes
1686	Who defeated the Spanish armada?	yes
1687	What president declared Mothers' Day?	yes
1688	What year was the gateway arch built?	no
1690	What does DEA stand for?	yes
1691	Where was the movie "Somewhere in Time" filmed?	no
1693	When was Jackie Robinson born?	yes
1698	When was Julius Caesar born?	yes
1701	Where was President Lincoln buried?	yes
1711	When was the first jet invented?	no
1712	Who invented the fishing reel?	no
1714	What province in Canada is Niagara Falls located in?	yes
1718	When was the White House built?	yes
1722	What year did poet Emily Dickinson die?	yes
1727	What does the abbreviation WASP mean?	no
1731	How often does the United States government conduct an official population census?	yes
1734	How do you say "pig" in Spanish?	no
1741	What author wrote under the pen name "Boz"?	yes
1744	What car company invented the Edsel?	yes
1745	What are hiccups caused by?	yes
1746	Who stabbed Monica Seles?	yes
1747	Where is the national hurricane center located?	no
1749	When was Sputnik launched?	yes

Continued on next page.

Table C.1 – Continued from previous page.

ID	Question	Correct
1753	When was the Vietnam Veterans Memorial in Washington, D.C. built?	yes
1754	When did the Persian Gulf War occur?	yes
1757	When did the battle of Iwo Jima take place?	yes
1759	Who wrote "Fiddler on the Roof?"	yes
1760	Where was C.S. Lewis born?	yes
1767	When was the first Ford Mustang made?	yes
1772	Who invented the cotton gin?	yes
1774	What does HTML stand for?	yes
1775	What is a group of antelope called?	no
1778	When did Walt Disney die?	yes
1783	What country are Volvo automobiles made in?	no
1790	What country is the holy city of Mecca located in?	yes
1791	When did they put Mir down?	no
1797	How did Adolf Hitler die?	no
1798	On what continent is Egypt located?	yes
1800	Which president was sworn into office on an airplane?	yes
1803	When did Willis Haviland Carrier make the air conditioner?	no
1809	When was the Buckingham Palace built in London, England?	no
1810	Where are the British Crown jewels kept?	no
1811	When was penicillin first used?	no
1813	When were the first postage stamps issued in the United States?	yes
1818	Where did Golda Meir grow up?	yes
1824	Which planet did the spacecraft Magellan enable scientists to research extensively?	no
1826	Which film received the first best picture Academy Award?	no
1827	Where was the battle of Alamo fought?	no
1834	Which disciple received 30 pieces of silver for betraying Jesus?	yes
1837	What year was Ebbets Field, home of Brooklyn Dodgers, built?	yes
1839	What country was formed in 1948?	yes
1843	In what month are the most babies born?	yes
1845	What province is Calgary located in?	yes
1856	What city is known as the rubber capital of the world?	yes
1861	Where was Bill Gates born?	yes
1863	Who said "I have not begun to fight!"?	no
1865	What is the major crop grown in Arizona?	no
1871	How much gravity exists on Mars?	no
1872	How did Eva Peron die?	no
1878	What year was the phonograph invented?	yes
1880	When was King Louis XIV born?	yes
1888	What year was the light bulb invented?	yes
1892	Where does cinnamon come from?	yes

References

[Ber96] A. Berger. A brief Maxent tutorial. http://www.cs.cmu.edu/~aberger/html/tutorial/tutorial.html, 1996.

[BN06] M.W. Bilotti and E. Nyberg. Evaluation for scenario question answering systems. *Proceedings of the International Conference on Language Resources and Evaluation*, 2006.

[BOCN07] M. Bilotti, P. Ogilvie, J. Callan, and E. Nyberg. Structured retrieval for question answering. *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development on Information Retrieval*, 2007.

[CM05] X. Carreras and L. Marquez. Introduction to the CoNLL-2005 shared task: Semantic role labeling. *Proceedings of the Ninth Conference on Computational Natural Language Learning*, 2005.

[Coh04] W.W. Cohen. Minorthird: Methods for identifying names and ontological relations in text using heuristics for inducing regularities from data. http://minorthird.sourceforge.net/, 2004.

[DLK06] H.T. Dang, J. Lin, and D. Kelly. Overview of the TREC 2006 question answering track. *Proceedings of the Fifteenth Text REtrieval Conference*, 2006.

[EP06] K. Erk and S. Pado. Shalmaneser - a flexible toolbox for semantic role assignment. *Proceedings of LREC*, 2006.

[Fel98] C. Fellbaum. *WordNet: An Electronic Lexical Database*. The MIT Press, 1998.

[FGM05] J.R. Finkel, T. Grenager, and C. Manning. Incorporating non-local information into information extraction systems by Gibbs sampling. *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, 2005.

[FS93] E. Fox and J. Shaw. Combination of multiple searches. *Proceedings of the Second Text REtrieval Conference*, 1993.

[FS96] Y. Freund and R.E. Schapire. A decision-theoretic generalization of online learning and an application to boosting. *Journal of Computer and System Sciences, 55(1):119-139*, 1996.

[FS99] Y. Freund and R.E. Schapire. A short introduction to boosting. *Journal of Japanese Society for Artificial Intelligence, 14(5):771-780, September*, 1999.

[Hor87] J.D. Horton. A polynomial-time algorithm to find the shortest cycle basis of a graph. *SIAM Journal on Computing, 16(2):358-366*, 1987.

[IFR01] A. Ittycheriah, M. Franz, and S. Roukos. IBM's statistical question answering system – TREC-10. *Proceedings of the Tenth Text REtrieval Conference*, 2001.

[Joh75] D.B. Johnson. Finding all the elementary circuits of a directed graph. *SIAM Journal on Computing, 4(1):77-84*, 1975.

[KP02] P. Kingsbury and M. Palmer. From TreeBank to PropBank. *Proceedings of LREC*, 2002.

[KP03] P. Kingsbury and M. Palmer. PropBank: The next level of TreeBank. *Proceedings of Treebanks and Lexical Theories*, 2003.

[KPRY05] P. Koomen, V. Punyakanok, D. Roth, and W. Yih. Generalized inference with multiple semantic role labeling systems. *Proceedings of the Ninth Conference on Computational Natural Language Learning*, 2005.

[KSN07] J. Ko, L. Si, and E. Nyberg. A probabilistic framework for answer selection in question answering. *Proceedings of HLT/NAACL*, 2007.

[Lee97] J.H. Lee. Analyses of multiple evidence combination. *Proceedings of the Twentieth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1997.

[Lem] Lemur toolkit for language modeling and information retrieval. http://www.lemurproject.org/.

[Lin] LingPipe Java library for the linguistic analysis of human language. http://www.alias-i.com/lingpipe/.

[LW06] H. Liu and J. Wang. A new way to enumerate cycles in graph. *Proceedings of AICT/ICIW*, 2006.

[MA01] M. Montague and J.A. Aslam. Relevance score normalization for metasearch. *Proceedings of the Tenth International Conference on Information and Knowledge Management*, 2001.

[MLS+07] T. Mitamura, F. Lin, H. Shima, M. Wang, J. Ko, J. Betteridge, M. Bilotti, A. Schlaikjer, and E. Nyberg. JAVELIN III: Cross-lingual question answering from Japanese and Chinese documents. *Proceedings of NTCIR-6*, 2007.

[MM01] R. Mihalcea and D.I. Moldovan. eXtended WordNet: progress report. *Proceedings of the NAACL 2001 Workshop on WordNet and Other Lexical Resources*, 2001.

[MN02] D. Moldovan and A. Novischi. Lexical chains for question answering. *Proceedings of COLING 2002, pp.674-680*, 2002.

[MSCT05] L. Màrquez, M. Surdeanu, P. Comas, and J. Turmo. A robust combination strategy for semantic role labeling. *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, 2005.

[NFM+05] E. Nyberg, R. Frederking, T. Mitamura, M. Bilotti, K. Hannan, L. Hiyakumoto, J. Ko, F. Lin, L. Lita, V. Pedro, and A. Schlaikjer. JAVELIN I and II systems at TREC 2005. *Proceedings of the Fourteenth Text REtrieval Conference*, 2005.

[NMF+05] E. Nyberg, T. Mitamura, R. Frederking, V. Pedro, M.W. Bilotti, A. Schlaikjer, and K. Hannan. Extending the JAVELIN QA system with domain semantics. *Proceedings of the Question Answering in Restricted Domains Workshop at AAAI*, 2005.

[OC03] P. Ogilvie and J. Callan. Combining document representations for known-item search. *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2003.

[ONL] OpenNLP Java API for various natural language processing tasks. http://opennlp.sourceforge.net/.

[Par] HTML Parser for the real-time parsing of real-world HTML in Java. http://htmlparser.sourceforge.net/.

[Por80] M.F. Porter. An algorithm for suffix stripping. *Program, 14(3):130-137*, 1980.

[PRY05] V. Punyakanok, D. Roth, and W. Yih. The necessity of syntactic parsing for semantic role labeling. *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence*, 2005.

[PWH+04] S. Pradhan, W. Ward, K. Hacioglu, J. Martin, and D. Jurafsky. Shallow semantic parsing using support vector machines. *Proceedings of HLT/NAACL*, 2004.

[REP+] J. Ruppenhofer, M. Ellsworth, M.R.L. Petruck, C.R. Johnson, and J. Scheffczyk. FrameNet II: Extended theory and practice. http://framenet.icsi.berkeley.edu/.

[Sch05] N. Schlaefer. Pattern learning and knowledge annotation for question answering. *Student Research Project*, 2005.

[SGS06] N. Schlaefer, P. Gieselmann, and G. Sautter. The Ephyra QA system at TREC 2006. *Proceedings of the Fifteenth Text REtrieval Conference*, 2006.

[SGSW06] N. Schlaefer, P. Gieselmann, T. Schaaf, and A. Waibel. A pattern learning approach to question answering within the Ephyra framework. *Proceedings of the Ninth International Conference on TEXT, SPEECH and DIALOGUE*, 2006.

[SHTT06] S. Stenchikova, D. Hakkani-Tür, and G. Tur. QASR: Question answering using semantic roles for speech interface. *Proceedings of ICSLP-Interspeech*, 2006.

[SJT+05] R. Sun, J. Jiang, Y.F. Tan, H. Cui, T.-S. Chua, and M.-Y. Kan. Using syntactic and semantic relation analysis in question answering. *Proceedings of the Fourteenth Text REtrieval Conference*, 2005.

[SS99] R.E. Schapire and Y. Singer. Improved boosting algorithms using confidence-rated predictions. *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*, 1999.

[Tie70] J.C. Tiernan. An efficient search algorithm to find the elementary circuits of a graph. *Communications of the ACM, 13(12):722-726*, 1970.

[Voo02] E.M. Voorhees. Overview of the TREC 2002 question answering track. *Proceedings of the Eleventh Text REtrieval Conference*, 2002.