Carnegie Mellon

Universität Karlsruhe (TH)

Pattern Learning and Knowledge Annotation for Question Answering

Interactive Systems Labs.

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

Student Project / Studienarbeit
Nico Schlaefer
nico@schlaefer.de

Supervisors:
M.A. Petra Gieselmann
Dr.-Ing. Thomas Schaaf
Prof. Dr. Alex Waibel

September 2005 – November 2005

Abstract

Question answering – a type of information retrieval that deals with natural language questions – is a task that has been adressed by numerous systems following a variety of linguistic and statistical approaches.

In this project, I developed the Ephyra question answering engine, which aims to be a modular and extensible framework that allows to integrate multiple approaches to question answering in one system.

The current Ephyra system follows a pattern-learning approach. It automatically learns text patterns that can be applied to text passages for answer extraction. The system can be trained on question-answer pairs, using conventional web search engines to fetch passages suitable for pattern extraction. I propose a new approach to question interpretation that abstracts from the original formulation of the question, which helps to improve both precision and recall of answer extraction.

In addition, Ephyra deploys knowledge annotation to support frequent question classes and questions that are hard to address with generic methods (e.g. definition questions, queries for the weather). Answers to such questions are extracted directly from structured web sites or web services.

Table of Contents

1 Introduction

This chapter gives an introduction to question answering in general and the scope of this project in particular.

1.1 What is Question Answering?

Question answering (QA) is a form of information retrieval. A user can query a question answering system, which then searches a knowledge source and returns one or more results.

In contrast to conventional information retrieval (IR), a QA system can handle questions in natural language and it responds by returning a short text passage or even an accurate answer rather than a whole document.

While early QA systems were often restricted to a specific domain, current research focuses on open-domain QA. Often, the Web or a closed corpus (e.g. a collection of newspaper articles) is deployed as an unstructured knowledge source, but some systems also incorporate structured sources such as relational databases. Web-based systems usually use statistical methods to exploit data redundancy, while systems that work on a closed corpus require linguistically sophisticated techniques to match questions and answers.

Some systems return text passages, others try to provide exact answers. While there are many systems that deal with factoid questions (e.g. “Who was the first president of the United States?”), unrestricted questions such as definition questions (“What is question answering?”) or list questions (“What are the 10 largest car manufacturers?”) are hard to address.

QA systems also differ in their user interface, which can be based on speech or text. Usually, the interaction is restricted to the user asking a question and the system returning an answer, but research is done on interactive QA.

The tutorial [Prager03] gives a general overview of QA and introduces several existing systems. [Lin03] presents the two major techniques for web-based QA: knowledge mining (the search on an unstructured knowledge source) and knowledge annotation (answer extraction from semi-structured sources).

1.2 Motivation

Conventional IR systems, e.g. web search engines such as Google, force the user to transform the question into keywords, allowing only a limited set of operators to express relationships between the keywords. QA systems introduce a more intuitive and flexible solution to the information retrieval task.

A user can query a QA system in a natural manner, similar to asking a wise human for help. In this way, the user does not have to adapt to the world of computers and also knowledge about the operation of IR systems is unnecessary.

Furthermore, natural language questions are far more expressive than search engine queries. Interrogatives can be used to specify the type of answer that is desired. For example, a "when"-question indicates that the answer is expected to be a date, while an answer to a "who"-question is likely to be a proper name. This allows to answer a question directly rather than to return an entire document that might contain the answer.
Another example for the expressiveness of natural language is the use of tenses. E.g. when submitting the query "winner presidential elections" to a web search engine right after the elections, the documents that are returned are likely to contain prognoses and not the actual results of the elections. In contrast, the question „Who won the presidential elections?" asks explicitly for the outcome of the elections.

These properties allow a wide range of applications of question answering. A QA system can be deployed by any kind of system that imitates human behavior to pretend general knowledge. Examples are intelligent dialog systems or humanoid robots such as the SFB 588[1], which is a household robot designed to help people in the kitchen. User studies have shown that users really expect a humanoid robot to know everything.
An intelligent system can also use QA autonomously, e.g. when confronted by the user with an unknown word. In this situation, QA can help to gather background information that allows the system to integrate the term into its internal ontology without bothering the user with unnecessary inquiries.

QA can also be utilized by systems that interact with humans, supporting them in their everyday tasks. Such systems may be required to provide a user with details on persons, locations or technical terms. An integration of QA in the CHIL[2] project,

[1] Developed at the Universität Karlsruhe, http://www.sfb588.uni-karlsruhe.de/

[2] Coordinated by the Interactive Systems Labs, http://isl.ira.uka.de/index.php?id=22

which aims at supporting people in their interaction as an electronic butler, seems to be beneficial.

Another interesting application is the field of language translation. A QA system can be used to assign confidence measures to translations. This can be done by rephrasing a translation into a decision question, which is then posed to the QA system. A translation can be considered to be correct, if its content can be verified by the QA system. However, if the content of the original text is incorrect – or only correct in a specific context – a translation that preserves this content will be considered to be wrong. In such cases, a translation would need to be reviewed manually.

1.3 Scope of the Project

In this project, I developed the Ephyra[3] QA system, an *open-domain* question answering system which is *based on the web* as a knowledge source.

Ephyra deploys the web as a huge *unstructured* database, which is accessible through commercial search engines such as Google or Yahoo. In addition, it provides an interface that allows to easily incorporate *(semi-)structured* sources. Examples for such sources are web sites, e.g. the online encyclopedia Wikipedia[4], or web services, e.g. the GlobalWeather[5] web service that provides meteorological data from gaging stations all over the world.

Ephyra does not try to understand the semantics of a question or answer but it uses *statistical methods* that rely on data redundancy. In addition, some *linguistic transformations* of the question are performed.

The Ephyra engine can handle *factoid questions* and *definition questions*. For most types of questions, it tries to return *accurate answers*. If this is not possible, it returns *short text passages*, which are single sentences or sentence fragments that are assumed to contain the answer.

Two modes of user interaction are supported. The primary mode is *text-based*. The user can key in a question either through a command line tool or through a web interface. The Ephyra engine then returns a ranked list of answers. Alternatively, the user can query the system in *spoken natural language* and the system replies by reading out an answer to the user.
All these interfaces are *non-interactive*, i.e. the user asks a single question and the system replies directly to the question without further inquiries.

[3] Ephyra was an oracle in ancient Greece, located near the village of Mesopótamos. Similar to an oracle, a QA system has the purpose of finding answers to questions.

[4] http://www.wikipedia.org/

[5] http://www.capescience.com/webservices/globalweather/

1.4 Outline

In Chapter 2, I introduce basic concepts and tools that have not been developed within this project but that are fundamental components of the Ephyra system discussed in the later chapters.

Chapter 3 gives an overview of related work in the field of question answering. I discuss several existing systems of similar scope (open-domain, web-based) that have inspired my own work.

Chapter 4 is an outline of the Ephyra QA system. The focus is on the actual QA engine, but the interfaces for user interaction are also discussed briefly. This chapter complements the system documentation that can be referred to for implementation details.

The Ephyra system uses text patterns to interpret questions and to extract answers from documents. Chapter 5 describes how these patterns are obtained.

In Chapter 6, I describe the experimental setup I used to evaluate the system and discuss the results.

Finally, Chapter 7 is a summarization of my work on QA and the Ephyra system. It concludes with an outlook on improvements of the system and future work on question answering.

2 Fundamentals

2.1 QA Architecture

QA systems typically consist of three core components: query formation, information retrieval and answer selection.

The *query formation* component takes a question string as input and transforms it into one or more queries, which are passed to the *information retrieval* component. Depending on the type of the question, the IR component queries one or more knowledge sources that are suitable for that type and aggregates the results. These results are then passed as answer candidates to the *answer selection* component, which drops candidates that are unlikely to answer the question and returns a ranked list of answers.

This overall architecture is illustrated in Figure 1.

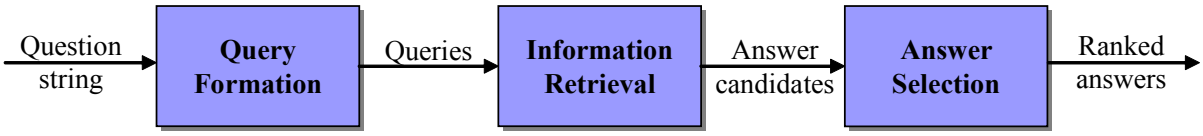

Figure 1 – Typical architecture of QA systems.

2.2 Part of Speech Tagging

A part of speech (POS) tagger parses a text and annotates each of the words with its part of speech, i.e. it identifies nouns, verbs, prepositions etc.
For example, “This is a sentence” would correctly be tagged “This/DT is/VBZ a/DT sentence/NN”, where DT stands for “determiner”, VBZ for “verb 3rd person singular present tense” and NN for “noun”.

POS tagging can be hard because a word can belong to multiple syntactic categories. E.g. the word “books” can either be the plural form of the noun “book” or the 2nd person singular present tense of the verb “to book”. Thus, it is not sufficient to look up a word in a dictionary but the context of the word needs to be considered.

The Ephyra system deploys part of speech tagging to determine the main verb in a question (see Section 4.1.1 on question normalization). I use the POS tagger from the openNLP project[6], an open source project that aims at developing a collection of tools for NLP. The performance using the default model is relatively poor. A model that is tailored for question strings could improve the results.

2.3 One4All

The One4All system is a toolkit which allows to create a speech application without particular knowledge on speech recognition.

The system consists of three modules: a communication server, a speech recognizer and a receiver. The communication server is the central component. The other modules connect to this server as agents and communicate by sending and receiving messages. For speech recognition, One4All uses the Ibis decoder, which is part of the Janus Recognition Toolkit (JRTk). The receiver is a GUI that can be used to record speech and to configure the One4All system.

The speech interface of Ephyra described in Section 4.4.3 is based on One4All. Ephyra connects to the communication server as an agent. It receives messages containing question strings from the speech recognizer and sends back messages that encapsulate the answers, which are processed by a speech synthesizer.

One4All and the JRTk were jointly developed at the Interactive Systems Labs[7] at Carnegie Mellon University and the University of Karlsruhe.

[6] http://opennlp.sourceforge.net/

[7] http://isl.ira.uka.de/

3 Related Work

Research on question answering already began in the 1960s. The early QA systems were restricted-domain systems that deployed a structured knowledge source such as a database containing manually collected knowledge on the respective domain. For example, the BASEBALL system could answer questions on the US baseball league and was restricted to one season.

In 1999, the Text REtrieval Conference (TREC) started a QA track. This annual evaluation of QA systems significantly advanced research and collaboration in the field of question answering. A substantial portion of the recent work on QA is published in the proceedings of TREC.

Current question answering systems are usually open-domain systems that use a collection of documents as an unstructured knowledge source. Often, closed text corpora are used for evaluations, since they do not change over time, but more and more systems deploy the web as the largest digital knowledge source. Examples of publicly available web-based QA systems are Ask Jeeves[8] or START[9].

In the following sections, I describe systems that particularly had an impact on my work. The AnswerBus[10] system follows the same core architecture as Ephyra and deploys word form modification and answer filtering techniques. AskMSR applies query reformulation and uses answer types to judge the answers. LAMP follows a pattern mining and matching approach that I extended in Ephyra.

Another system that was relevant for my work is Aranea [Lin02], which introduced the idea of combining knowledge mining and knowledge annotation techniques in a single framework.

---

[8] http://www.ask.com/

[9] developed at MIT, http://start.csail.mit.edu/

[10] http://www.answerbus.com/

3.1 AnswerBus

The architecture of AnswerBus [Zheng02] is similar to the QA architecture described in Section 2.1. AnswerBus accepts questions in multiple languages and uses the online-translator BabelFish[11] from AltaVista to translate the questions to English. Depending on the type of question, it deploys different web search engines.

The query formation component drops function words and frequent words from the question string and creates a set of keywords, similar to Ephyra's "bag of words" generator described in Section 4.1.2. In addition, AnswerBus converts verbs to the form that is likely to occur in the answer. E.g. the verb "end" in the question "When did the Jurassic Period end?" is transformed into "ended". Similarly, Ephyra uses POS tagging to identify the main verb of a question and handles verb constructions with auxiliary verbs (see Section 4.1.1).

AnswerBus downloads the top documents that are returned by the search engines and parses them into sentences. It then judges these candidate sentences by evaluating several features such as the number of keywords that occur in a sentence and the hit position of the sentence in the search results. Similar techniques are implemented in Ephyra, that uses a dynamic set of filters to manipulate, score and drop answer candidates (Section 4.3).

3.2 AskMSR

AskMSR, developed by Microsoft Research, is a relatively simple web-based QA system that makes use of the redundancy in the web rather than applying linguistically sophisticated techniques.
See [Banko02] for an overview of the AskMSR system and [Brill01], [Brill02] for a more detailed description and performance analysis.

The system processes and answers questions in four steps. At first, it generates rewrites of the question that may occur in a declarative answer. E.g. the question "Where is the Louvre Museum located?" is rephrased into "the Louvre Museum is located in". The Ephyra system also generates query reformulations by applying reformulation rules that can be specified in resource files (Section 4.1.3).
Secondly, AskMSR queries a web search engine for these rewrites, fetches the snippets from the search results and breaks them down into 1-, 2- and 3-grams. These n-grams are then judged by comparing their type to the expected answer type of the question. E.g. if the question asks for a number ("How many …?"), then the score of an n-gram that contains digits is boosted. Ephyra's answer type filter (see Section 4.3.5) follows a similar yet more general approach. In addition, it assigns probabilities to answer type matches and it looks up proper names in named entity lists.

[11] http://babel.altavista.com/

Finally, overlapping n-grams are tiled together to form longer phrases from the existing answers.

3.3 LAMP

The LAMP system [Zhang02] uses text patterns to classify a question and to extract key phrases from the question that can be used to query a web search engine. It knows a relatively small set of 22 question classes (compared to 67 in Ephyra). For each of the classes, several templates describe what a question of that class looks like. E.g. the templates for the class *ACRONYM* describe possible formulations of a question that asks for the abbreviation of a term.
Ephyra uses a more general approach to question interpretation (see Section 4.1.4) that distinguishs "just as many question classes as necessary"[12] to extract the right answers and that supports multiple key phrases.

Furthermore, for each question class, a second set of textual patterns is used to extract answers from text snippets returned by a search engine. These patterns can be learned and assessed automatically, using questions and answer patterns from previous TREC evaluations as training data.
Ephyra also applies text patterns for answer extraction (Section 4.3.6). The pattern learning algorithm described in Chapter 5 is in many ways similar to the approach followed by LAMP, but I extended it to generate a larger number of more generic patterns.

[12] clarified in Section 5.3

4 Architecture of the Ephyra QA System

The Ephyra QA system follows the core architecture described in Section 2.1. It consists of components for query formation, information retrieval and answer selection.

In addition, it comprises multiple user interfaces that support two distinct modes of interaction. A simple command line tool and a web interface can be used for *text-based* interaction. Furthermore, speech recognition and synthesis components allow for *spoken* interaction.

The architecture of the Ephyra system aims to be highly *modular*, which makes it easy to extend it and integrate additional techniques. When discussing the individual components, I also point out how they can be extended. Another major goal was to make the system *adaptable* to other languages. This was achieved by separating language-specific and language-independent code and by defining patterns that are specific for the English language in resource files rather than in the source code.

Figure 2 gives an overview of the overall structure of the system. In this chapter, each component is discussed in detail. The core components (query formation, information retrieval and answer selection) as well as the text-based user interfaces were developed in Java, and the package structure reflects the structure of this chapter. All classes, methods and fields have been documented using Javadoc comments. Please see the Javadoc documentation for an in-depth description of these components, including the format of resource files and implementation decisions.

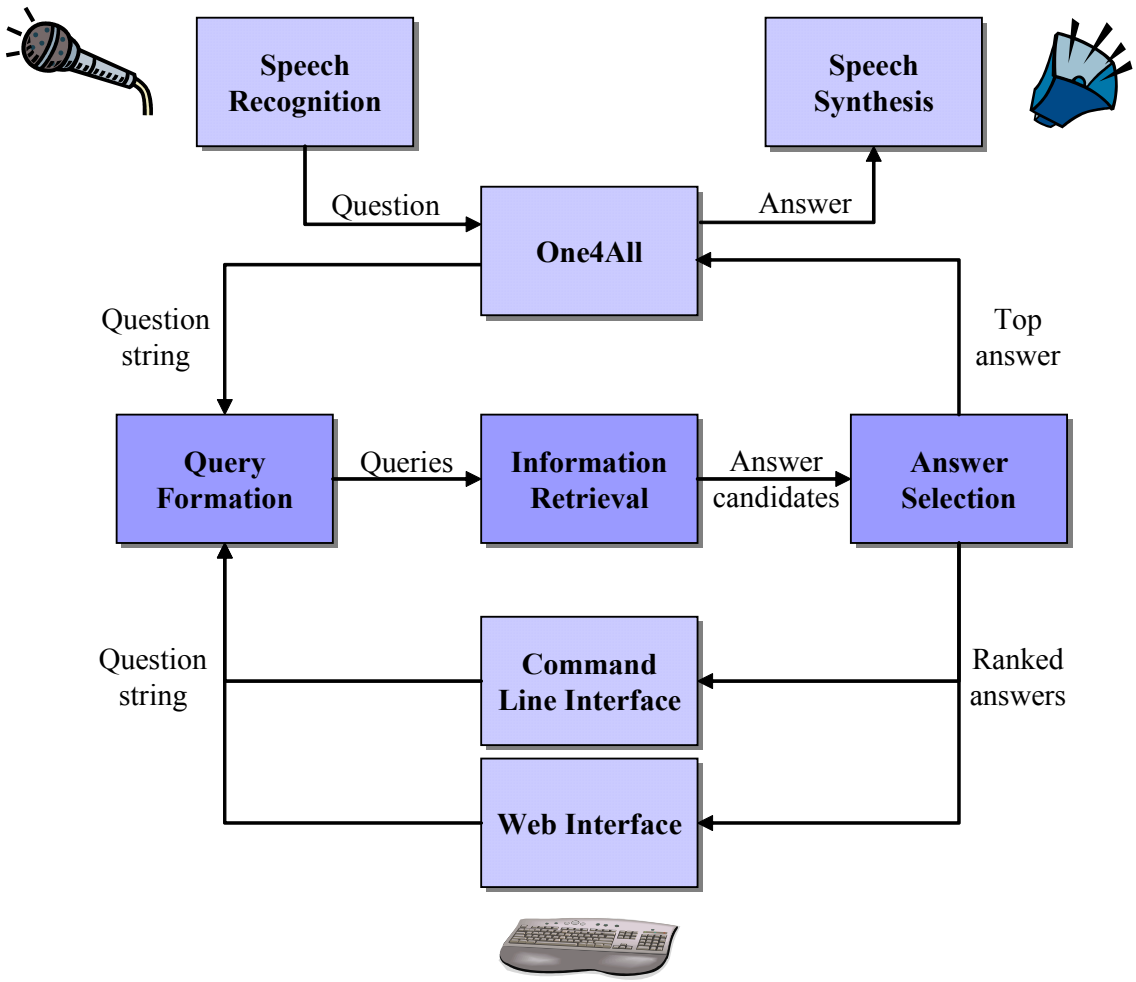

Figure 2 – The overall architecture of the Ephyra QA system.

4.1 Query Formation

The query formation component transforms a question string into a set of queries by applying linguistic techniques. This component is crucial for the performance of the system. By "asking the right questions", the relevance of the results returned by a search engine can be increased significantly.

This is achieved by creating reformulations of the question that are likely to occur in a text passage that contains the answer. Furthermore, a new approach to question interpretation is used to abstract from the original formulation of the question, which allows to return precise answers, even in case of low redundancy. Additionally, a simple set of keywords, a "bag of words", is created as a backup to ensure that the search returns some results.

This component is designed to be modular. Query generators are responsible for transforming question strings into queries. The query formation can be extended by simply creating a new query generator that instantiates a given super class.

Figure 3 shows the components responsible for the query formation. The question normalizer and the query generators are discussed in detail in the following subsections.

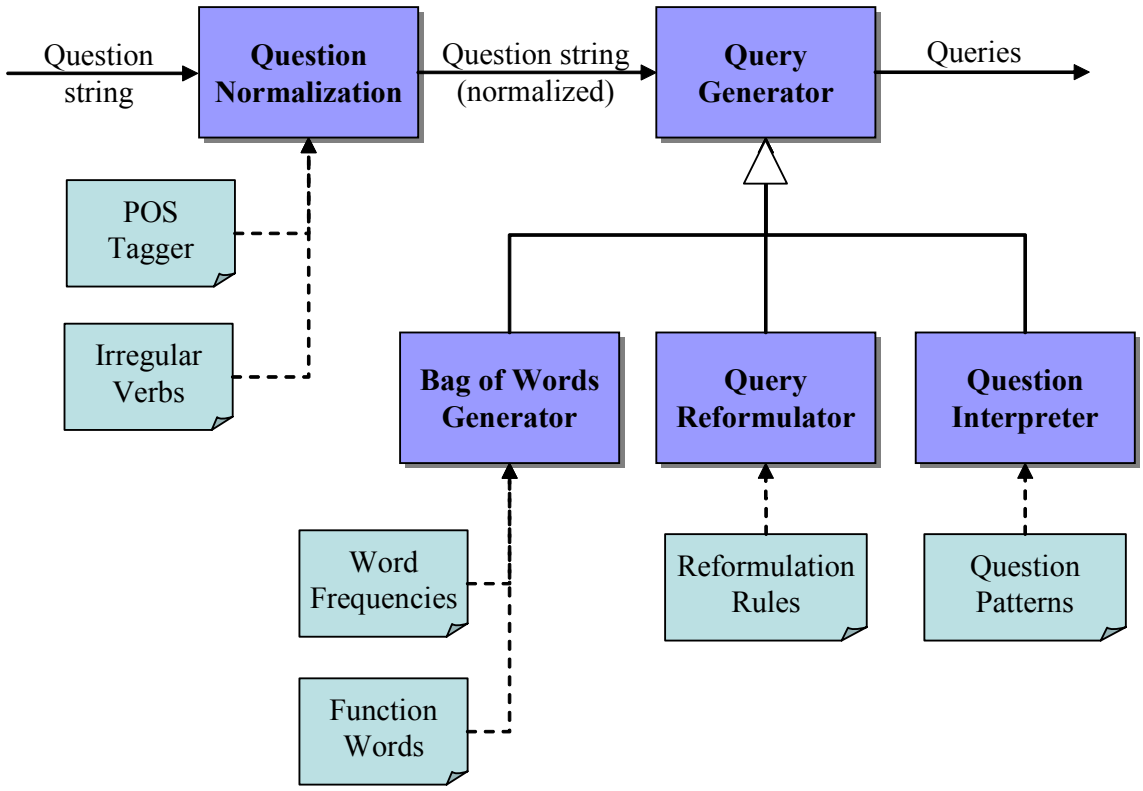

Figure 3 – Overview of the query formation component.

4.1.1 Question Normalization

This component applies a number of normalization steps to the question string. It creates a normalized form of the question that meets certain premises of the query generators.

The question normalizer drops duplicate whitespaces and removes punctuation and quotation marks. The first letter of the question is converted to lower case. Shortforms of the verbs “is” and “are” are replaced by their full forms, e.g. “who’s” is transformed into “who is”.

Furthermore, questions often contain verb constructions with auxiliary verbs which have been replaced in the answer by a simple verb form. Such questions require a special treatment.

Let us for example consider the question “When did Shakespeare write Hamlet?”. A text passage that answers this question is likely to contain the string “Shakespeare wrote Hamlet in”. The auxiliary “did” does not occur in the answer and therefore should be removed from the query string. Furthermore, the main verb “write” should be transformed to simple past.

This is done by first checking whether the question contains an auxiliary verb. Table 1 lists the auxiliaries that are treated by this component as well as the rules that are applied.

Auxiliaries in questions	Transformations
is/are/was/were […] gerund / past participle	is/are/was/were gerund / past participle
can/could/will/would/shall/should/ may/might/must [...] infinitive	can/could/will/would/shall/should/ may/might/must infinitive
have/has/had [...] past participle	has/have/had past participle / simple past
do [...] infinitive	infinitive
does [...] infinitive	3rd person singular present tense
did [...] infinitive	simple past

Table 1 – Auxiliary verbs in questions and how they are transformed.

Note that some of the above verbs are not necessarily auxiliaries (e.g. "is" in "What is a kiwi?"). This component only transforms these verbs if they occur as auxiliaries. Otherwise they are handled by the query reformulation component, which is discussed in Section 4.1.3.

A part of speech tagger (see Section 2.2) is used to identify the main verb in the question. In some cases, the auxiliary needs to be moved before the main verb, in other cases the auxiliary is dropped and the main verb is transformed.

To transform the main verb to simple past, a couple of rules has to be applied (e.g. if the verb ends in 'y', the 'y' is replaced by 'i' before the ending 'ed' is appended). Furthermore, a comprehensive list of irregular verbs, comprising their infinitive, simple past and past participle, is used.
When forming the 3rd person singular present tense of a verb, there are also a few exceptions that need to be considered (e.g. if the verb ends in 'o', 'es' is appended instead of just 's').

These normalization rules are specific for the English language. They are implemented in a separate Java class that needs to be replaced when adapting the system to another language.

4.1.2 Bag of Words

For all questions, independent of their type, a simple „bag of words" query is created, which is a set of keywords. This is done by extracting the words in the question and dropping function words[13] and frequent words.

[13] Wikipedia: "Function words are words that have little lexical meaning or have ambiguous meaning, but instead serve to express grammatical relationships with other words within a sentence, or specify the attitude or mood of the speaker."

All words are looked up in a list that contains the function words except numbers. If a word occurs in the list, it is dropped. Function words express little content but are part of the grammar of a language. Thus, they are not relevant as keywords. An exception are numbers, which should obviously be part of a query.

In the second step, the number of remaining words is checked against a threshold. If it exceeds the threshold by *n*, the *n* most frequent words are dropped. This is done by looking up the words in a dictionary containing words together with their frequencies of occurrence.

The index should ideally be created from a corpus that is representative for the data that is searched. I used a subset of the Gigaword[14] corpus, a collection of English newspaper articles, assuming that the distribution of the words is similar for the WWW. The size of the index can be reduced significantly by removing rare words (which are not relevant since they should never be dropped).

4.1.3 Query Reformulation

For questions that match certain patterns, additional queries are generated, which are more specific than a "bag of words" and are therefore more likely to return relevant results.

These alternative queries are defined in resource files that comprise a question pattern and a number of reformulations for questions that match that pattern, paired with scores that are assigned to the reformulations. A question pattern is a regular expression (in Java syntax, see [RegTut] for details). A reformulation is a string that may contain references to substrings of the original question. The score of a reformulation is used by the answer selection component to rank answer candidates that result from the reformulation (see Section 4.3.3). It should be the higher the more specific the reformulation is.

An example of such a reformulation file is shown in Figure 4. Numbers enclosed in brackets refer to the corresponding capturing groups in the regular expression (denoted by parentheses). The operator '<' is used to place the group on the right side between each two words in the group on the left.

```
QuestionPattern:
(?i)when (is|are|was|were) (.*)

QuestionReformulations:
[2]
3.0
[2] [1]
4.0
[2] [1] in
5.0
```

[14] from the Linguistic Data Consortium (LDC), http://www.ldc.upenn.edu/

```
[2] [1] on
5.0
[2] [1] at
5.0
[2]<[1]
4.0
[2]<[1] in
5.0
[2]<[1] on
5.0
[2]<[1] at
5.0
```

Figure 4 – A query reformulation resource file, comprising a regular expression and a number of reformulations paired with scores.

Let us consider the question "When was Einstein born?" as an example to clarify how the reformulation works. The question matches the pattern in the sample resource file. The expression `[1]` refers to the string "was" and `[2]` refers to "Einstein born". The operation `[2]<[1]` results in the string "Einstein was born". The following reformulations are created:

- "Einstein born"
- "Einstein born was"
- "Einstein born was in"
- "Einstein born was on"
- "Einstein born was at"
- "Einstein was born"
- "Einstein was born in"
- "Einstein was born on"
- "Einstein was born at"

All these reformulations are passed to the IR component and result in a search engine query. "Einstein was born in" is assigned a higher score than e.g. "Einstein born" because it is more specific.

Note that this component may generate senseless reformulations (e.g. "Einstein born was" in the example). To avoid this, linguistically sophisticated techniques would be required. On the other hand, the search for such reformulations usually fails and thus the search results are not corrupted.

4.1.4 Question Interpretation

This component, together with the answer extraction filter discussed in Section 4.3.6, implements a new approach to question answering based on text patterns.
The Ephyra system uses two types of text patterns. Question patterns are applied to question strings to interpret the questions and extract relevant information. In this section, I describe how text patterns can be used for question interpretation.
On the other hand, the answer extraction filter uses a second type of patterns called answer patterns to extract answers from relevant text snippets.

In Chapter 5, I decribe in detail how the question and answer patterns can be obtained and I explain the format of the patterns.

The basic assumption behind this approach is that each question can be reduced to 3 components: A question asks for a *property* of a *target* in a specific *context*.
Let us e.g. analyze the question "How many calories are there in a Big Mac?" (TREC8, question 56). This question can be interpreted as follows:

- Property: *NUMBER*
- Target: "calories"
- Context: "Big Mac"

The question asks for a property of the target object "calories", which is its number. The context object "Big Mac" narrows down the scope of the question to a particular food.
Together, the 3 components (the property, the target object and an arbitrary number of context objects) form the *interpretation* of a question.

Ephyra knows about 70 properties that a question may ask for. Examples of such properties are the *DATE* of an event, the *NAME* of a person or the *LONGFORM* of an abbreviation.
For each of these properties, a number of so-called question patterns is specified in a separate resource file. These patterns are basically regular expression. In addition, a pattern contains exactly one target tag (<*T*>), indicating the target of the question, and *0* to *n* context tags (<*C*>), indicating context information. These tags are replaced by capturing groups before a pattern is applied to a question. Furthermore, the tags can be combined with *object types* to restrain the format of the objects they stand for and *shortcuts* can be used to conveniently define more generic patterns. These two concepts are described in detail in Section 5.1.

Ephyra can automatically interpret a question by sequentially applying all the question patterns. If the question matches a pattern, Ephyra knows the property that the question asks for (this is the property that the matching pattern belongs to). As a side effect, capturing groups in the questing pattern extract the target and context objects (if any) from the question string.
In our example, the question could be interpreted with a pattern "how many <*T*> are (there)?(in|on|at) <*C*>" belonging to the property *NUMBER*.

Sometimes, a question matches more than one question pattern. In this case, the interpretations are sorted by their length, which is the total number of characters in the target and all context objects. Only interpretations of minimal length are further considered. This approach is reasonable since shorter interpretations are usually more specific. E.g the question "Who is the president of France?" could be interpreted in the following ways:

- Property: *NAME*
  Target: "president of France"
- Property: *LEADER*
  Target: "France"

Both interpretations are correct, but the second (and shorter) one is certainly more specific.

The interpretations of minimal length are then transformed into search engine queries. The query string is the concatenation of the target object and all context objects in quotation marks plus all the keywords in the question string (see Section 4.1.2 for details on how the keywords are determined).
In the above example, Ephyra would build the following query string:

```
"calories" "Big Mac" calories Big Mac
```

The score of the queries should be larger than the score of the "bag of words" since they are more specific, but smaller than the score of queries generated by the query reformulator.

4.2 Information Retrieval

The information retrieval component comprises two types of searcher objects for unstructured and (semi-)structured knowledge sources. Knowledge miners are used to query unstructured sources while knowledge annotators allow to integrate web sites and other resources that provide structured information.

Figure 5 shows how queries are processed by the IR component.

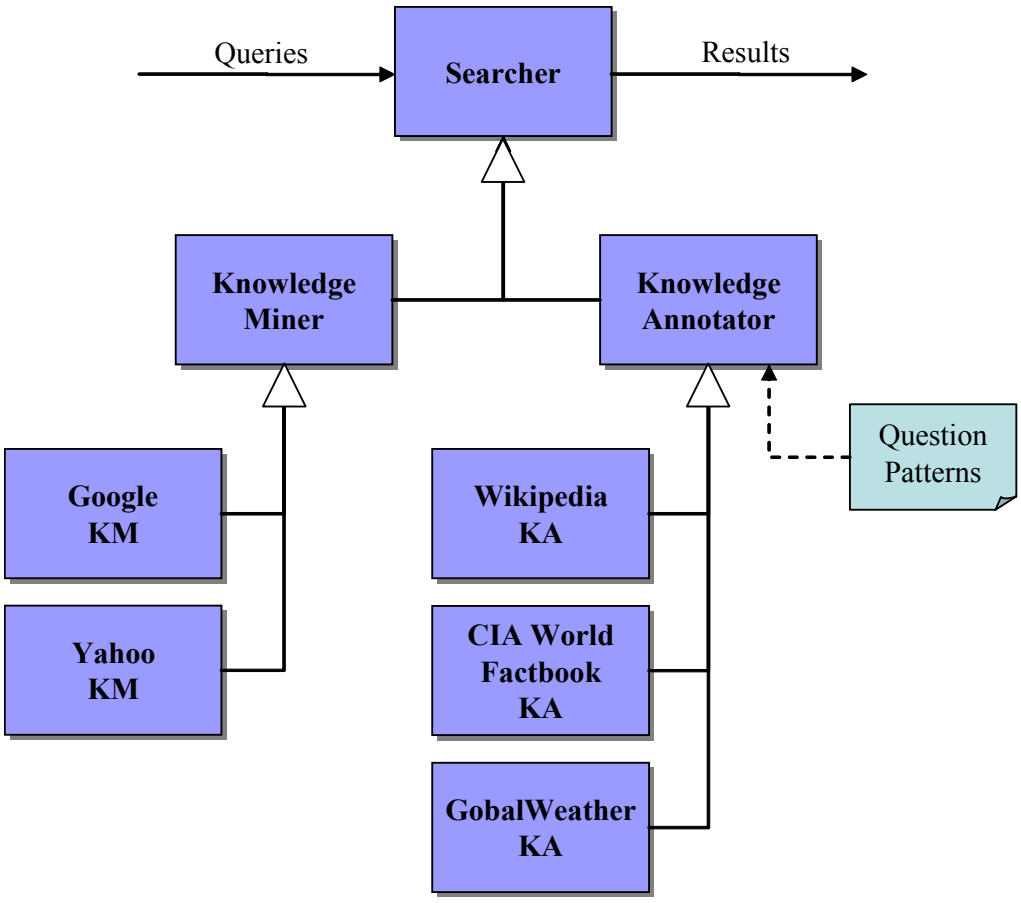

Figure 5 – Knowledge miners and knowledge annotators in the IR component.

Knowledge miners and knowledge annotators are implemented as threads, thus several queries can be performed in parallel. The results are aggregated and passed to the answer selection component.

4.2.1 Knowledge Mining

A knowledge miner (KM) deploys a conventional IR system to search an unstructured knowledge source such as the World Wide Web or a closed corpus.

The results returned by a KM are text snippets or even whole documents. They are never returned as an answer to the user but the answer selection component (Section 4.3) breaks them down into sentences or even extracts precise answers.
The answer selection component scores the results to create a ranked list of answers. To avoid that a "raw" result from a KM is returned as an answer to the user, the initial score of these results is set to *–Infinity*.

I implemented two knowledge miners that use Google and Yahoo to search the WWW and fetch the snippets that are returned by the search engines. Both search engines provide APIs for Java applications. Since the maximum number of search results is limited for these APIs (10 results for Google and 100 results for Yahoo), Ephyra creates several knowledge miners for each query to fetch different ranges of the snippets in parallel.

This approach has the drawback that the snippets are limited in their length and thus sentences are often shortened and the answer phrase might not occur in the snippet. Alternatively, whole documents could be downloaded, but this turned out to be very time intensive and only a small number of documents could be parsed.

4.2.2 Knowledge Annotation

A knowledge annotator (KA) deploys a structured or semi-structured resource such as a web site or a web service that provides answers to a specific class of questions. Answers from such resources can be assumed to be correct and therefore Ephyra returns them as the top ranked answers. This is achieved by setting their score to *+Infinity*.

For demonstration purposes, I implemented a KA that queries the Wikipedia online encyclopedia for answers to definition questions as well as a KA that extracts country information from the CIA World Factbook.
I also created a KA that deploys the GlobalWeather web service to answer questions on the current weather in cities all over the world and the location of cities.

A pattern file containing a set of regular expressions is used to determine whether a KA is appropriate for a specific question. Each pattern is paired with an expression identifying the relevant content of a question, i.e. the content required to extract the

answer from the knowledge source. The format of these expressions is similar to the format used to describe query reformulations (see Section 4.1.3).

The pattern file for the CIA World Factbook is shown in Figure 6. The KA is appropriate for questions such as "What are the natural resources in the United States?". The relevant content that is extracted from that question would be "natural resources#united states".

```
KnowledgeAnnotator:
WorldFactbook

QuestionPatterns:
(?i)(.* )?(what (is|are)|name) the
     (area|capital|climate|languages|natural resources).*
     (of|in) (.*?)(\?|\.)?
[4]#[6]
```

Figure 6 – A pattern file describing questions that can be handled by the KA for the CIA World Factbook.

Incorporating structured sources means a fair amount of manual work. Each web site or web service has a different interface and requires an individual parser that extracts the answers. However, the performance of a QA system can be improved significantly, as shown in [Lin02]. By analyzing question logs or the questions from the TREC QA track for frequent question classes, the effect can be maximized.

4.3 Answer Selection

The answer selection component is, besides the query formation, the second crucial component of the QA system. From the huge amount of information retrieved from the knowledge sources, it extracts the information that is relevant to the user and returns a ranked list of answers.

This is achieved by applying a set of filters to the answer candidates. A filter can create new answer candidates from existing candidates and it can drop candidates that are unlikely to answer the question. Furthermore, each candidate has a score that is manipulated by some filters. These filters check the candidates for a specific feature and promote those candidates that are promising with respect to that feature.

The architecture of the answer selection component is highly modular. A new feature can be integrated by simply creating a filter and adding it to the queue at the desired position. The filters should be arranged in a manner that minimizes computation time, i.e. if possible, filters that drop results should be applied first and computationally expensive filters at last.

Figure 7 gives an overview of the answer selection component. The individual filters are described in detail in the following sections. The order in which the filters are applied is indicated in the figure.

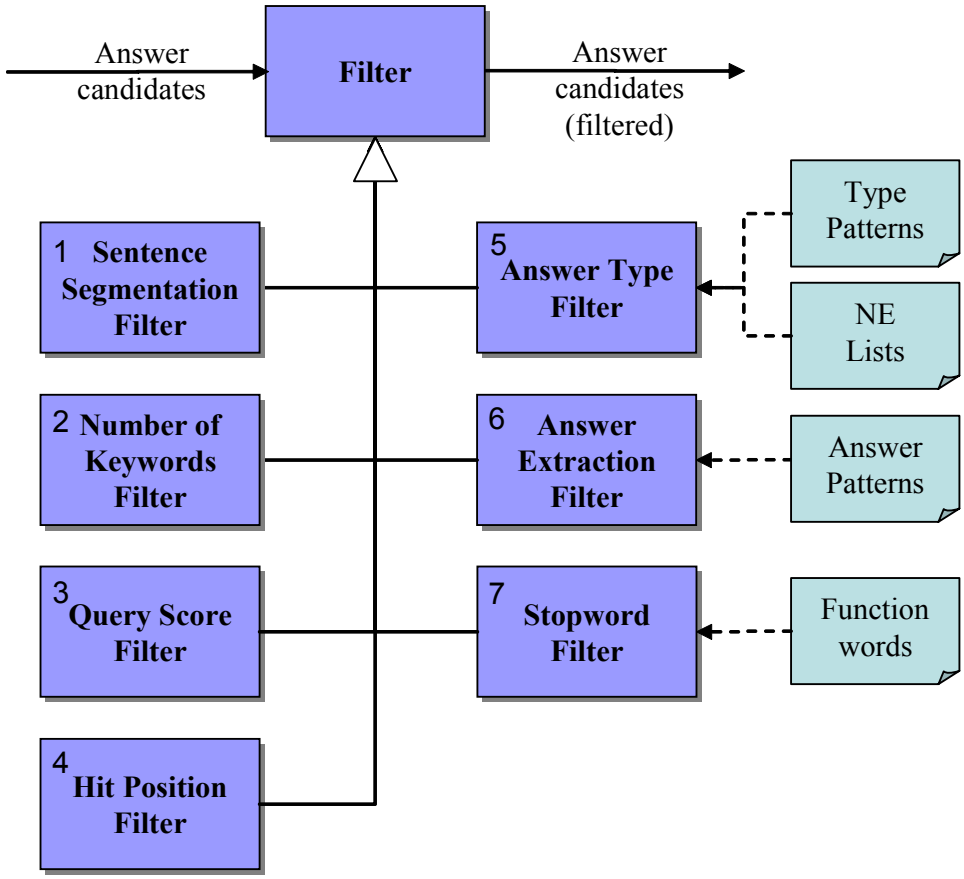

Figure 7 – The answer selection component applies a set of filters to the answer candidates.

4.3.1 Sentence Segmentation

This filter breaks down the snippets that are returned by the knowledge miners into sentences. For each sentence, it creates a new answer candidate. The initial score of the answer candidates is set to zero.

The filters that are described in the Sections 4.3.2 to 4.3.5 are only applied to answer candidates created by this component, i.e. whole sentences.

4.3.2 Number of Keywords

A simple feature that can be used to judge the quality of an answer candidate is the number of keywords that it contains. A candidate is dropped, if it does not satisfy the condition

$$M \geq \left\lfloor \sqrt{K-1} \right\rfloor + 1$$

where K is the number of keywords in the question and M is the number of keywords that also occur in the answer string. If the condition is satisfied, a candidate is promoted by a score which is proportional to M.

This formula is borrowed from [Zheng02] and is reasonable, which can be understood by inserting numbers. If the question for example contains three keywords, then the formula suggests that at least two out of these keywords should also occur in the answer.

4.3.3 Query Score

Another relevant feature is the score that is assigned to a query by the query generators discussed in Section 4.1. The higher the score, the more specific is the query, which implies that answer candidates that are retrieved with this query are more relevant. Thus, candidates are promoted by a score proportional to the query score.

4.3.4 Hit Position

Search engines commonly return the search results in the order of their confidence, starting with the most promising result. This is taken into consideration by the hit position filter, which decrements the score of a candidate by a value proportional to the hit position.

4.3.5 Answer Type

A more sophisticated filter is the answer type filter. It estimates the probability of an answer candidate being of the same type as the question and increments the score of the candidate by a value that is proportional to the probability. E.g. a “where”-question asks for a place, so an answer candidate that contains a location is promoted.

Examples of answer types are *PLACE*, *NUMBER* or *DATE*. For each of the types, a resource file lists a number of expressions that describe questions of that type, along with an estimated probability that an expression classifies a question correctly. A second list of expressions describes answers of the given type. Again, the expressions are paired with probabilities. Figure 8 shows the resource file for the answer type *PLACE*.

```
AnswerType:
PLACE

QuestionPatterns:
(?i)(.*, )?where.*
1.0
(?i).*(what|which|name).*(capital|city|continent|country|
      mountain|place|river|state).*
1.0

AnswerPatterns:
```

```
[NE:res/namedentities/PLACE]
0.8
.*(in|on|at) [A-Z].*
0.8
.* [A-Z].*
0.3
```

Figure 8 – Resource file for the answer type PLACE.

An expression can either be a regular expression in Java syntax or a list of words. In the latter case, the expression has the format *[filename]* or *[NE:filename]*, where *filename* refers to a file that contains a word list. A question or answer matches such an expression, if any of the words appear in the list. *NE* indicates, that only the named entities are looked up in the list, i.e. all the capitalized words. The word lists are loaded into dictionaries that internally use hash tables and therefore allow lookups in constant time.
Ephyra uses named entity lists for person, place and organization names. In the above example, *res/namedentities/PLACE* is a file that contains a list of location names.

For each of the answer types, Ephyra computes the probability of the question being of that type. This is the maximum of the probabilities of all matching expressions or *0*, if the question does not match any of the expressions. Ephyra also determines for each answer candidate the probability that the answer is of that type. The probability of the question and answer being of the same type is the product of these probabilities.
For example, the estimated probability that the question "Where is Pittsburgh?" is of type *PLACE* would be *1.0* and the probability that the answer "Pittsburgh is in Pennsylvania" is of that type would be *0.8*. So the probability of the question and the answer being of the same type is $1.0 \times 0.8 = 0.8$.

The expressions that describe the questions can overlap for different answer types, thus a question can be of more than one type. In that case, the maximum of the probabilities for all these types is used.

4.3.6 Answer Extraction

This component processes the answer candidates that result from queries generated by the question interpretation component described in Section 4.1.4. It is applied to the "raw" results from the knowledge miners and not to individual sentences since it uses cross-sentence patterns.

For each such candidate, there is a corresponding question interpretation that comprises the target object and context objects (if any) of the question as well as the property of the target that the question asks for. This information, along with a set of so-called answer patterns, is used to create a ranked list of answers from the answer candidates.

The answer candidates are text snippets that contain the target and context objects of the question. First of all, these objects are replaced by the respective tags (<*T*> and <*C*>).

In the running example ("How many calories are there in a Big Mac?", see Section 4.1.4), the snippet "One Big Mac contains 560 calories and 32 grams of fat." would be transformed into "One <*C*> contains 560 <*T*> and 32 grams of fat."

Ephyra uses answer patterns to extract answers from the modified snippets. For each of the specified properties that a question may ask for, there is a distinct resource file that defines a set of answer patterns. These patterns are basically regular expressions, but they contain a target tag (<*T*>) and a property tag (<*P*>). For convenience, an answer pattern may also contain shortcuts that represent regular expressions, e.g. the shortcut <*be*> stands for the regular expression *(is|are|was|were)*. See Section 5.1 for details on this concept.
The answer patterns were assessed on training snippets and the number of correct answers and incorrect answers that were extracted with each pattern was counted. These values are stored in the resource files along with the patterns and can be used to compute a confidence measure for a pattern.
Section 5 describes how the answer patterns were obtained and assessed and explains the structure of the patterns in detail.

Ephyra picks the answer patterns for the property that a question asks for and replaces the property tags (<*P*>) with the general capturing group *(.*)*. It then applies each pattern to all the modified snippets. Whenever a snippet matches a pattern, the part of the snippet that is represented by the capturing group is extracted. This string is considered as a potential answer to the question.
An example of an answer pattern that would extract the correct answer from the above snippet is "contains <*P*> <*T*>". It would extract the object represented by the <*P*> tag, which is "560".
If the extracted string does not already occur in the set of answer candidates, then a new candidate is created and its initial score is set to the confidence measure of the pattern. Else, the score of the existing candidate is simply incremented by that confidence value.

The result is a ranked list of answer candidates. Ephyra always favors answers that are extracted by this component over answers that are created by the sentence segmentation filter (Section 4.3.1) because they are usually shorter and more reliable. This is achieved by adding an offset value to the scores of the answers from this component. The offset must be larger than the maximum score of a sentence but not *+Infinity*.

4.3.7 Stopwords

This filter is applied to all answer candidates (both precise answers and whole sentences) and drops an answer candidate if the answer string

- is a function word but not a number, e.g. "they", "many", "here". (See Section 4.1.2 for details on function words.)
- contains a single bracket or quotation mark without the counterpart.
- starts with an interrogative or ends in a question mark, i.e. it is a question.
- is empty.

Such candidates are considered unlikely to answer the question.

4.4 Interfaces

Ephyra provides three user interfaces, which are discussed beneath. The command line interface and the web interface are text-based. The last interface allows for spoken interaction. All interfaces log the questions and the answers returned by the Ephyra system.

4.4.1 Command Line Interface

The command line interface allows to key in questions and displays a ranked list of answers, including their scores and supporting web documents.

Besides, it displays status and error messages that facilitate debugging. Examples of status messages are interpretations of the questions (property, target and context objects) and generated query strings. Error messages are e.g. exceptions thrown by the search APIs.

4.4.2 Web Interface

The functionality of the web interface[15] is similar to the command line interface, but it does not display any status or error messages to hide the internal state of the engine. The log file of this interface can help to improve the system and to judge its performance over time.

4.4.3 One4All

The Ephyra engine is integrated in the One4All[16] system as an agent, i.e. it can communicate with other agents through messages. The Ephyra agent waits for a message from the speech recognizer that contains a transcripted question string and poses the question to the engine. It then sends back a message that contains the top-ranked answer or a standard message, if the engine did not return any answers. This message is processed by the speech synthesizer.

This approach works in principle but suffers from various problems. Questions posed to a QA system typically contain named entities. I analyzed the questions from the TREC8 QA track and found out that there are named entities in 65% of the questions

[15] currently available at http://www.is.cs.cmu.edu:8080/EphyraWeb/

[16] see Section 2.3

(not counting country names). Unfortunately, the vocabulary of a speech recognizer is limited and thus named entities are often misinterpreted.
The transcripted questions from the speech recognizer are case insensitive and do not contain any punctuation marks. This complicates the recognition of named entities and can result in wrong POS tagging.
In addition, both speech recognition and question answering are prone to errors. These errors sum up in a combination of the two technologies.

5 Pattern Learning

The technique based on question interpretation and answer extraction (described in the Sections 4.1.4 and 4.3.6) uses different types of text patterns. Question patterns are used to interpret a question, i.e. to determine the property it asks for and to extract the target and context objects. Answer patterns are used to extract answers from text passages. This chapter describes how these patterns are generated.

Initially, the properties and question patterns need to be specified manually (see Section 5.1). In a second step, the system automatically learns answer patterns for each of the properties, using question-answer pairs as training data (Section 5.2).

5.1 Question Patterns

The current Ephyra system uses 67 properties with 1 to 20 question patterns each. These properties and corresponding question patterns were specified manually. For each property, there is a resource file that defines the question patterns. The resource file for the property NAME is shown in Figure 9. In this section, I describe how I generated the question patterns.

```
who <be> <T_NONE>
<what> <be> <T_NOABBR>
(who|<what>) <be> <T> (in|of) <C>
(who|<what>) <be> <C>'s <T>
(name the|<what> <be> the (names?|term) (for|given
     to|of)|who <be> considered to be) <T>
(name the|<what> <be> the (names?|term) (for|given
     to|of)|who <be> considered to be) <T> (in|of) <C>
(name the|<what> <be> the (names?|term) (for|given
     to|of)|who <be> considered to be) <C>'s <T>
what <T> <be> (called|known as|named)
what <T> (in|of) <C> <be> (called|known as|named)
```

```
what <C>'s <T> <be> (called|known as|named)
<T> <be> (called|known as|named) what
<T> (in|of) <C> <be> (called|known as|named) what
<C>'s <T> <be> (called|named) what
what do you call <T>
what do you call <T> (in|of) <C>
```

Figure 9 – Question patterns for the property NAME.

At first, I analyzed the questions from the TREC9 QA track (700 questions in total). I determined the properties that the questions ask for and created an initial set of question patterns by simply replacing the target and context objects in the questions by <*T*> and <*C*> tags respectively.

Then I converted the first letter to lower case, dropped the final punctuation mark and reordered auxiliary verbs in the same way as it is done by the question normalization component (Section 4.1.1). This is necessary because the question patterns are applied to a question string after normalization.

In the next step, I merged patterns by extending them to regular expressions (e.g. "who (assassinated|killed|murdered|shot) …"). I then used my personal knowledge and dictionaries such as WordNet[17] to further extend the patterns by adding synonyms, and I added patterns to cover other possible formulations I could think of.

To further generalize the patterns, I invented shortcuts, which are tags that represent regular expressions. These shortcuts are specified in a separate resource file. Table 2 shows the shortcuts and the regular expressions they stand for.

Shortcut	Regular expression
<be>	(is|are|was|were)
<that>	(that|which)
<what>	(what|which)

Table 2 – Shortcuts and corresponding regular expressions.

Sometimes it is hard to determine the property a question asks for. E.g. a question of the format "What is …?" could ask for the *DEFINITION* of a term, the *LONGFORM* of an abbreviation or the *NAME* of a place. To allow a definite interpretation of such questions, I invented a mechanism that uses object types. A target tag can be associated with an object type to restrict the format of a target object. Such target tags are not replaced by the general capturing group *(.*)* but by a more constraining regular expression. For example, the tag <*T_ABBR*> indicates that the target must be an abbreviation, i.e. a sequence of upper case letters. Table 3 gives an overview of the object types.

[17] http://wordnet.princeton.edu/

Object type	Meaning	Capturing group
<ABBR>	abbreviation	([A-Z0-9][A-Z0-9]+)
<NOABBR>	no abbreviation	(.*?[^A-Z0-9].*?)
<NE>	named entity	((?:an? |the)?[A-Z]\w*(?:.*?[A-Z]\w*)?)
<NONE>	no named entity	(.*? [^A-Z].*?|[^A-Z]\w*)

Table 3 – Object types, their meanings and the corresponding capturing groups.

Finally, I reviewed the properties and question patterns. I dropped rare properties and merged some properties with similar patterns. E.g. first I distinguished between the property *NAME* for proper names and the property *TERM* for technical terms, which turned out to be impractical.

5.2 Answer Patterns

While the question patterns need to be defined manually, the answer patterns (i.e. the patterns that are used to extract answers from text passages) are learned automatically.

The pattern learning algorithm is subdivided into two steps. In the first step (described in Section 5.2.1), the answer patterns are extracted from text snippets. Secondly, these patterns are assessed and unreliable or irrelevant patterns are dropped (see Section 5.2.2).

5.2.1 Pattern Extraction

The Ephyra system can learn answer patterns from question-answer pairs, e.g. from the TREC QA track. I trained the system on the 700 questions and answers from TREC9, which is the same training set that I used when I specified the question patterns (Section 5.1).

At first, the questions in the training set are interpreted by applying the question patterns. For each question, the target and context objects (if any) are extracted and the property that it asks for is determined. Questions that cannot be interpreted are discarded. If there is more than one interpretation for a question, only interpretations of minimal length are further considered (see Section 4.1.4).
Since I used the same questions when I specified the question patterns, only about 25% of the TREC9 questions could not be interpreted correctly. These questions often ask for a very specific property (e.g. “What is the wingspan of a condor?”) or their target object cannot be extracted without linguistic transformations[18]. Other questions specify the granularity of the property (e.g. “What year …?” asks for a *DATE* of granularity “year”) and thus cannot be handled by the current system.

[18] see example in Section 6.3

The interpretations of the training questions and the corresponding answers are written to output files; there is one file for each property. A line in an output file has the following format:

```
Target#Context 1#...#Context n#Answer#Regex
```

There can be *0* to *n* context objects. The last field is a regular expression that describes the answer and can be generated automatically, i.e. it is optional. Figure 10 shows an example output file for the property *LEADER*.

```
Australia#"Paul Keating"#Paul Keating
Japan#"Akhito"#Akhito
Bolivia#"Victor Paz Estenssoro"#Victor Paz Estenssoro
United Kingdom#"Elizabeth II"#Elizabeth II
Cuba#"Fidel Castro"#
Russia#"Wladimir Putin"#
```

Figure 10 – Question interpretations for the property LEADER.

Now the output files can be reviewed manually and tuples can be added for properties that have not sufficiently been covered by the training questions.

In the next step, a search engine query is generated for each tuple. It comprises the target object, any context objects and the answer string (i.e. the property).
In the running example ("How many calories are there in a Big Mac?"; see Sections 4.1.4 and 4.3.6), the query string is:

```
"calories" "Big Mac" "560"
```

The query strings are then submitted to the knowledge miners (currently there are knowledge miners for Google and Yahoo) and the snippets are collected. All occurrences of target, context and property objects are replaced by the tags <*T*>, <*C*> and <*P*> respectively.
E.g. the snippet "One Big Mac contains 560 calories and 31 grams of fat." is transformed into "One <*C*> contains <*P*> <*T*> and 31 grams of fat.".

Now the answer patterns can be extracted from the snippets. This is done by applying the following two regular expressions[19]:

- `\B<T>\B(.*?)\B<P>\B\s*(\W|\w+)`
- `(\W|\w+)\s*\B<P>\B(.*?)\B<T>\B`

An answer pattern is basically a regular expression that covers a target tag (<*T*>), a property tag (<*P*>) and any characters in between these tags. In addition, it covers one word or special character preceding or following the <*P*> tag (depending on whether the <*P*> tag comes before or after the <*T*> tag). This is necessary to know where a property object begins or ends.
Similarly to question patterns (see Section 5.1), some words are replaced by shortcuts, e.g. "is", "are", "was" and "were" are all replaced by the tag <*be*>. This is to create more generic patterns.
In our example, the second of the above patterns can be used to extract the following answer pattern:

[19] Similar to the patterns used by the LAMP QA system, see [Zhang02]

```
contains <P> <T>
```

Finally, the answer patterns are written to output files. Again, there is one file for each property.

5.2.2 Pattern Assessment and Filtering

For some properties, the pattern learning algorithm extracted too many answer patterns to allow their application in a reasonable time (e.g. ~8000 patterns for the property *PLACE*).
Some of these patterns are too specific, e.g. they contain named entities or large numbers, and thus most likely cannot be used to extract answers to questions other than those in the training set. Other patterns are too general and therefore do not reliably extract correct answers.
To address this issue, the pattern learning algorithm assesses the answer patterns and drops patterns that are too specific or unreliable.

In the previous step, a set of resource files containing question interpretations and corresponding answers was created. This training data can be reused to assess the answer patterns. Alternatively, a different set of question-answer pairs can be used to generate a new set of resource files. The assessment of the answer patterns is done automatically but yet it is time-intensive, so it may be necessary to cut down the training data to reduce the runtime.

Similarly to the previous step, the tuples from the resource files are transformed into search engine queries, snippets are fetched and all occurrences of target, context and property objects are replaced by the respective tags.

The algorithm then applies each of the previously extracted answer patterns to all the snippets for the respective property (e.g. patterns for the property *CAPITAL* are only applied to snippets that contain an answer to a *CAPITAL* question). The extracted answers are evaluated automatically, using a regular expression (the last field in the respective tuple).
For each answer pattern *a*, the algorithm records how often it could be used to extract a correct answer *(#* $correct_a$) and how often it extracted a wrong answer *(#* $incorrect_a$). Furthermore, for each property *p*, the total number of snippets is recorded *(#* $snippets_p$). These values are used to compute the following two measures:

$$Confidence_a = \frac{\#correct_a}{\#correct_a + \#incorrect_a}$$

$$Support_a = \frac{\#correct_a}{\#snippets_p}$$

These measures are then compared to thresholds. Patterns with a low confidence are considered to be unreliable and are dropped. A low support value means that a pattern is too specific and thus it is also dropped.

The remaining patterns are saved to resource files, along with their values for # *correct* and # *incorrect*. The answer extraction component (described in Section 4.3.6) reuses these values to compute confidence measures for the answers. Figure 11 shows the top 5 (out of about 800) answer patterns for the property *DATEOFBIRTH*, sorted by # *correct*.[20]

```
<T> \(<P>-
746
103
<T> \(<P> -
138
55
<T> <be> born on <P> in
114
102
<T> <P>-
53
1248
<T>,<P>-
53
136
```

Figure 11 – Top 5 answer patterns for the property DATEOFBIRTH (along with #correct and #incorrect).

5.3 Discussion

The interpretation of a question is largely independent of the original formulation of the question. Thus, this approach works even if questions and answers use different terms (e.g. "Who killed John F. Kennedy?" – "Kennedy was assassinated by Lee Harvey Oswald."). This is important if there are only few occurrences of the answer in the knowledge source.
Furthermore, the answer strings that are extracted are usually exact. They just contain the answer and no unnecessary information.

A question interpretation comprises a target object plus an arbitrary number of context objects. While the LAMP approach (Section 3.3) cannot handle questions with multiple key phrases, Ephyra can deal with such questions (e.g. the running example "How many calories are there in a Big Mac?" contains the key phrases "calories" and "Big Mac").

The classification of questions based on properties seems to be optimal, since there are just enough classes to ensure that the right aspect of a target is extracted. However, the performance can be increased by adding additional properties that are currently covered by a "backup property". E.g. the question "What is the name of the wife of Bill Clinton?" asks for the *NAME* of the target "wife of Bill Clinton" or, more specifically, for the *WIFE* of Bill Clinton. The second interpretation has the advantage

[20] Note that the patterns are regular expressions, thus metacharacters are preceded by "\".

of being more independent of the formulation of the question (e.g. “wife” or “spouse”). On the other hand, additional properties require more training data.

The pattern learning algorithm extracts and assesses patterns for each of the properties independently. Thus, if the system does not perform well for a specific property, it is possible to revise the question patterns or training data and rerun the algorithm for just that property.

6 Experiments and Results

6.1 Experimental Setup

I evaluated the Ephyra system on the questions from the question answering track of TREC8 [Voor99]. The test set comprises 200 questions and corresponding answer patterns, i.e. regular expressions that describe correct answers. See Appendix A.1 for a complete list of the questions.[21]

For each of the questions, Ephyra returned a maximum of 5 answers. The answers were compared to the patterns provided by TREC. Answers that did not match the patterns were reviewed manually to judge whether the answers were correct or not. For two of the questions (131: "Which Japanese car maker had its biggest percentage of sale in the domestic market?" and 184: "When was Queen Victoria born?"), there were no answer patterns. Since Ephyra did not return any reasonable answers for question 131, I just assumed the answers to be wrong. For question 184, I assumed the correct answer to be "1819".[22]

Another issue is that the TREC evaluation was conducted on the AQUAINT corpus[23] and not on the web. The patterns provided by TREC are good to recognize correct answers from that corpus (at least they cover the correct answers returned by the participants of the evaluation) but not from the web.
For this reason, I also accepted answers that did not match the patterns. However, I did this in a conservative manner and I marked such answers.
I considered an answer to be correct, if it is at least as precise as the answer pattern. E.g. if the answer pattern is "February 1994", then the answer "1994" is wrong, but "February 14th 1994" is correct (assuming that this is really the correct date).

[21] The answer patterns can be found on the web site of TREC: http://trec.nist.gov/

[22] from http://en.wikipedia.org/wiki/Queen_victoria/

[23] distributed by the LDC: http://www.ldc.upenn.edu/Catalog/docs/LDC2002T31/

For some questions, the answers provided by TREC are outdated. The AQUAINT corpus comprises newspaper articles from the years 1996 to 2000. E.g. Ephyra's answer to question 155 ("Who was the Democratic nominee in the American presidential election?") was "Kerry", while the answer pattern suggests "Clinton". I assumed "Kerry" to be a correct answer.
Finally, some questions are ambiguous, e.g. question 178 ("What is the capital of Congo?"). There are two countries called Congo, so there are two correct answers, "Kinshasa" and "Brazzaville".

Furthermore, I distinguished between answers that are correct and answers that are exact. An exact answer is correct and does not contain any unnecessary words, i.e. it is either the fact that was asked for (e.g. "1819") or an answer sentence (e.g. "Queen Victoria was born in 1819."). In contrast, the answer "Born in 1819, Victoria was Queen of the United Kingdom." is correct but not exact.

6.2 Results

The complete results per question can be found in Appendix A.2. The second column specifies the technique that was used to obtain the answers and can be one of the following:

- *Knowledge Annotation:* See Section 4.2.2
- *PROP:* Pattern learning approach using the patterns for the property PROP (See Sections 4.1.4, 4.3.6 and Chapter 5)
- *Backup:* Query reformulation and filtering techniques (See Sections 4.1.2, 4.1.3, 4.3.1 – 4.3.5)
- *None:* Ephyra did not return any answers

The third column shows the rank of the first correct (or exact) answer or *0*, if all the answers were wrong.
The last column indicates the quality of that answer and can be a combination of the following shortcuts:

- *W:* wrong answer
- *C:* correct answer
- *E:* exact answer (see previous chapter)
- *P:* answer matches TREC pattern

Table 4 is a summarization of the results for each of the above techniques and the overall system. It shows the number of questions that have been addressed, the number of correct and exact answers and the sum of these values.
The *precision* is the percentage of questions for which at least one of the up to five answers is correct or exact. The rank of this answer is irrelevant for this measure.

$$Precision = \frac{\#Total\ Correct}{\#Questions}$$

To take the rank of the answer into account, I use the *Mean Reciprocal Rank (MRR)*, which is the average of the reciprocal ranks over all questions; it is used within the

TREC evaluation since TREC8. The reciprocal rank is the inverse of the rank of the first correct (or exact) answer or *0*, if all answers are wrong:

$$MRR = \frac{1}{n}\sum_{i=1}^{n} RR(question_i),$$

$$RR(question_i) = \begin{cases} 0, & Rank = 0 \\ \frac{1}{Rank}, & Rank \in \{1,...,5\} \end{cases}$$

	Knowl. Annot.	Pattern Learn.	Backup	All
# Questions	4	96	99	200
# Correct Answers	0	8	43	51
# Exact Answers	3	47	4	54
# Total Correct	3	55	47	105
Precision	0,75	0,57	0,47	**0,53**
MRR	0,75	0,40	0,32	**0,36**

Table 4 – Overview of the performance for each of the techniques.

Figure 12 shows the distribution of the questions that were addressed using the pattern learning approach (96 questions in total). For each of the properties, it shows the number of correctly and wrongly answered questions.

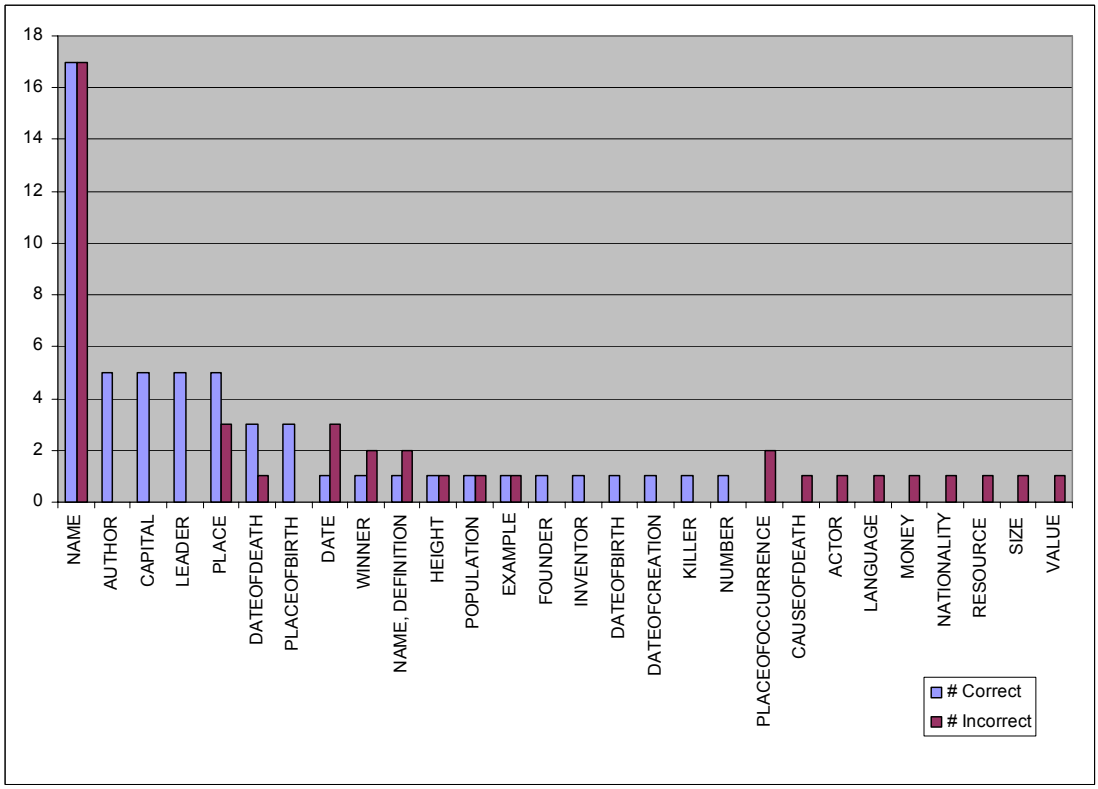

Figure 12 – Number of correct and incorrect answers for each property.

6.3 Discussion and Comparison

An MRR of 0.36 is a good result taking into account that the MRR of the 41 systems that participated in the TREC8 QA track vary in the range from *0.06* to *0.66*. Therefore, Ephyra can be found somewhere in the upper middle class.
However, these results have to be compared with care for several reasons. First of all, Ephyra was not specifically tailored for this domain. In this project, I focused on developing and integrating different techniques rather than optimizing the system for an evaluation. Besides, more than 50% of the answers returned by Ephyra were exact answers, which was not required for the TREC8 evaluation.
In addition, the evaluation was conducted on the AQUAINT corpus, which was guaranteed to contain answers to all the questions, while I used the web as a knowledge source. Since the answer patterns provided by TREC are not suitable for answers extracted from the web, I had to judge answers manually, which I did conservatively (see Section 6.1).
Furthermore, the questions from the old TREC QA tracks are often cited on web sites that deal with QA. When searching for the keywords in these question, web search engines often return such web sites rather than sites that provide the answers.

The results show that the pattern learning approach has a high precision, but on the other hand a relatively low recall. Therefore, it is reasonable to combine it with a backup technique. The confidence score of the answers is a good indicator for their reliability.

The pattern learning approach not only has a higher precision than the backup (based on query reformulations and filtering), but it also has the benefit of mainly returning exact answers instead of sentences or sentence fragments. This allows a human user to perceive the answer at a glance and a computer system can further process the answer. In addition, the performance of the pattern learning approach can be boosted by increasing the amount or quality of the training data.

Figure 12 shows that the pattern learning approach can answer certain classes of questions, such as *AUTHOR*, *CAPITAL* or *LEADER* (e.g. of a country or company), quite reliably. The performance drops when it comes to long questions or questions that ask for a concrete number, e.g. the *SIZE* of a region or the monetary *VALUE* of an award. Note that the number of test questions is not sufficient to judge the performance for properties such us *LANGUAGE* or *NATIONALITY*.

There is another interesting conclusion that can be drawn from Figure 12. The patterns for the property *NAME* were applied exceptionally often. This class serves as a backup for questions that do not fit into any of the other classes. E.g. the question “Who was President Cleveland's wife?” could be handled by a class *WIFE*, but since this class is not defined, the most appropriate class is *NAME*.

Still, about 50% of the questions had to be handled using backup techniques. For some questions, there was no appropriate class or the question patterns did not cover the questions and thus the questions could not be interpreted.

Other questions are just not suitable for this approach because the target cannot be extracted without complicated semantic transformations. E.g. the target of the question "When did Spain and Korea start ambassadorial relations?" would be "start of ambassadorial relations between Spain and Korea".
Other issues are faulty transformations of auxiliaries due to wrong POS tagging and grammatical errors in the test set (e.g. 176: "How many people in Tucson?").

The pattern learning approach that I use in Ephyra is inspired by the LAMP QA system [Zhang02]. In fact, it is a generalization of the LAMP approach.
The authors of LAMP also evaluated their system on the TREC8 questions, using the TREC9 and TREC10 questions as training data. In contrast, I only used the TREC9 questions but extended the training data for some classes manually.
Table 5 shows the total number of test questions (definition questions and questions without answer patterns were discarded), the number of questions the system returned some answer for, the number of correctly answered questions, the MRR restricted to the questions that were answered and the MRR over all questions.

# Questions	*# Answered*	*# Correct*	*MRR answered*	*MRR all*
196	93	52	**0.46**	**0.22**

Table 5 – Performance of the LAMP QA system on the TREC8 questions (taken from [Zhang02]).

Note that it is hard to compare these numbers since the authors of LAMP may have judged correct answers that did not match the TREC patterns differently. Furthermore, they discarded some questions.
I defined the properties and the associated question patterns without looking at the TREC8 questions. As a consequence, some questions that could in general be handled with my approach could not be interpreted.
On the other hand, the authors of LAMP did not use any backup techniques for the remaining questions. While their performance restricted to the answered questions is a little bit better, the Ephyra system has a much better overall performance.

7 Conclusion and Outlook

7.1 Summary

In this project, I developed the Ephyra QA system, which is designed to be a modular framework that can be used to efficiently implement different approaches to QA. Ephyra is basically a library of components for query formation, information retrieval and answer selection. A QA engine can be assembled dynamically from these components.
A developer that e.g. wants to try out a new technique for answer extraction can simply implement an additional answer filter without caring about the query formation or search.

The current system is tailored for the English language, but it is adaptable to other languages. Language-specific code is separated from language-independent code and textual patterns are defined in resource files rather than in the source code.

Ephyra supports the two major approaches to question answering, knowledge annotation and knowledge mining. Two distinct types of searchers are used to browse the unstructured web and to extract answers from semi-structured sources. Knowledge annotation allows to reliably answer certain classes of questions while knowledge mining can deal with any factoid questions.

Based on this framework, I developed a new approach that uses text patterns to interpret questions and to extract answers from relevant documents. The patterns for answer extraction are learned automatically.
This approach has two major advantages. It is mostly independent of the formulation of the question and the answer and it usually returns exact answers.

The Ephyra QA engine is a Java API than can easily be integrated into other NLP systems. So far, I implemented two text-based interfaces, a command line interface and a web interface, as well as a speech interface. Beyond that, Ephyra can be deployed in other systems, e.g. a dialog system, as a back-end service.

7.2 Future Work

The pattern learning approach discussed in Chapter 5 can be extended to also learn question patterns automatically. Initially, the properties and for each property an small set of questions patterns would be specified manually and the system would learn answer patterns as described in Chapter 5. These answer patterns could then be used to learn more question patterns from additional question-answer pairs. By searching for the keywords of a question and applying all the answer patterns to the fetched text passages, the best fitting property can be determined. This is the property whose answer patterns extracted the best answers. But it can be challenging to extract the target and context objects from the question and to transform it into a generic question pattern.
Another improvement of this approach would be to invent generic patterns, such as *TRANSLATION<SPANISH>*, meaning that a question asks for a translation of a word, in particular the translation into Spanish. It would also be useful to support different granularities of properties. This would allow to handle questions such as "In which year ...?" and "In what city …?". These questions ask for properties of type *DATE* and *PLACE* respectively, but further specify the desired granularity ("year" and "city").
Sometimes, a question asks for a property of a combination of target objects, e.g. the *DISTANCE* between two places. The pattern learning approach could be extended to support multiple targets.
The Ephyra system applies a stopword filter to the answers, but this filter does not distinguish between the various properties. Property-specific filters could e.g. ensure that the answer to a question for the property *NUMBER* is really numerical or that the answer to a *PLACE* question is a location.

The current Ephyra system is non-interactive. An interactive system based on a dialogue manager would have various advantages.
The QA engine could ask for additional information to expand or narrow down the search. E.g. if the user asks a question such as "What is the name of the disease that causes involuntary movements?", the system could ask for further details: "Can you name any other symptoms of that disease?".
When confronted with an ambiguous question, the system could ask for clarification: "What is the capital of Congo?" – "There are two countries called Congo, the Republic of Congo and the Democratic Republic of Congo. Which one do you mean?"
On the other hand, a mixed-initiative system could also allow the user to pose further inquiries. The user could ask for an explanation of the answer, e.g. "What is meant by the term …?" or could ask follow-up questions, e.g.: "When did the French Revolution start?" "And when did it end?".
Furthermore, meta-communication would allow feedback on the correctness of the answer. The system could then automatically adjust configuration parameters.
While ambiguity is hard to address, particularly for open-domain QA systems that deploy an unstructured database, user feedback and the request for extra information (depending on the number of search results) are feasible.
[Akker05] discusses possible features of an interactive QA system.

The annotation of (semi-)structured web sites and services is a time-consuming task. Each time the layout of a web site is changed, the knowledge annotation component needs to be adjusted.
This task could at least partially be automated. Given question-answer pairs as training data, the system could automatically learn how to extract answers.

Computers tend to be weak in judging whether they are right or not. In the field of question answering, it would be desirable that a QA system returns a reliable confidence measure for an answer. In case of a low confidence, it might be better to return no answer rather than a wrong answer. The scores assigned to answers during answer selection, in particular the scores assigned by the pattern learning approach, could serve as reliable measures.
Based on confidence scores, system parameters could automatically be optimized to maximize the scores for a given set of questions.

Question answering evaluations are usually based on closed text corpora, e.g. the TREC QA track is conducted on the AQUAINT corpus. To participate in an evaluation, the Ephyra system would need to be extended to either search directly on a closed corpus or to project answers found in the web on the corpus (i.e. find a supporting document in the corpus).
While the web and the algorithms used by web search engines change over time, question answering on a closed text corpus facilitates the comparison of results.

Finally, it is possible to deploy automatic summarization for question answering. In the pattern learning approach described in Chapter 5, answer patterns are extracted from text passages. It is possible to generate more generic patterns by first summarizing the text passages before extracting the patterns. Also, before applying the answer patterns to a document, the document could be summarized.
About 50% of the correct answers returned by the Ephyra system are not exact in the sense that they contain unnecessary words. Automatic summarization could be used to trim these answers to make it easier for a human to extract the desired information. Summarization could also help to improve the results in an evaluation. E.g. the performance measure for the “Other” questions in the TREC 2004 QA track (see [Voor04]) depends on the total length of the answer.

Acknowledgement

I would like to thank my advisors, Petra Gieselmann at the University of Karlsruhe and Thomas Schaaf at Carnegie Mellon University, for their support and patience. Thomas has been a great advisor during my stay in Pittsburgh and helped me to develop new ideas in numerous discussions. Petra continuously supported me with valuable tips and directions and by proof-reading this document.

My special thanks go to Prof. Alexander Waibel, who gave me the opportunity to do this research at the Interactive Systems Labs in Pittsburgh.

Thanks also go to Matthias Eck, who helped me to set up an application server to host the web interface, and Hartwig Holzapfel, who supported me in integrating the Ephyra system in One4All.

A Evaluation

A.1 TREC8 Questions

1. Who is the author of the book, "The Iron Lady: A Biography of Margaret Thatcher"?
2. What was the monetary value of the Nobel Peace Prize in 1989?
3. What does the Peugeot company manufacture?
4. How much did Mercury spend on advertising in 1993?
5. What is the name of the managing director of Apricot Computer?
6. Why did David Koresh ask the FBI for a word processor?
7. What debts did Qintex group leave?
8. What is the name of the rare neurological disease with symptoms such as: involuntary movements (tics), swearing, and incoherent vocalizations (grunts, shouts, etc.)?
9. How far is Yaroslavl from Moscow?
10. Name the designer of the shoe that spawned millions of plastic imitations, known as "jellies".
11. Who was President Cleveland's wife?
12. How much did Manchester United spend on players in 1993?
13. How much could you rent a Volkswagen bug for in 1966?
14. What country is the biggest producer of tungsten?
15. When was London's Docklands Light Railway constructed?
16. What two US biochemists won the Nobel Prize in medicine in 1992?
17. How long did the Charles Manson murder trial last?
18. Who was the first Taiwanese President?
19. Who was the leader of the Branch Davidian Cult confronted by the FBI in Waco, Texas in 1993?
20. Where is Inoco based?
21. Who was the first American in space?
22. When did the Jurassic Period end?
23. When did Spain and Korea start ambassadorial relations?
24. When did Nixon visit China?
25. Who was the lead actress in the movie "Sleepless in Seattle"?
26. What is the name of the "female" counterpart to El Nino, which results in cooling temperatures and very dry weather?
27. Where did the 6th annual meeting of Indonesia-Malaysia forest experts take place?
28. Who may be best known for breaking the color line in baseball?
29. What is the brightest star visible from Earth?
30. What are the Valdez Principles?
31. Where was Ulysses S. Grant born?
32. Who received the Will Rogers Award in 1989?
33. What is the largest city in Germany?
34. Where is the actress, Marion Davies, buried?
35. What is the name of the highest mountain in Africa?
36. In 1990, what day of the week did Christmas fall on?
37. What was the name of the US helicopter pilot shot down over North Korea?

38. Where was George Washington born?
39. Who was chosen to be the first black chairman of the military Joint Chiefs of Staff?
40. Who won the Nobel Peace Prize in 1991?
41. What is the legal blood alcohol limit for the state of California?
42. What was the target rate for M3 growth in 1992?
43. What costume designer decided that Michael Jackson should only wear one glove?
44. Who is the director of the international group called the Human Genome Organization (HUGO) that is trying to coordinate gene-mapping research worldwide?
45. When did Lucelly Garcia, a former ambassador of Columbia to Honduras, die?
46. Who is the mayor of Marbella?
47. What company is the largest Japanese ship builder?
48. Where is the massive North Korean nuclear complex located?
49. Who fired Maria Ybarra from her position in San Diego council?
50. When was Dubai's first concrete house built?
51. Who is the president of Stanford University?
52. Who invented the road traffic cone?
53. Who was the first doctor to successfully transplant a liver?
54. When did Nixon die?
55. Where is Microsoft's corporate headquarters located?
56. How many calories are there in a Big Mac?
57. What is the acronym for the rating system for air conditioner efficiency?
58. Name a film that has won the Golden Bear in the Berlin Film Festival?
59. Who was President of Costa Rica in 1994?
60. What is the fare cost for the round trip between New York and London on Concorde?
61. What brand of white rum is still made in Cuba?
62. What is the name of the chronic neurological autoimmune disease which attacks the protein sheath that surrounds nerve cells causing a gradual loss of movement in the body?
63. What nuclear-powered Russian submarine sank in the Norwegian Sea on April 7, 1989?
64. Who is the voice of Miss Piggy?
65. Name a country that is developing a magnetic levitation railway system?
66. Name the first private citizen to fly in space.
67. What is the longest river in the United States?
68. What does El Nino mean in spanish?
69. Who came up with the name, El Nino?
70. How many lives were lost in the China Airlines' crash in Nagoya, Japan?
71. In what year did Joe DiMaggio compile his 56-game hitting streak?
72. When did the original Howdy Doody show go off the air?
73. Where is the Taj Mahal?
74. Who leads the star ship Enterprise in Star Trek?
75. What cancer is commonly associated with AIDS?
76. In which year was New Zealand excluded from the ANZUS alliance?
77. Who played the part of the Godfather in the movie, "The Godfather"?
78. Which large U.S. city had the highest murder rate for 1988?
79. What did Shostakovich write for Rostropovich?
80. What is the name of the promising anticancer compound derived from the pacific yew tree?
81. How many inhabitants live in the town of Ushuaia?
82. How many consecutive baseball games did Lou Gehrig play?
83. What is the tallest building in Japan?
84. Which country is Australia's largest export market?
85. Which former Ku Klux Klan member won an elected office in the U.S.?
86. Who won two gold medals in skiing in the Olympic Games in Calgary?
87. Who followed Willy Brandt as chancellor of the Federal Republic of Germany?
88. What is Grenada's main commodity export?
89. At what age did Rossini stop writing opera?
90. Who is the founder of Scientology?
91. Which city in China has the largest number of foreign financial companies?
92. Who released the Internet worm in the late 1980s?
93. Who first circumnavigated the globe?
94. Who wrote the song, "Stardust"?
95. What country is the worlds leading supplier of cannabis?
96. What time of day did Emperor Hirohito die?
97. How large is the Arctic refuge to preserve unique wildlife and wilderness value on Alaska's north coast?
98. Where is the highest point in Japan?
99. What is the term for the sum of all genetic material in a given organism?
100. What is considered the costliest disaster the insurance industry has ever faced?
101. How many people live in the Falklands?

102. Who is the Voyager project manager?
103. How many people died when the Estonia sank in 1994?
104. What language is most commonly used in Bombay?
105. How many people does Honda employ in the U.S.?
106. What is the second highest mountain peak in the world?
107. When was China's first nuclear test?
108. Which company created the Internet browser Mosaic?
109. Where does Buzz Aldrin want to build a permanent, manned space station?
110. Who killed Lee Harvey Oswald?
111. How long does it take to travel from Tokyo to Niigata?
112. Who is the President of Ghana?
113. What is the name of the medical condition in which a baby is born without a brain?
114. How much stronger is the new vitreous carbon material invented by the Tokyo Institute of Technology compared with the material made from cellulose?
115. What is Head Start?
116. Which team won the Super Bowl in 1968?
117. What two researchers discovered the double-helix structure of DNA in 1953?
118. What percentage of the world's plant and animal species can be found in the Amazon forests?
119. What Nobel laureate was expelled from the Philippines before the conference on East Timor?
120. Who held the endurance record for women pilots in 1929?
121. Who won the first general election for President held in Malawi in May 1994?
122. Who is section manager for guidance and control systems at JPL?
123. How many Vietnamese were there in the Soviet Union?
124. What was Agent Orange used for during the Vietnam War?
125. In what city is the US Declaration of Independence located?
126. When did Israel begin turning the Gaza Strip and Jericho over to the PLO?
127. Which city has the oldest relationship as a sister-city with Los Angeles?
128. Who was the second man to walk on the moon?
129. How many times was pitcher, Warren Spahn, a 20-game winner in his 21 major league seasons?
130. When was Yemen reunified?
131. Which Japanese car maker had its biggest percentage of sale in the domestic market?
132. What is the capital of Uruguay?
133. What is the name for the technique of growing certain plants in soils contaminated with toxic metals, wherein the plants take up the toxic metals, are harvested, and the metals recovered for recycling?
134. Where is it planned to berth the merchant ship, Lane Victory, which Merchant Marine veterans are converting into a floating museum?
135. What famous communist leader died in Mexico City?
136. Who is the Queen of Holland?
137. Who is the president of the Spanish government?
138. What is the name of the normal process in all living things, including humans, in which cells are programmed to "commit suicide"?
139. How many people did the United Nations commit to help restore order and distribute humanitarian relief in Somalia in September 1992?
140. How many people on the ground were killed from the bombing of Pan Am Flight 103 over Lockerbie, Scotland, December 21, 1988?
141. What is the duration of the trip from Bristol to London by rail?
142. What is the population of Ulan Bator, capital of Mongolia?
143. Where does most of the marijuana entering the United States come from?
144. How many megawatts will the power project in Indonesia, built by a consortium headed by Mission Energy of US, produce?
145. What did John Hinckley do to impress Jodie Foster?
146. In what year did Ireland elect its first woman president?
147. Who is the prime minister of Japan?
148. How many soldiers were involved in the last Panama invasion by the United States of America?
149. Where is the Bulls basketball team based?
150. What is the length of border between the Ukraine and Russia?
151. Where did Dylan Thomas die?
152. How many people live in Tokyo?
153. What is the capital of California?
154. How many Grand Slam titles did Bjorn Borg win?
155. Who was the Democratic nominee in the American presidential election?
156. When was General Manuel Noriega ousted as the leader of Panama and turned over to U.S. authorities?
157. Where is Dartmouth College?
158. How many mines can still be found in the Falklands after the war ended?
159. Why are electric cars less efficient in the north-east than in California?

160. When did French revolutionaries storm the Bastille?
161. How rich is Bill Gates?
162. What is the capital of Kosovo?
163. What state does Charles Robb represent?
164. Who is the leading competitor of Trans Union Company?
165. Which type of submarine was bought recently by South Korea?
166. When did communist control end in Hungary?
167. What nationality is Pope John Paul II?
168. Who was the captain of the tanker, Exxon Valdez, involved in the oil spill in Prince William Sound, Alaska, 1989?
169. Whom did the Chicago Bulls beat in the 1993 championship?
170. Who was President of Afghanistan in 1994?
171. Who is the director of intergovernmental affairs for the San Diego county?
172. Where is the Keck telescope?
173. How many moons does Jupiter have?
174. When did Jaco Pastorius die?
175. When did beethoven die?
176. How many people in Tucson?
177. How tall is Mt. Everest?
178. What is the capital of Congo?
179. What is the capital of Italy?
180. What is the capital of Sri Lanka?
181. What novel inspired the movie BladeRunner?
182. What was the first Gilbert and Sullivan opera?
183. What was the name of the computer in "2001: A Space Odyssey"?
184. When was Queen Victoria born?
185. When was the battle of the Somme fought?
186. Where did the Battle of the Bulge take place?
187. Where was Lincoln assassinated?
188. When was the women's suffrage amendment ratified?
189. Where is Qatar?
190. Where is South Bend?
191. Where was Harry Truman born?
192. Who was Secretary of State during the Nixon administration?
193. Who was the 16th President of the United States?
194. Who wrote "The Pines of Rome"?
195. Who wrote "Dubliners"?
196. Who wrote "Hamlet"?
197. What did Richard Feynman say upon hearing he would receive the Nobel Prize in Physics?
198. How did Socrates die?
199. How tall is the Matterhorn?
200. How tall is the replica of the Matterhorn at Disneyland?

A.2 Results

Question	Technique	Rank	Quality
1	AUTHOR	1	EP
2	Backup	0	W
3	Backup	0	W
4	Backup	0	W
5	NAME	1	CP
6	Backup	0	W
7	Backup	1	C
8	Backup	0	W
9	Backup	3	C
10	Backup	0	W
11	NAME	5	E
12	Backup	0	W
13	Backup	0	W
14	Backup	0	W
15	DATEOFCREATION	2	EP
16	Backup	0	W
17	Backup	2	CP
18	NAME	0	W
19	Backup	1	CP
20	None	0	W
21	NAME	0	W
22	DATE	0	W
23	Backup	0	W
24	Backup	1	C
25	NAME	0	W
26	Backup	0	W
27	PLACEOFOCCURRENCE	0	W
28	Backup	0	W
29	NAME, DEFINITION	1	EP
30	NAME, DEFINITION	0	W
31	PLACEOFBIRTH	1	E
32	Backup	0	W
33	NAME	2	EP
34	Backup	0	W
35	NAME	2	EP
36	Backup	0	W
37	Backup	3	C
38	PLACEOFBIRTH	1	EP
39	NAME	0	W
40	WINNER	4	CP
41	NAME	0	W
42	NAME	0	W
43	Backup	0	W
44	Backup	0	W
45	DATEOFDEATH	0	W
46	NAME	2	E
47	Backup	0	W
48	PLACE	0	W
49	Backup	0	W
50	Backup	4	CP
51	LEADER	5	EP
52	INVENTOR	4	CP
53	NAME	0	W
54	DATEOFDEATH	2	EP
55	PLACE	1	E
56	NUMBER	1	E

Question	Technique	Rank	Quality
101	POPULATION	0	W
102	NAME	2	E
103	Backup	0	W
104	LANGUAGE	0	W
105	Backup	1	C
106	NAME	1	EP
107	DATE	3	EP
108	Backup	3	CP
109	Backup	0	W
110	KILLER	3	EP
111	Backup	0	W
112	LEADER	2	E
113	Backup	0	W
114	Backup	0	W
115	Knowledge Annotation	1	E
116	Backup	0	W
117	Backup	0	W
118	Backup	0	W
119	Backup	0	W
120	Backup	1	C
121	WINNER	0	W
122	NAME	0	W
123	Backup	0	W
124	Backup	0	W
125	Backup	0	W
126	Backup	4	CP
127	Backup	1	CP
128	NAME	1	EP
129	Backup	0	W
130	Backup	1	CP
131	Backup	0	W
132	Knowledge Annotation	1	EP
133	NAME	0	W
134	Backup	0	W
135	Backup	1	CP
136	NAME	0	W
137	LEADER	2	C
138	NAME	0	W
139	Backup	0	W
140	Backup	0	W
141	NAME	2	E
142	Knowledge Annotation	0	W
143	RESOURCE	0	W
144	Backup	0	W
145	Backup	1	C
146	Backup	3	CP
147	LEADER	2	E
148	Backup	0	W
149	Backup	1	CP
150	NAME	2	EP
151	Backup	0	W
152	POPULATION	3	E
153	CAPITAL	1	EP
154	Backup	2	CP
155	NAME	1	E
156	DATE	0	W

57	Backup	0	W
58	EXAMPLE	0	W
59	NAME	0	W
60	VALUE	0	W
61	Backup	2	CP
62	Backup	0	W
63	Backup	2	CP
64	NAME	1	EP
65	EXAMPLE	2	CP
66	NAME	0	W
67	NAME	5	EP
68	Backup	1	E
69	Backup	1	E
70	Backup	2	C
71	Backup	1	CP
72	Backup	1	CP
73	PLACE	1	EP
74	LEADER	2	CP
75	Backup	2	CP
76	Backup	4	CP
77	ACTOR	0	W
78	Backup	4	CP
79	Backup	5	CP
80	Backup	5	CP
81	Backup	1	CP
82	Backup	2	C
83	NAME	3	CP
84	Backup	0	W
85	Backup	5	C
86	WINNER	0	W
87	Backup	0	W
88	NAME	0	W
89	Backup	1	C
90	FOUNDER	2	EP
91	Backup	4	CP
92	Backup	0	W
93	Backup	0	W
94	AUTHOR	3	E
95	Backup	0	W
96	Backup	0	W
97	SIZE	0	W
98	PLACE	0	W
99	Backup	3	C
100	NAME, DEFINITION	0	W
157	PLACE	1	EP
158	Backup	0	W
159	Backup	1	CP
160	Backup	1	CP
161	MONEY	0	W
162	CAPITAL	1	EP
163	Backup	1	CP
164	Backup	3	CP
165	Backup	1	C
166	Backup	0	W
167	NATIONALITY	0	W
168	Backup	0	W
169	Backup	2	CP
170	NAME	0	W
171	NAME	0	W
172	PLACE	1	EP
173	Backup	1	E
174	DATEOFDEATH	3	CP
175	DATEOFDEATH	5	EP
176	Backup	4	E
177	HEIGHT	4	EP
178	CAPITAL	1	EP
179	CAPITAL	1	EP
180	CAPITAL	1	EP
181	Backup	2	C
182	NAME	1	EP
183	NAME	2	EP
184	DATEOFBIRTH	1	E
185	DATE	0	W
186	PLACEOFOCCURRENCE	0	W
187	PLACE	0	W
188	Backup	1	C
189	Knowledge Annotation	1	EP
190	PLACE	1	EP
191	PLACEOFBIRTH	1	E
192	NAME	0	W
193	NAME	1	EP
194	AUTHOR	1	EP
195	AUTHOR	1	EP
196	AUTHOR	1	EP
197	Backup	0	W
198	CAUSEOFDEATH	0	W
199	HEIGHT	0	W
200	Backup	1	CP

Table 6 – Results of the evaluation of Ephyra on the TREC8 questions.

Bibliography

[Akker05] R. op den Akker, H. Bunt, S. Keizer, B. van Schooten. *From Question Answering to Spoken Dialogue: Towards an Information Search Assistant for Interactive Multimodal Information Extraction.* Interspeech, 2005.

[Banko02] Michele Banko, Eric Brill, Susan Dumais, Jimmy Lin. *AskMSR: Question answering using the World Wide Web.* In Proceedings of 2002 AAAI Spring Symposium on Mining Answers from Texts and Knowledge Bases, 2002.

[Brill01] E. Brill, J. Lin, M. Banko, S. Dumais, A. Ng. *Data-intensive question answering.* In Proceedings of the Tenth Text REtrieval Conference, 2001.

[Brill02] Eric Brill, Susan Dumais, Michele Banko. *An analysis of the AskMSR question-answering system.* In Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing, 2002.

[Lin02] Jimmy Lin, Boris Katz. *Extracting Answers from the Web Using Knowledge Annotation and Knowledge Mining Techniques.* In Proceedings of the Eleventh Text REtrieval Conference, 2002.

[Lin03] Jimmy Lin, Boris Katz. *Question Answering Techniques for the World Wide Web.* MIT Artifical Intelligence Laboratory, 2003.

[Prager03] John M. Prager. *Question Answering Tutorial.* IBM T.J. Watson Research Center, 2003.

[RegTut] Sun Microsystems. *The Java Tutorial Tutorial → Trails Available Online Only → Regular Expressions.* (http://java.sun.com/docs/books/tutorial/extra/regex/)

[Voor99] Ellen M. Voorhees, Dawn M. Tice. *The TREC-8 Question Answering Track Evaluation.* In Proceedings of the Eighth Text REtrieval Conference, 1999.

[Voor04] Ellen M. Voorhees. *Overview of the TREC 2004 Question Answering Track.* In Proceedings of the Thirteenth Text REtrieval Conference, 2004.

[Zhang02] Dell Zhang, Wee Sun Lee. *Web based Pattern Mining and Matching Approach to Question Answering.* In Proceedings of the Eleventh Text REtrieval Conference, 2002.

[Zheng02] Zhiping Zheng. *AnswerBus Question Answering System.* University of Michigan, 2002.

Index