In Search of the Why

Developing a system for answering *why*-questions

Suzan Verberne

ISBN/EAN: 978-90-9025079-3
Cover photo: Gerard van der Laan, http://www.flickr.com/photos/one96five/,
Creative Commons — BY-NC-ND 2.0
Cover text: http://en.wikipedia.org/wiki/Flamingo, January 2010

In Search of the Why

Developing a system for answering *why*-questions

Een wetenschappelijke proeve op het gebied van de Letteren

Proefschrift

ter verkrijging van de graad van doctor
aan de Radboud Universiteit Nijmegen
op gezag van de rector magnificus prof. mr. S.C.J.J. Kortmann,
volgens besluit van het college van decanen,
in het openbaar te verdedigen op dinsdag 27 april 2010
om 15.30 uur precies

door

Suzan Verberne

geboren op 29 april 1980
te Nijmegen

Promotores:	Prof. dr. L. Boves
	Prof. dr. P.A. Coppen
Copromotor:	Dr. N. Oostdijk
Manuscriptcommissie:	Prof. dr. ir. W. Kraaij (Voorzitter)
	Prof. dr. A. van den Bosch (Universiteit van Tilburg)
	Prof. dr. M. de Rijke (Universiteit van Amsterdam)

Contents

Dankwoord (acknowledgements)

Een promovendus werkt nooit alleen. Tijdens de afgelopen jaren heb ik van verschillende mensen hulp gehad. Degenen die ik het meest dank ben verschuldigd zijn mijn promotor Lou Boves en mijn co-promotor Nelleke Oostdijk.

Lou heeft er in de eerste twee jaar van mijn promotieonderzoek voor gezorgd dat ik me bleef focussen. Zijn terugkerende vraag "wat is je doel?" is essentieel gebleken tijdens experimenteren en publiceren. In de laatste twee jaar bleek dat Lou de beste reviewer is die een promovendus zich kan wensen. Bovendien wil ik hem heel hartelijk bedanken voor zijn onvermoeibare inspanningen in de laatste paar maanden voordat mijn proefschrift naar de drukker ging.

Nelleke is de auteur van het oorspronkelijke onderzoeksvoorstel "In Search of the Why". Ik vermoed dat ik het aan haar te danken heb, dat ik de baan gekregen heb. Nelleke heeft mij gedurende het hele project het gevoel gegeven volledig achter mij te staan, hoewel ik met de uitwerking van mijn proefschrift niet erg dicht bij haar oorspronkelijke onderzoeksvoorstel ben gebleven. Voor haar niet aflatende steun ben ik haar dankbaar.

Mijn derde begeleider, Peter-Arno Coppen, is de reden geweest dat ik meer dan 10 jaar geleden Taal, Spraak & Informatica ben gaan studeren. Hij is een fantastische docent en ik hoop dat mijn hoorcolleges ooit net zo goed worden als die van hem. Doordat hij van baan en werkplek is veranderd, is zijn rol als begeleider in mijn onderzoeksproject langzaamaan kleiner geworden. Ik wil hem toch bedanken voor de inspiratie die ik van hem heb gekregen.

Mijn collega Daphne Theijssen heeft een bijzonder grote rol gespeeld in de voltooiing van mijn proefschrift. Ze heeft in 2006 en 2007 stage gelopen op mijn project en toen ze zelf promovendus werd, werden we kamergenoten. Als ik een greep doe uit de dingen waar ze mee geholpen heeft dan levert dat een bonte verzameling op: het begrijpen van SVM's, het hacken in de LaTeX-stylesheet voor de CLIN-bundel, het formuleren van de perfecte tactische email aan een lastige co-auteur, het vinden van fouten in mijn artikelen en het bijkomen van dat alles onder het genot van radio 3 en thee in 28 smaken.

Ook van andere collega's heb ik de afgelopen jaren op allerlei verschillende manieren hulp gekregen. In het bijzonder zijn dat Eric, Louis, Hans, Folkert, Bert en

Hella, ieder op zijn/haar eigen manier. Daarvoor wil ik hen, maar ook mijn andere collega's, heel erg bedanken.

In 2008 heb ik acht maanden lang twee dagen per week op de Universiteit van Amsterdam (UvA) gewerkt. Maarten de Rijke en Valentin Jijkoun wil ik bedanken voor de kans die ik kreeg om bij ILPS te werken. Toen ik voor een seminar op de UvA was, kon ik per direct beginnen en ik heb er heel veel geleerd. Mijn ILPS-collega's wil ik bedanken voor de fijne tijd die ik in Amsterdam gehad heb, waarbij de concerten in Paradiso en onze 'trip to (waterslide) paradise' de hoogtepunten waren.

Ik heb een aantal conferenties bezocht waar ik veel geleerd heb van andere onderzoekers. Daarom wil ik de mensen bedanken van wie ik waardevolle en sympathieke input heb gekregen tijdens EACL 2006, SIGIR 2007, ECIR 2008, COLING 2008 en SIGIR 2009. Naast die internationale conferenties zijn de Nederlandse netwerken van CLIN en DIR ook heel belangrijk gebleken. Mijn vakgenoten uit Tilburg, Enschede, Groningen, Amsterdam en Delft ben ik daarom ook dank verschuldigd.

Eva, Jeroen, Robin, Koen, Liesbeth en Annemieke hebben geheel belangeloos delen van mijn proefschrift gecontroleerd op tekstuele en opmaak-fouten. Gerdieneke en Esther hebben mij geadviseerd over de opmaak van de kaft. Bedankt daarvoor!

Ik wil mijn ouders graag bedanken omdat ze mij een opvoeding hebben gegeven waarin zelfstandigheid en nieuwsgierigheid centraal stonden; twee eigenschappen die onmisbaar zijn voor het schrijven van een proefschrift. Mijn vrienden bedank ik voor hun interesse in mijn werk en de vele etentjes.

Tot slot wil ik Paul heel hartelijk bedanken voor zijn onvoorwaardelijke steun. Zonder hem had ik vele avonden in de Refter doorgebracht en vele weekenden achter mijn computer. Toen de deadline van mijn manuscript naderde, heeft hij ervoor gezorgd dat ik juist veel uren kon maken op mijn proefschrift. Bovendien heeft hij me gestimuleerd om een paar maanden in het buitenland te gaan werken en daar ben ik hem heel dankbaar voor.

1

Introduction

Information searching was once a task performed almost exclusively by librarians and domain experts. With the rise of the Internet, searching for information by means of retrieval engines has become a daily activity for many people. The most widespread form of web searching is ad-hoc document retrieval: The user types in a query in the form of a (usually small) unstructured set of key words, and the search engine retrieves a list of pointers to web pages. The web pages are ranked according to their estimated relevance to the input query. Ad-hoc document retrieval and ranking is the core business of the large web search engines Google[1], Yahoo![2] and Bing[3].

The value of ad-hoc document retrieval for the user depends on his/her information need. If the user is searching for general information on a topic then it is useful to retrieve a set of documents on this topic. For example, the query "flamingos" can be interpreted as "I would like to get information about flamingos" and the web page that is ranked first by Google, Yahoo! and Bing (`http://en.wikipedia.org/wiki/Flamingo`) will probably satisfy this information need. However, if the information need is more specific than that, a complete web document (albeit helpful) is often not directly of use to the searcher. For example, the query "flamingos food chain" (taken from Google's history of frequent searches for the query offset 'flamingos') requires scrolling through at least one of the retrieved documents in search of the answer to the user's information need. It is fair to say that in many cases where users need more specific information than what is easily expressed in

[1] See `www.google.com`

[2] See `www.yahoo.com`

[3] See `www.bing.com`

a few keywords, ad-hoc document retrieval is not focused enough. For increasingly specific query types, the task of the retrieval engine moves from document retrieval towards information extraction.

Questions in natural language that start with an interrogative pronoun or adverb (*who*, *what*, *which*, *where*, *when*, *why*, or *how*) tend to be even more specific than the example query "flamingos food chain". Not only is the user's information need much better expressed by a natural language question than by the set of keywords typically used in ad-hoc retrieval, the unit of retrieval is also smaller and can be pointed out more specifically than the retrieval unit for ad-hoc queries. For example, the question "where do flamingos nest?" (again taken from Google's search history) describes the searcher's information need very clearly and expects a clearly defined answer: a location.

The problem of automatically answering natural language questions by pinpointing the exact answer in a large text (web) corpus has been studied since the mid 1990s (see Section 1.2 below). Most research has been directed at answering so-called 'factoid' questions: questions that expect a short, clearly identifiable answer; usually a named entity such as a person name, location or year. Answering natural language questions is one of the best defined information retrieval tasks. In this thesis we specifically focus on the problem of answering *why*-questions.

Why-questions require a different approach than factoid questions because their answers tend to be longer and more complex. Moreover, because of the complexity of the problem, *why*-question answering (*why*-QA) is an interesting case study for a linguistically motivated approach to information retrieval. In most cases, *why*-questions take on the form of complete sentences (e.g. "Why are flamingos pink?"). Consequently, they contain structural (syntactic and semantic) information on the underlying information need (see Section 1.4). Answers to *why*-questions involve reasoning in natural language, which makes analyses on the semantic and rhetorical levels interesting options.

This thesis is positioned halfway between knowledge-based and data-driven approaches to natural language processing. Throughout the thesis, the first step after formulating a research question is the analysis of the data at hand. This analysis then leads to the formulation of a hypothesis. In order to test our hypotheses, we design experiments in which we implement structured linguistic knowledge.

1.1 Main Research objective

In this thesis, we aim at developing an approach for answering *why*-questions. We assume that structured linguistic knowledge can play an important role in the imple-

mentation of this approach. We will investigate which levels of linguistic information (lexico-semantic, syntactic, discourse) are the most informative.

We restrict ourselves to questions in English. We do not focus on a specific topic domain, but we aim at developing an approach to open domain *why*-question answering: the type of *why*-questions that are asked to general, online QA systems. Furthermore, we only consider questions that start with the question word *why* and are complete, grammatically well-formed sentences.

1.2 Background and related work in QA

Question Answering (QA) research emerged in the 1960s with the development of natural language interfaces to databases containing specialized information about a topic [86]. In the 1970s, QA research was mainly aimed at the development of intelligent systems that used knowledge bases and rule-based reasoning for the understanding of stories in natural language [83]. In the 1980s and 1990s, these knowledge-intensive approaches reached their limits because they could not easily be generalized to open domain text understanding [25].

Research into open domain QA then emerged in the field of information retrieval (IR) in the mid-1990s, boosted by the NIST evaluation campaigns TREC[4] (for English QA) and CLEF[5] (for multi-lingual QA). One of the benefits of these evaluation campaigns is the use of common evaluation measures. In the first few editions of TREC, Mean Reciprocal Rank (MRR) was used as the main evaluation criterion. MRR is based on the rank of the highest ranked correct answer for each question. It is calculated as follows: for each question, the reciprocal rank (RR) is 1 divided by the rank of the highest ranked correct answer ($RR = 0$ if no correct answer is retrieved). MRR is the mean RR over all questions.

At the first edition of TREC-QA (TREC-8 in 1999), the best QA system already reached an MRR of 0.66 on a test collection mainly consisting of factoid questions. All successful systems in TREC-8 used standard text retrieval techniques for retrieving passages of text that share content with the question. On these passages, named entity recognition was applied for extracting entities of the type asked for in the question (e.g., a person for *who*, a time designation for *when*). The best systems were able to answer two out of three factoid questions correctly, which was considered very successful [117].

In the years following TREC-8, more difficult question types (e.g., definition

[4] See `http://trec.nist.gov/`

[5] See `http://www.clef-campaign.org/`

and list-type questions) and more realistic questions were added to the test collection [110]. Some developers tried to include knowledge about question structure in their system, but from 2001 onward a steadily increasing preference for a shallow approach based on standard information retrieval techniques can be observed [112]. This approach to answering factoid questions remained unchanged over the years [114, 113, 115, 116], and although the difficulty of the task increased, the overall system performance for the task did not suffer and was in fact found to be quite stable.

A number of attempts were made to improve the retrieval-based QA systems with linguistic knowledge. Most of these experiments focus on adding syntactic information to baseline systems that represent both the query and the response documents as bags of words. Systems using NLP techniques generally show a small but significant improvement over the bag-of-words baseline [80, 88, 92]. We will come back to this in Chapter 4.

As pointed out above, most attention in QA research has been paid to factoid questions. Work addressing non-factoids such as questions starting with *how* and *why* has largely been limited to subtasks of systems that try to answer all types of questions within one framework [65, 38]. The fact that non-factoid questions have been given little attention cannot be attributed fully to their lower relative frequency in a QA context. Microsoft's Web Search Click Data, a collection of queries from US users entered into the Microsoft Live search engine in the summer of 2006, contains 86,391 queries starting with *who*, *what*, *which*, *where*, *when*, *how* or *why*. Of these, queries starting with *how* are the most frequent by far (58%).[6] *Why*-questions are less frequent: 3% of the *wh*-questions in the MSN click data start with *why*. The Webclopedia data collection, which was crawled from the online QA engine `answers.com` by Hovy et al. [39], contains 17,000 questions, 4.7% of which are *why*-questions.

Although *why*-questions are not the most frequent type of questions that users ask online information systems (see above), they are frequent enough to be worth investigating. Moreover, the problem of answering *why*-questions is challenging since the QA systems that use one approach for all types of *wh*-questions fail to answer *why*-questions [50, 78, 62].

[6] We must note here that questions starting with *how* are very diffuse in their expected answer type. A large proportion of the questions starting with *how* are *how to*-questions ("how to get rid of ants in the bathroom", 76%) and quantity-questions (*how much*, *how many*, *how long*, etc, approx. 10%), which are very different from manner-type questions ("how does dopamine effect behavior?", approx. 7%). A small proportion of *how*-questions are closely related to *why*-questions, because they expect an explanation as answer ("how does hyperthyroidism lead to osteoporosis").

1.3 Data

1.3.1 Data for why-QA

As explained in the introduction, the research in this thesis is an exercise in resource-based language analysis. Therefore, it is vital to have a proper data collection of *why*-questions and their answers, as well as a widely accepted procedure for designing and conducting experiments. Using an existing data collection is to be preferred over creating a new data set because re-using existing data sets saves time and makes it possible to compare results to findings of other researchers who used the same data. We formulated a number of requirements that a data set must meet in order to be appropriate for research and development of an approach to *why*-QA.

The first requirement for an appropriate data set concerns the form of the questions. In the context of the current research, a *why*-question is defined as an interrogative sentence in which the interrogative adverb *why* occurs in initial position. Secondly, we only consider the subset of *why*-questions that might be posed in a QA context and for which the answer can be expected to be present in some related document set. This means that our data set should only comprise *why*-questions for which the answer can be found in a fixed collection of documents. Thirdly, the data set should not only contain questions, but also the corresponding answers and source documents. A fourth requirement is that the size of the data set should be large and rich enough so that it is reasonable to expect that it covers the variation that occurs in *why*-questions in a QA context. Finally, we do not want the question–answer pairs to be domain-specific since we aim at developing an open domain system.

We considered several sources of *why*-questions. We looked into TREC data, lists of frequently asked questions (FAQs), questions from web search logs and questions elicited to an existing corpus. Since the TREC-QA data only contains a few *why*-questions each year, this set was too small to use in our experiments. FAQ data can easily be crawled from the web in large portions (e.g. [43]) and contains both questions and answers, but these questions appeared to be quite domain specific, often related to one specific web site or product. Questions from web search logs are open domain and representative for user-generated questions but their answers are not available with the queries.[7] Elicited questions to a predefined set of documents are not representative for web questions because they tend to be very closely related to the topic of a specific document, but their answers are available, and potentially annotated (if we choose an annotated corpus for the elicitation).

[7] Microsoft's Web Search Click Data set does contain the documents that were clicked on by the users of the engine, but these data were only made available to researchers in 2009.

After these considerations we decided to work with two different types of data. First, we collected two sets of *why*-questions that we elicited from native speakers who read a number of newspaper texts: unannotated texts from Reuters and Guardian (see Chapter 2), and Wall Street Journal articles that had been annotated on the syntactic and discourse level (the RST Treebank [18], see Chapter 3). The second type of data (which is described in Chapter 4) comprises the set of *why*-questions from the Webclopedia data collection (crawled from `answers.com`) [39]. For these questions, we manually extracted the answers from the Wikipedia XML 2006 corpus [24]. The first type of data is used in Chapter 2 and 3, the second in Chapter 4, 5 and 6.

1.3.2 External resources

For the implementation of our experiments, we often need external resources that provide us with (structured) linguistic information. In this section, we briefly describe the most important external resources that we exploit in this thesis. Note that we do not aim at developing or improving these resources, but we apply them in our experiments as they are.

- For the implementation of experiments that require term matching or word overlap counts, lemmatization or stemming of words is often needed. We decided to perform lexicon-based lemmatization using the CELEX lemma lexicon [3]. From the English CELEX lexicon, we extracted a list of all lexical entries and their lemma. The task of lemmatization is then reduced to a table look-up task for each word form. We found that this is a robust method for lemmatization of English words.

- For retrieval and word overlap experiments, it is sometimes helpful to filter out stop words (function words and other content-poor words) [30]. To this end, we use a stop word list downloaded from `http://marlodge.supanet.com/museum/funcword.html`, from which we removed the numerals and the word *why*.

- We use the lexico-semantic information contained in the WordNet synonym sets [27] in a number of our experiments in order to find synonyms or hypernyms of words. We use the WordNet Similarity Tool [76] as a more general way of determining similarity between words that is not limited to relations between words of the same word class. We extracted semantic verb classes from the Levin Verb index [52] for one of our experiments.

- We use a number of existing text corpora. For our first experiments with elicited *why*-questions, we use newspaper texts from Reuters' *Textline Global News* (1989) and *The Guardian on CD-ROM* (1992). For our experiments with discourse-annotated texts, we exploit the RST Discourse Treebank [18]. The Wikipedia XML 2006 corpus [24] is used as answer corpus for the encyclopaedia-type questions from the Webclopedia question set [39].

- Since we experiment with the contribution of syntactic information for answering *why*-questions, we rely on syntactic parsers for some of our experiments. For parsing large amounts of data we use the well-known dependency parser built by Charniak [21], which is fast but not very precise. For question analysis tasks, where the precision of constituent extraction is important, we use TOSCA's syntactic parser [71] and its successor Pelican.[8]

1.4 Characteristics of why-questions

1.4.1 The syntax of why-questions

As introduced in Section 1.1, we decided to exclude syntactically incomplete sentences such as "why a Bachelor's degree?" from our data collection. As a result, all questions that we work with are complete interrogative sentences. We found that *why*-questions have a relatively strict word order. In Chapter 2, we define the default word order for questions that start with *why* as:

```
WHY OPERATOR SUBJECT PREDICATE,
```

in which optional adverbials can be inserted. Within this frame, the `OPERATOR` is realized by an auxiliary or a form of *be* or *have*, optionally followed by a negator. `SUBJECT` and `PREDICATE` comprise the same constituents as in declarative sentences. Both the subject and the predicate can potentially be complex, but in most *why*-questions, the subject is relatively short. In examples (1) and (2) below, the subjects are semantically poor (*people*, *you*) since they do not refer to specific entities. This means that in these questions, the predicate is the only content-bearing part of the question.

(1) Why do people get more colds in the winter?

(2) Why do you get a headache?

[8] See `http://lands.let.ru.nl/projects/pelican/`

In examples (3), (4) and (5) below, both the subject and the predicate are semantically rich.

(3) Why are Hush Puppies called Hush Puppies?

(4) Why is the sky red at sunset and also colorful at sunrise?

(5) Why did B.B. King name his guitar 'Lucille'?

1.4.2 The semantics of why-questions

In QA system development, the notion of answer type is very important: Predicting the type of entity that is expected as an answer facilitates answer identification. For example, it is to a high extent possible to recognize person names, dates and locations in a text. In most QA systems, the answer type is predicted directly from the question word: *who* leads to 'person', *when* to 'date' and *where* to 'location'. In the case of *why*-questions, the expected answer type is 'explanation' (or 'reason', according to Moldovan et al. [65]). In Chapter 2, we work out a distinction into subtypes of the general answer type for *why*-questions. We found that the most frequent answer types for *why*-questions are cause and motivation. Of the example questions above, (1), (2) and (4) are causal questions, while (5) is a motivation-type question. Question (3) has the answer type 'etymology'. We come back to the answer types for *why*-questions in chapters 2 and 4.

1.4.3 The semantics of answers to why-questions

Prager et al. [78] define twenty different answer patterns for *wh*-questions but they do not create such a pattern for *why*-questions. Instead, they recognize explanations by cue phrases such as *because* and *in order to*. We found however that many answers to *why*-questions do not contain such explicit cues [102]. In many cases, a word or phrase in the answer can be pointed out as a cue for explanation, but these phrases occur often in context in which they do not relate to some kind of explanation. At the same time it holds that not all explanations are marked by cue phrases. Consider for example the answers to questions (1) and (5):

(1) Colds are somewhat more common in winter since during that time of the year people spend more time indoors in close proximity to others, and ventilation is less efficient, increasing the infection risk.

> (5) In the winter of 1949, King played at a dance hall in Twist, Arkansas. In order to heat the hall, a barrel half-filled with kerosene was lit, a common practice. During a performance, two men began to fight, knocking over the burning barrel and sending burning fuel across the floor. This triggered an evacuation. Once outside, King realized that he had left his guitar inside the burning building. He entered the blaze to retrieve his guitar, a Gibson acoustic. Two people died in the fire. The next day, King discovered that the two men were fighting over a woman named Lucille. King named that first guitar Lucille, as well as every one he owned since that near-fatal experience, "to remind me never to do a thing like that again."

The answer to question (1) contains the cue word *since*, which is used directly after a clause that rephrases the question ("Colds are somewhat more common in winter"). This example can be seen as a classic instance, for which system developers design their answer patterns. The answer to question (5) on the other hand contains a line of reasoning in which it is not possible to point out a clear cue or an explanation. The last sentence contains a syntactic cue in the form of the adverbial infinitive clause *to remind me never to do a thing like that again*. However, infinitive clauses can have several different non-explanation meanings making it an ambiguous cue. Moreover, *to remind me never to do a thing like that again* is not the complete answer to the question.

1.5 Research questions and thesis outline

Based on our research objective (see Section 1.1), we formulate the following main question to be answered in this thesis:

Main Question What are the possibilities and limitations of an approach to *why*-QA that uses linguistic information in addition to text retrieval techniques?

We formulate five subquestions for this research question, which will be addressed in Chapters 2 to 6 respectively.

In Chapter 2 and 3 we investigate the role that linguistic information can play in answering *why*-questions.

RQ I. What types of linguistic information can play a role in question analysis for *why*-QA?

In Chapter 2, we define four semantic answer types of *why*-questions. We hypothesize that information on the syntactic structure of the question gives valuable information on the type of answer that is expected and we develop an approach for predicting this answer type.

RQ II. To what extent can annotations on the level of rhetorical structure (discourse analysis) help in extracting answers to *why*-questions?

In Chapter 3 we investigate the value of annotations on the discourse level for the extraction of answers. We implement an answer extraction module that uses manual annotations of the RST Treebank for extracting answer passages from the documents. Although the results from our experiments described in Chapter 3 are promising, our proposed method relies on annotated data. Therefore, we shift our focus in the second half of this thesis to an approach for which shallow text processing techniques suffice.

In Chapters 4 and 5, we describe our attempts at building a system for *why*-QA that combines off-the-shelf text retrieval technology with linguistic and other structural knowledge of *why*-questions and their answers. Our system retrieves and ranks a set of candidate answers per question using the Lemur retrieval engine.[9] Then it re-ranks the answers on the basis of a set of linguistically motivated features that describe the relative overlap between the question and each of its candidate answers.

RQ III. To what extent can we improve answer ranking by adding structural linguistic information to a passage retrieval module for *why*-QA and which information from the question and its candidate answers is the most important?

In Chapter 4, we compile the feature set that covers this information and evaluate it using a set of *why*-questions from the Webclopedia question set and the Wikipedia XML corpus.

RQ IV. Which machine learning techniques are the most suitable for learning the ranking of *why*-answers, using a set of linguistically motivated overlap features?

In Chapter 5, we optimize the ranking component of our system by evaluating a number of machine learning techniques for the task of learning to rank the answers to *why*-questions.
In Chapter 6, we review the problem of *why*-QA in detail:

[9] See `http://www.lemurproject.org/`

RQ V. What are the limitations of an IR approach to *why*-QA and to what extent can models of human text understanding explain the shortcomings of state-of-the art QA systems?

We analyze the subset of *why*-questions from Webclopedia for which we were not able to find the answer in Wikipedia manually. We also make a profound analysis of the questions for which we have manually found an answer in the corpus while our system fails to retrieve it. From this analysis, we identify the limitations of an IR-based approach to *why*-QA and we relate these limitations to the human capability of recognizing answers to *why*-questions.
Chapter 7 concludes this thesis with a summary of Chapters 2 to 6 and answers to the research questions posed in this chapter.

2

Data and Question Analysis for Why-QA

Edited from: S. Verberne. Developing an Approach for Why-Question Answering. In: *Conference Companion of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2006)*, pages 39-46, Trento, Italy, 2006. Association for Computational Linguistics.

Abstract

In this chapter, we describe a data collection of *why*-questions that we created as a first step in the development and evaluation of a method for automatically answering *why*-questions (*why*-QA). The resulting collection comprises 395 *why*-questions. For each question, the source document and one or two user-formulated answers are available in the data set. With the collected data, we developed a question analysis method for *why*-questions, based on syntactic categorization and answer type determination. The quality of the output of this module is promising for future development of our method for *why*-QA.

2.1 Introduction

Until now, research in the field of automatic question answering (QA) has focused on factoid (closed-class) questions like *who*, *what*, *where* and *when* questions. Results reported for the QA track of the Text Retrieval Conference (TREC) show that these types of *wh*-questions can be handled rather successfully [113].

In the current thesis, we aim at developing an approach for automatically answering *why*-questions. So far, *why*-questions have largely been ignored by researchers in the QA field. One reason for this is that the frequency of *why*-questions in a QA context is lower than that of other questions like *who*- and *what*-questions [39]. However, although *why*-questions are less frequent than some types of factoids (*who*, *what* and *where*), their frequency is not negligible: in a QA context, they comprise about 5 percent of all *wh*-questions [38, 43] and they do have relevance in QA applications [62]. A second reason for ignoring *why*-questions until now, is that it has been suggested that the techniques that have proven to be successful in QA for closed-class questions are not suitable for questions that expect a procedural answer rather than a noun phrase [50]. This chapter aims to find out whether the suggestion is true that factoid-QA techniques are not suitable for *why*-QA. We want to investigate whether principled syntactic parsing can make QA for *why*-questions feasible.

In this chapter, we report on the work that has been carried out until now. More specifically, Sections 2.2 and 2.3 describe the approach taken to data collection and question analysis and the results that were obtained. Then, in Section 2.4, we discuss the plans and goals for the work that will be carried out in the remainder of the thesis.

2.2 Data for why-QA

In research in the field of QA, data sources of questions and answers play an important role. Appropriate data collections are necessary for the development and evaluation of QA systems [117]. While in the context of the QA track of TREC data collections in support of factoid questions have been created, so far, no resources have been created for *why*-QA. For the purpose of the present research therefore, we have developed a data collection comprising a set of questions and corresponding answers. In doing so, we have extended the time tested procedures previously developed in the TREC context.

In this section, we describe the requirements that a data set must meet to be appropriate for system development and we discuss a number of existing sources of *why*-questions. Then we describe the method employed for data collection and the

main characteristics of the resulting data set.

The first requirement for an appropriate data set concerns the nature of the questions. In the context of the current research, a *why*-question is defined as an interrogative sentence in which the interrogative adverb *why* (or one of its synonyms) occurs in (near) initial position. We consider the subset of *why*-questions that could be posed in a QA context and for which the answer is known to be present in the related document set. This means that the data set should only comprise *why*-questions for which the answer can be found in a fixed collection of documents. Secondly, the data set should not only contain questions, but also the corresponding answers and source documents. The answer to a *why*-question is a clause or sentence (or a small number of coherent sentences) that answers the question without giving supplementary context. The answer may not always be literally present in the source document, but can be deduced from it. For example, a possible answer to the question "Why are 4300 additional teachers required?", based on the source snippet "The school population is due to rise by 74,000, which would require recruitment of an additional 4,300 teachers", is "Because the school population is due to rise by a further 74,000."

Finally, the size of the data set should be large enough to cover all relevant variation that occur in *why*-questions in a QA context.

There are a number of existing sources of *why*-questions that we may consider for use in our research. However, for various reasons, none of these appear suitable.

Why-questions from corpora like the British National Corpus [15], in which questions typically occur in spoken dialogues, are not suitable because the answers are not structurally available with the questions, or they are not extractable from a document that has been linked to the question. The same holds for the data collected for the Webclopedia project [39], in which neither the answers nor the source documents were included. One could also consider questions and answers from frequently asked questions (FAQ) pages, like the large data set collected by Valentin Jijkoun [43]. However, in FAQ lists, there is no clear distinction between the answer itself (a clause that answers the question) and the source document that contains the answer.

The questions in the test collections from the TREC-QA track do contain links to the possible answers and the corresponding source documents. However, these collections contain too few *why*-questions to qualify as a data set that is appropriate for developing *why*-QA.

Given the lack of available data that match our requirements, a new data set for QA research into *why*-questions had to be compiled. In order to meet the given requirements, it would be best to collect questions posed in an operational QA environment, like the compilers of the TREC-QA test collections did: they extracted fac-

toid and definition questions from search logs donated by Microsoft and AOL [113]. Since we do not have access to comparable sources, we decided to revert to the procedure used in earlier TRECs, and imitate a QA environment in an elicitation experiment. We extended the conventional procedure by collecting user-formulated answers in order to investigate the range of possible answers to each question. We also added paraphrases of collected questions in order to extend the syntactic and lexical variation in the data collection.

In the elicitation experiment, ten native speakers of English were asked to read five texts from Reuters' *Textline Global News* (1989) and five texts from *The Guardian on CD-ROM* (1992). The texts were around 500 words each. The experiment was conducted over the Internet, using a web form and some CGI scripts. In order to have good control over the experiment, we registered all participants and gave them a code for logging in on the web site. Every time a participant logged in, the first upcoming text that he or she did not yet finish was presented. The participant was asked to formulate one to six *why*-questions for this text, and to formulate an answer to each of these questions. The participants were explicitly told that it was essential that the answers to their questions could be found in the text. After submitting the form, the participant was presented the questions posed by one of the other participants and he or she was asked to formulate an answer to these questions too. The collected data was saved in text format, grouped per participant and per source document, so that the source information is available for each question. The answers have been linked to the questions.

In this experiment, 395 questions and 769 corresponding answers were collected. The number of answers would have been twice the number of questions if all participants would have been able to answer all questions that were posed by another participant. However, for 21 questions (5.3%), the second participant was not able to answer the first participant's question. Note that not every question in the elicitation data set has a unique topic:[1] on average, 38 questions were formulated per text, covering around twenty topics per text.

The collected questions have been formulated by people who had constant access to the source text. As a result of that, the chosen formulations often resemble the original text, both in the use of vocabulary and sentence structure. In order to expand the dataset, a second elicitation experiment was set up, in which five participants from the first experiment were asked to paraphrase some of the original *why*-questions. The 166 unique questions were randomly selected from the original data set. The participants formulated 211 paraphrases in total for these questions.

[1]The topic of a *why*-question is the proposition that is questioned. A *why*-question has the form 'WHY P', in which P is the topic [97].

This means that some questions have more than one paraphrase. The paraphrases were saved in a text file that includes the corresponding original questions and the corresponding source documents.

We studied the types of variation that occur among questions covering the same topic. First, we collected the types of variation that occur in the original data set and then we compared these to the variation types that occur in the set of paraphrases.

In the original data set, the following types of variation occur between different questions on the same topic:

- Lexical variation, e.g. *for the second year running* vs. *again*;
- Verb tense variation, e.g. *have risen* vs. *have been rising*;
- Optional constituents variation, e.g. *class sizes* vs. *class sizes in England and Wales*;
- Sentence structure variation, e.g. *would require recruitment* vs. *need to be recruited*

In the set of paraphrases, the same types of variation occur, but as expected the differences between the paraphrases and the source sentences are slightly bigger than the differences between the original questions and the source sentences. We measured the lexical overlap between the questions and the source texts as the number of content words that occur in both the question and the source text. The average relative lexical overlap (the number of overlapping words divided by the total number of words in the question) between original questions and source text is 0.35; the average relative lexical overlap between paraphrases and source text is 0.31.

The size of the resulting collection (395 original questions, 769 answers, and 211 paraphrases of questions) is large enough to initiate serious research into the development of *why*-QA.

Our collection meets the requirements that were formulated with regard to the nature of the questions and the presence of the answers and source documents for every question.

2.3 Question analysis for why-QA

The goal of question analysis is to create a representation of the user's information need. The result of question analysis is a query that contains all information about the answer that can be extracted from the question. So far, no question analysis procedures have been created for *why*-QA specifically. Therefore, we have developed

an approach for proper analysis of *why*-questions. Our approach is based on existing methods of analysis of factoid questions. This will allow us to verify whether methods used in handling factoid questions are suitable for use with explanation-type questions. In this section, we describe the components of successful methods for the analysis of factoid questions. Then we present the method that we used for the analysis of *why*-questions and indicate the quality of our method.

The first (and most simple) component in current methods for question analysis is keyword extraction. Lexical items in the question give information on the topic of the user's information need. In keyword selection, several different approaches may be followed. Moldovan et al. [65], for instance, select as keywords all named entities that were recognized as proper nouns. In almost all approaches to keyword extraction, syntax plays a role. Shallow parsing is used for extracting noun phrases, which are considered to be relevant key phrases in the retrieval step. Based on the query's keywords, one or more documents or paragraphs can be retrieved that may possibly contain the answer.

A second, very important, component in question analysis is determination of the question's semantic answer type. The answer type of a question defines the type of answer that the system should look for. Often-cited work on question analysis has been done by Moldovan et al. [66, 65], Hovy et al. [38], and Ferret et al. [28]. They all describe question analysis methods that classify questions with respect to their answer type. In their systems for factoid-QA, the answer type is generally deduced directly from the question word: *who* leads to the answer type *person*; *where* leads to the answer type *place*, etc. This information helps the system in the search for candidate answers to the question. Hovy et al. find that, of the question analysis components used by their system, the determination of the semantic answer type makes by far the largest contribution to the performance of the entire QA system.

For determining the answer type, syntactic analysis may play a role. When implementing a syntactic analysis module in a working QA system, the analysis has to be performed fully automatically. This may lead to concessions with regard to either the degree of detail or the quality of the analysis. Ferret et al. implement a syntactic analysis component based on shallow parsing. Their syntactic analysis module yields a syntactic category for each input question. In their system, a syntactic category is a specific syntactic pattern, such as 'WhatDoNP' (e.g. "What does a defibrillator do?") or 'WhenBePNborn' (e.g. "When was Rosa Park born?"). They define 80 syntactic categories like these. Each input question is parsed by a shallow parser and hand-written rules are applied for determining the syntactic category. Ferret et al. find that the syntactic pattern helps in determining the semantic answer type (e.g. *company*, *person*, *date*). They unfortunately do not describe how they created

the mapping between syntactic categories and answer types.

As explained above, determination of the semantic answer type is the most important task of existing question analysis methods. Therefore, the goal of our question analysis method is to predict the answer type of *why*-questions.

In the work of Moldovan et al. [65], all *why*-questions share the single answer type *reason*. However, we believe that it is necessary to split this answer type into subtypes, because a more specific answer type helps the system select potential answer sentences or paragraphs. The idea behind this is that every subtype has its own lexical and syntactic cues in a source text.

Based on the classification of adverbial clauses by Quirk [81] (section 15.45), we distinguish the following subtypes of *reason*: cause, motivation, circumstance (which combines reason with conditionality), and purpose.

Below, an example of each of these answer types is given.

- Cause: "The flowers got dry because it hadn't rained in a month."
- Motivation: "I water the flowers because I don't like to see them dry."
- Circumstance: "Seeing that it is only three, we should be able to finish this today."
- Purpose: "People have eyebrows to prevent sweat running into their eyes."

The *why*-questions that correspond to the reason clauses above are respectively "Why did the flowers get dry?", "Why do you water the flowers?", "Why should we be able to finish this today?", and "Why do people have eyebrows?". It is not always possible to assign one of the four answer subtypes to a *why*-question. We will come back to this later.

Often, the question gives information on the expected answer type. For example, compare the two questions below:

1. Why did McDonald's write Mr. Bocuse a letter?
2. Why have class sizes risen?

Someone asking question 1. expects as an answer McDonald's motivation for writing a letter, whereas someone asking question 2. expects the cause for rising class sizes as answer. The corresponding answer paragraphs do indeed contain the equivalent answer subtypes:

> "McDonald's has acknowledged that a serious mistake was made. 'We have written to apologise and we hope to reach a settlement with Mr. Bocuse this week,' said Marie-Pierre Lahaye, a spokeswoman for McDonald's France, which operates 193 restaurants."

> "Class sizes in schools in England and Wales have risen for the second year running, according to figures released today by the Council of Local Education Authorities. The figures indicate that although the number of pupils in schools has risen in the last year by more than 46,000, the number of teachers fell by 3,600."

We aim at creating a question analysis module that is able to predict the expected answer type of an input question. In the analysis of factoid questions, the question word often gives the needed information on the expected answer type. In case of *why*, the question word does not give information on the answer type since all *why*-questions have *why* as question word. This means that other information from the question is needed for determining the answer subtype.

We decided to use Ferret's approach, in which syntactic categorization helps in determining the expected answer type. In our question analysis module, the TOSCA (TOols for Syntactic Corpus Analysis) system [71] is explored for syntactic analysis. TOSCA's syntactic parser takes a sequence of unambiguously tagged words and assigns function and category information to all constituents in the sentence. The parser yields one or more possible output trees for (almost) all input sentences. For the purpose of evaluating the maximum contribution to a classification method that can be obtained from a principled syntactic analysis, the most plausible parse tree from the parser's output is selected manually.

For the next step of question analysis, we created a set of hand-written rules, which are applied to the parse tree in order to choose the question's syntactic category. We defined six syntactic categories for this purpose:

- Action questions, e.g. "Why did McDonald's write Mr. Bocuse a letter?"
- Process questions, e.g. "Why has Dixville grown famous since 1964?"
- Intensive complementation questions, e.g. "Why is Microsoft Windows a success?"
- Monotransitive *have* questions, e.g. "Why did compilers of the OED have an easier time?"
- Existential *there* questions, e.g. "Why is there a debate about class sizes?"

- Declarative layer questions, e.g. "Why does McDonald's spokeswoman think the mistake was made?"

The choice for these categories is based the information that is available from the parser, and the information that is needed for determining the answer type.

For some categories, the question analysis module only needs fairly simple cues for choosing a category. For example, a main verb with the feature *intens* leads to the category 'intensive complementation question' and the presence of the word *there* with the syntactic category EXT leads to the category 'existential *there* question'. For deciding on declarative layer questions, action questions and process questions, complementary lexical-semantic information is needed. In order to decide whether the question contains a declarative layer, the module checks whether the main verb is in a list that corresponds to the union of the verb classes *say* and *declare* from Verbnet [48], and whether it has a clausal object. The distinction between action and process questions is made by looking up the main verb in a list of process verbs. This list contains the 529 verbs from the causative/inchoative alternation class (verbs like *melt* and *grow*) from the Levin verb index [52]; in an intransitive context, these verbs are process verbs. We have not yet developed an approach for passive questions.

Based on the syntactic category, the question analysis module tries to determine the answer type. Some of the syntactic categories lead to an answer type directly. All process questions with non-agentive subjects get the expected answer type *cause*. All action questions with agentive subjects get the answer type *motivation*. We extracted information on agentive and non-agentive nouns from WordNet: all nouns that are in the lexicographer file `noun.person` were selected as agentive.

Other syntactic categories need further analysis. Questions with a declarative layer, for example, are ambiguous. The question "Why did they say that migration occurs?" can be interpreted in two ways: "Why did they say it?" or "Why does migration occur?". Before deciding on the answer type, our question analysis module tries to find out which of these two questions is supposed to be answered. In other words: the module decides which of the clauses has the question focus. This decision is made on the basis of the semantics of the declarative verb. If the declarative is a factive verb - a verb that presupposes the truth of its complements - like *know*, the module decides that the main clause has the focus. The question consequently gets the answer type *motivation*. In case of a non-factive verb like *think*, the focus is expected to be on the subordinate clause. In order to predict the answer type of the question, the subordinate clause is then treated the same way as the complete question was. For example, consider the question "Why do the school councils believe that class sizes will grow even more?". Since the declarative (*believe*) is non-factive,

the question analysis module determines the answer type for the subordinate clause ("class sizes will grow even more"), which is *cause*, and assigns it to the question as a whole.

Special attention is also paid to questions with a modal auxiliary. Modal auxiliaries like *can* and *should*, have an influence on the answer type. For example, consider the questions below, in which the only difference is the presence or absence of the modal auxiliary *can*:

1. Why did McDonald's not use actors to portray chefs in amusing situations?

2. Why can McDonald's not use actors to portray chefs in amusing situations?

Question 1 expects a motivation as answer, whereas question 2 expects a cause. We implemented this difference in our question analysis module: `can` (can, could) and `have to` (have to, has to, had to) lead to the answer type *cause*. Furthermore, the modal auxiliary `shall` (shall, should) changes the expected answer type to *motivation*.

When choosing an answer type, our question analysis module follows a conservative policy: in case of doubt, no answer type is assigned.

We did not yet perform a complete evaluation of our question analysis module. For proper evaluation of the module, we need a reference set of questions and answers that is different from the data set that we collected for development of our system. Moreover, for evaluating the relevance of our question analysis module for answer retrieval, further development of our approach is needed.

However, to have a general idea of the performance of our method for answer type determination, we compared the output of the module to manual classifications. We performed these reference classifications ourselves.

First, we manually classified 130 *why*-questions from our development set with respect to their syntactic category. Evaluation of the syntactic categorization is straightforward: 95 percent of *why*-questions got assigned the correct syntactic category using 'perfect' parse trees. The erroneous classifications were due to differences in the definitions of the specific verb types. For example, *argue* is not in the list of declarative verbs, as a result of which a question with argue as main verb is classified as action question instead of declarative layer question. Also, *die* and *cause* are not in the list of process verbs, so questions with either of these verbs as main verb are labeled as action questions instead of process questions.

Secondly, we performed a manual classification into the four answer subtypes (cause, motivation, circumstance and purpose). For this classification, we used the

same set of 130 questions as we did for the syntactic categorization, combined with the corresponding answers. Again, we performed this classification ourselves.

During the manual classification, we assigned the answer type *cause* to 23.3 percent of the questions and *motivation* to 40.3 percent. We were not able to assign an answer subtype to the remaining pairs (36.4 percent). These questions are in the broader class *reason* and not in one of the specific subclasses None of the question-answer pairs was classified as circumstance or purpose. Descriptions of purpose are very rare in news texts because of their generic character (e.g. "People have eyebrows to prevent sweat running into their eyes"). The answer type circumstance, defined by Quirk [81] (section 15.45) as a combination of reason with conditionality, is also rare as well as difficult to recognize.

For evaluation of the question analysis module, we mainly considered the questions that did get assigned a subtype (motivation or cause) in the manual classification. Our question analysis module succeeded in assigning the correct answer subtype to 62.2% of these questions, the wrong subtype to 2.4%, and no subtype to the other 35.4%. The set of questions that did not get a subtype from our question analysis module can be divided in four groups:

1. Action questions for which the subject was incorrectly not marked as agentive (mostly because it was an agentive organization like *McDonald's*, or a proper noun that was not in WordNet's list of nouns denoting persons, like *Henk Draijen*);

2. Questions with an action verb as main verb but a non-agentive subject (e.g. "Why will restrictions on abortion damage women's health?");

3. Passive questions, for which we have not yet developed an approach (e.g. "Why was the Supreme Court reopened?");

4. Monotransitive *have* questions. This category contains too few questions to formulate a general rule.

Group (1), which is by far the largest of these four (covering half of the questions without subtype), can be reduced by expanding the list of agentive nouns, especially with names of organizations. For groups (3) and (4), general rules may possibly be created in a later stage.

With this knowledge, we are confident that we can reduce the number of questions without subtype in the output of our question analysis module.

These first results predict that it is possible to reach a relatively high precision in answer type determination. (Only 2% of questions got assigned a wrong subtype.)

A high precision makes the question analysis output useful and reliable in the next steps of the question answering process. On the other hand, it seems difficult to get a high recall. In this test, only 62.2% of the questions that were assigned an answer type in the reference set, was assigned an answer type by the system — this is 39.6% of the total.

2.4 Conclusions and further research

We created a data collection for research into *why*-questions and for development of a method for *why*-QA. The collection comprises a sufficient amount of *why*-questions. For each question, the source document and one or two user-formulated answers are available in the data set. The resulting data set is of importance for our research as well as other research addressing the problem of *why*-QA.

We developed a question analysis method for *why*-questions, based on syntactic categorization and answer type determination. We believe that the test results, which show a high precision and reasonable recall, are promising for future development of our method for *why*-QA.

We think that, just as for factoid-QA, answer type determination can play an important role in question analysis for *why*-questions. Therefore, Kupiec' suggestion that conventional question analysis techniques are not suitable for *why*-QA [50] can be made more precise by saying that these methods may be useful for a (potentially small) subset of *why*-questions. The issue of recall, both for human and machine processing, needs further analysis.

In the near future, our work will focus on development of the next part of our approach for *why*-QA.

Until now we have focused on the first of four subtasks in QA, viz. (1) question analysis (2) retrieval of candidate paragraphs; (3) analysis and extraction of text fragments; and (4) answer generation. Of the remaining three subtasks, we will focus on paragraph analysis (3). In order to clarify the relevance of the paragraph analysis step, let us briefly discuss the QA-processes that follows question analysis.

The retrieval module, which comes directly after the question analysis module, uses the output of the question analysis module for finding candidate answer paragraphs (or documents). Paragraph retrieval can be straightforward: in existing approaches for factoid-QA, candidate paragraphs are retrieved based on keyword matching only. For this thesis, we do not aim at creating our own paragraph retrieval technique but apply an off-the-shelf retrieval engine instead.

More interesting for this thesis than paragraph retrieval is the next step of QA: paragraph analysis. The paragraph analysis module tries to determine whether the

candidate paragraphs contain potential answers. In case of *who*-questions, noun phrases denoting persons are potential answers; in case of *why*-questions, reasons (explanations) are potential answers. In the paragraph analysis stage, our answer subtypes come into play. The question analysis module determines the answer type for the input question, which is motivation, cause, purpose, or circumstance. The paragraph analysis module uses this information for searching candidate answers in a paragraph. As has been said before, the procedure for assigning the correct subtype needs further investigation in order to increase the coverage and the contribution that answer subtype classification can make to the performance of *why*-QA.

In the next chapter, we will investigate the use of Rhetorical Structure Theory (RST) for paragraph analysis and answer extraction.

3

Discourse-based Answering of Why-questions

Edited from: S. Verberne, L. Boves, N. Oostdijk, and P.A. Coppen. Discourse-based Answering of Why-Questions. *Traitement Automatique des Langues (TAL), special issue on "Discours et document: traitements automatiques"*, 47(2):21-41, 2007.

Abstract

In this chapter, we investigate the value of discourse structure for *why*-question answering (*why*-QA). We developed a module for answer extraction that employs the discourse relations in a pre-annotated document collection (the RST Treebank). With this method, we obtain a recall of 53.3% with a mean reciprocal rank (MRR) of 0.662. We argue that the maximum recall that can be obtained from the use of Rhetorical Structure Theory (RST relations) as proposed in this chapter is 58.0%. If we discard the questions that require world knowledge, maximum recall is 73.9%. We conclude that discourse structure can play an important role in complex question answering, but that more forms of linguistic processing are needed for increasing recall.

3.1 Introduction

Researchers in the field of discourse analysis have investigated whether knowledge about discourse structure can be put to use in a number of applications, among which language generation, text summarization, and machine translation [18]. The relevance of discourse analysis for QA applications has been suggested by Marcu and Echihabi [61] and Litkowski [55]. Breck et al. [10] suggest that knowledge about discourse relations would have allowed their system for TREC-8 to answer *why*-questions. In this chaper we take on the challenge and investigate to what extent discourse structure does indeed enable answering *why*-questions.

In the context of our research, a *why*-question is defined as an interrogative sentence in which the interrogative adverb *why* (or a synonymous word or phrase) occurs in (near) initial position. Furthermore, we only consider the subset of *why*-questions that could be posed to a QA system (as opposed to questions in a dialogue or in a list of frequently asked questions) and for which the answer is known to be present in some related document collection. In particular, our research is limited to a set of questions that we obtained from a number of subjects who were asked to read documents from the collection and formulate *why*-questions that another person would be able to answer given the text.

The answer to a *why*-question is a clause or sentence (or a small number of coherent sentences) that answers the question without adding supplementary and redundant context. The answer is not necessarily literally present in the source document, but it must be possible to deduce it from the document.

An approach for automatically answering *why*-questions, like general approaches for factoid-QA, will involve at least four subtasks, as listed in Chapter 2: (1) question analysis and query creation, (2) retrieval of candidate paragraphs or documents, (3) analysis and extraction of text fragments, and (4) answer generation. In the current research, we want to investigate whether structural analysis and linguistic information can make QA for *why*-questions feasible. In Chapter 2, we focused on question analysis for *why*-questions. From other research reported on in the literature it appears that knowing the answer type helps a QA system in selecting potential answers. Therefore, we created a syntax-based method for the analysis of *why*-questions that was aimed at predicting the semantic answer type. We defined the following answer types for *why*-questions, based on [81]: motivation, cause, circumstance and purpose. Of these, cause (52%) and motivation (37%) are by far the most frequent types in our set of *why*-questions pertaining to newspaper texts. With our syntax-based method, we were able to predict the correct answer type for 77.5% of these questions [102].

After analysis of the input question, the QA system will retrieve a small set of documents that possibly contain the answer. Analysis of the retrieved documents is then needed for extracting potential answers. Thus, a system for *why*-QA needs a text analysis module that yields a set of potential answers to a given *why*-question. Although we now have a proper answer type determination approach, the problem of answer extraction is still difficult. As opposed to factoid-QA, where named entity recognition can play an important role in the extraction of potential answers, finding potential answers to *why*-questions is still an unsolved problem. This means that we need to investigate how we can recognize the parts of a text that are potential answers to *why*-questions.

We decided to approach this answer extraction problem as a discourse analysis task. In this chapter, we aim to find out to what extent discourse analysis can help in selecting answers to *why*-questions. We also investigate the possibilities of a method based on textual cues, and use that approach as a baseline for evaluating our discourse-based method. Below, we will first introduce Rhetorical Structure Theory (RST) as a model for discourse analysis (Section 3.2). Then we present our method for employing RST for *why*-QA (Section 3.3), followed by the results that we obtained (Section 3.4). We conclude this chapter with a discussion of the limitations and possibilities of discourse analysis for the purpose of *why*-QA and the implications for future work (Sections 3.5 and 3.6).

3.2 Rhetorical Structure Theory (RST)

The main reasons for using RST as a model for discourse structure in the present research are the following. First, a treebank of manually annotated English texts with RST structures is available for training and testing purposes. This RST Discourse Treebank, created by Carlson et al. [18], contains a selection of 385 Wall Street Journal articles from the Penn Treebank that have been annotated with discourse structure in the framework of RST. Carlson et al. adapted the default set of discourse relations proposed by Mann and Thompson for the annotation of the Wall Street Journal articles in the treebank. The annotations by Carlson et al. are largely syntax-based, which fits the linguistic perspective of the current research. A second reason for using RST is that relatively good levels of agreement have been measured between human annotators of RST, which indicates that RST analyses do not strongly depend on subjective interpretations of the structure of a text [9].

In RST, the smallest units of discourse are called elementary discourse units (EDUs). In terms of the RST model, a rhetorical relation typically holds between two EDUs, one of which (the nucleus) is more essential for the writer's intention

than the other (the satellite). If two related EDUs are of equal importance, there is a multinuclear relation between them. Two or more related EDUs can be grouped together in a larger span, which in its turn can participate in another relation. By grouping and relating spans of text, a hierarchical structure of the text is created. In the remainder of this chapter, we will refer to such a hierarchical structure as an *RST tree*.

3.3 Our method for discourse-based why-QA

3.3.1 Main ideas and procedure

Let us consider a *why*-question–answer pair and the RST structure of the corresponding source text. We hypothesize the following:

1. The question topic[1] corresponds to a span of text in the source document and the answer corresponds to another span of text;

2. In the RST structure of the source text, an RST relation holds between the text span representing the question topic and the text span representing the answer.

If both hypotheses are true, then RST can play an important role in answering *why*-questions.

For the purpose of testing our hypotheses, we need a number of RST annotated texts and a set of question–answer pairs that are linked to these texts. Therefore, we set up an elicitation experiment using the RST Treebank as data set. We selected seven texts from the RST Treebank of 350–550 words each. Then we asked native speakers of English to read one of these texts and to formulate *why*-questions for which the answer could be found in the text. The subjects were also asked to formulate answers to each of their questions. This resulted in a set of 372 *why*-question and answer pairs, pertaining to seven texts from the RST Treebank. On average, 53 question–answer pairs were formulated per source text. There is much overlap in the topics of the questions, as we will see later.

A risk of gathering questions following this method, is that the participants may feel forced to come up with a number of *why*-questions. This may lead to a set of questions that is not completely representative for a user's real information need. We believe however that our elicitation method is the only way in which we can collect

[1] The topic of a *why*-question is the proposition that is questioned. A *why*-question has the form 'WHY P?', in which the proposition P is the topic [97].

questions connected to a specific (closed) set of documents. We will come back to the representativeness of our data collection in Section 3.5.3.

We performed a manual analysis on 336 of the collected question–answer pairs in order to check our hypotheses – we left out the other (randomly selected) pairs for future testing purposes (not addressed in this thesis). We chose an approach in which we analyzed our data according to a clear step-by-step procedure, which we expect to be suitable for answer extraction performed by a QA system. This means that our manual analysis will give us an indication of the upper bound of the performance that can be achieved using RST following the proposed approach.

First, we selected a number of relation types from Carlson's relation set [18], which we believed might be relevant for *why*-QA. We started with the four answer types mentioned in the introduction of this chapter (cause, purpose, motivation and circumstance), but it soon appeared that there is no one-to-one relation between the four classes we defined based on Quirk et al. [81] and relation types in Carlson's set. For instance, Carlson's relation set does not contain the relation type *motivation*, but uses *reason* instead. Moreover, we found that the set of relations to which at least one *why*-question in our data collection refers is broader than just cause, circumstance, purpose and reason. Therefore, we extended the list during the manual analysis. The final set of selected relations is shown in Table 3.1.

Table 3.1: Selected relation types

Cause	Circumstance	Condition
Elaboration	Explanation-argumentative	Evidence
Interpretation	List	Problem-Solution
Purpose	Reason	Result
Sequence		

For the majority of these relations, the span of text that needs explanation (or elaboration, evidence, etc.) is the nucleus of the relation, and the span of text giving this explanation is the satellite. The only exception to this rule is the cause relation, where the cause is given by the nucleus and its result by the satellite. Knowing this, we used the following procedure for analyzing the questions and answers:

1. Identify the topic of the question.

2. In the RST tree of the source document, identify the span(s) of text that express(es) the same proposition as the question topic.

3. Is the found span the nucleus of a relation of one of the types listed in Table

3.1 (or, in case of cause relations, the satellite)? If it is, go to 4. If it is not, go to 5.

4. Select the related satellite (or nucleus in case of a cause relation) of the found span as an answer.

5. Discard the current text span.

The effects of the procedure can best be demonstrated by means of an example. Consider the following question, formulated by one of the subjects after he had read a text about the launch of a new TV channel by Whittle Communications L.P.

> (1) Q: "Why does Christopher Whittle think that Channel One will have no difficulties in reaching its target?"

The topic of this question is "Christopher Whittle thinks that Channel One will have no difficulties in reaching its target". According to our first hypothesis, the proposition expressed by the question topic matches a span in the RST structure of the source document. We manually selected the following text fragment which expresses the proposition of the question topic:

> (2) "What we've done in eight weeks shows we won't have enormous difficulties getting to the place we want to be", said Mr. Whittle.

This sentence covers span 18–22 in the corresponding RST tree, which is shown in Figure 3.1.

In this way, we tried to identify a span of text corresponding to the question topic for each of the 336 questions. In cases where we succeeded in selecting a span of text in the RST tree corresponding to the question topic, we searched for potential answers following step 3 and 4 from the analysis procedure. As we can see in Figure 3.1, the span "What we've done in eight weeks shows we won't have enormous difficulties getting to the place we want to be, said Mr. Whittle" is the nucleus of an evidence relation. Since we assumed that an evidence relation may lead to a potential answer (Table 3.1), we can select the satellite of this relation, span 23–28, as an answer (see Figure 3.2 below):

> (3) A: "He said his sales force is signing up schools at the rate of 25 a day. In California and New York, state officials have opposed Channel One. Mr. Whittle said private and parochial schools in both states will be canvassed to see if they are interested in getting the programs."

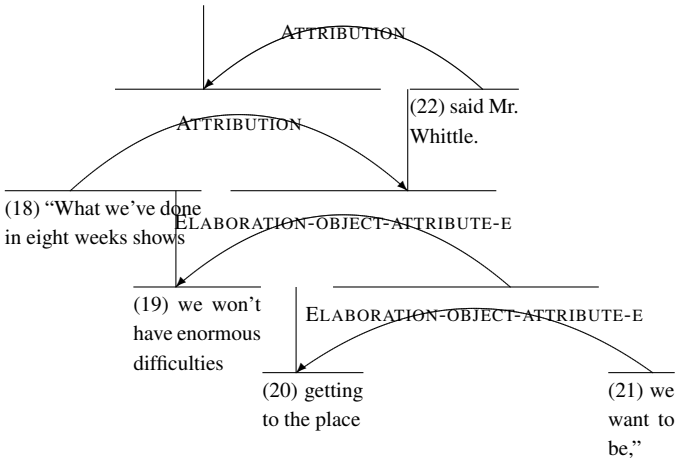

Figure 3.1: RST subtree for the text span "What we've done in eight weeks shows we won't have enormous difficulties getting to the place we want to be, said Mr. Whittle."

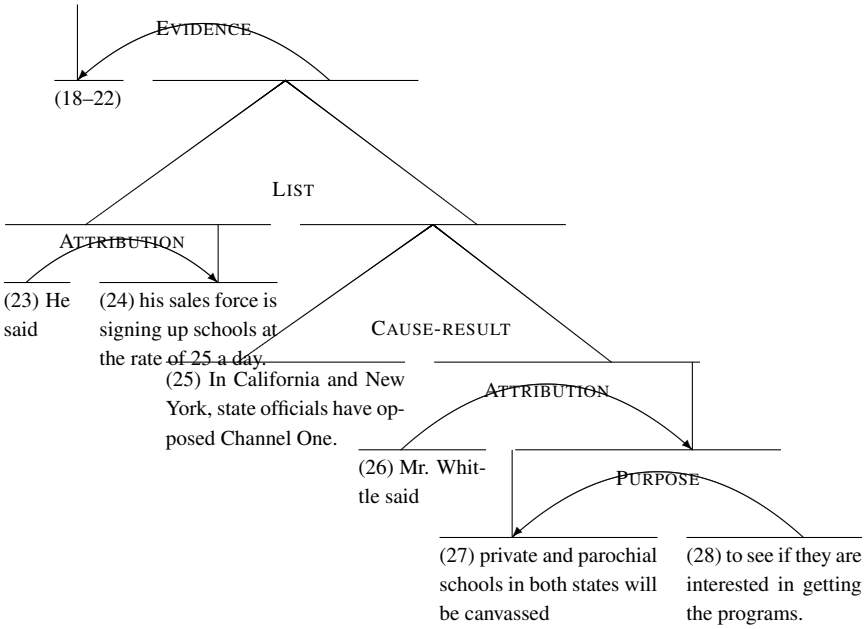

Figure 3.2: RST subtree containing the satellite span "He said his sales force ... to see if they are interested in getting the programs."

We analyzed all 336 *why*-questions following this procedure. The result of this manual analysis is a table containing all questions and for each question the following fields: (a) the manually identified topic from the source text with its corresponding span from the RST tree; (b) the answer span that we found for the question topic; (c)

the type of relation that holds between topic span and answer span, if there is a relation; and (d) information about whether the answer found is correct. We will come back to this in Section 3.4.1, where we discuss the outcome of the manual analysis.

3.3.2 Implementation

We implemented the procedure presented above in a Perl script. In Section 3.3.1, we assumed that the RST structure can lead to a possible answer span once the topic span has been identified as a nucleus of a relevant relation. Therefore, the most critical task of our procedure is step 2: to identify the span(s) of text that express(es) the same proposition as the question topic.

Since we were only interested in those spans of text that participate in an RST relation (step 3), we needed a list of all nuclei and satellites for each document in our data collection, so that our module could select the most relevant nuclei. Therefore, we built an indexing script that takes as input file the RST structure of a document, and searches for instances of relevant relations (Table 3.1). For each relation, it then extracts its nucleus, satellite and relation type and saves it to an index file (in plain text). In case of a multinuclear relation, the script saves both nuclei to the index file. Moreover, cause relations are treated a bit differently from the other relation types. In cause relations, as explained before, the span of text describing the cause is marked as nucleus, not as satellite. Thus, the satellite of cause relations should be indexed for matching to the question topic instead of the nucleus. Therefore, nucleus and satellite are transposed when indexing cause relations. Below, where we use the term *nucleus* in describing the retrieval process, we mean the satellite for cause relations and the nucleus for all other relations. Figures 3.3 and 3.4 below illustrate the conversion from an RST structure file to an index file. We created indexes for all documents in the RST Treebank.

```
( Nucleus (span 29 32) (rel2par span)
  ( Nucleus (leaf 29) (rel2par span) (text \_!that interior regions of Asia
                                          would be among the first\_!) )
  ( Satellite (span 30 32) (rel2par elaboration-object-attribute-e)
    ( Nucleus (span 30) (rel2par span) (text \_!to heat up in a global warming\_!) )
    ( Satellite (span 31 32) (rel2par consequence-n-e)
      ( Nucleus (leaf 31) (rel2par span) (text \_!because they are far from oceans,\_!) )
      ( Satellite (leaf 32) (rel2par elaboration-additional)
                            (text \_!which moderate temperature changes.<P>\_!) )
    )
  )
)
```

Figure 3.3: Fragment of the original RST structure

```
> consequence
   1. Nucleus (30): to heat up in a global warming
   2. Satellite (31 32): because they are far from oceans, which moderate temperature changes

> elaboration
   1. Nucleus (31): because they are far from oceans
   2. Satellite (32): which moderate temperature changes
```

Figure 3.4: Fragment of the resulting index

For the actual retrieval task, we wrote a second Perl script that takes as input one of the document indices, and a question related to the document. Then it performs the following steps:

1. Read the index file and normalize each nucleus in the index. Normalization includes at least removing all punctuation from the nucleus. Other forms of normalizing that we explored are lemmatization, applying a stop list, and adding synonyms for each content word in the nucleus. These normalization forms are combined into a number of configurations, which are discussed in Section 3.4.2;

2. Read the question and normalize it, following the same normalization procedure as for the nuclei;

3. For each nucleus in the index, calculate the likelihood $P(Nucleus|Question)$ using the following language model (N = nucleus; Q = question; R = relation type for nucleus):

   Nucleus likelihood $P(N|Q) \sim P(Q|N) \cdot P(N)$

   Question likelihood $P(Q|N) = \frac{\#\ question\ words\ in\ nucleus}{\#\ words\ in\ nucleus}$

   Nucleus Prior $P(N) = \frac{1}{\#\ nuclei\ in\ document} \cdot P(R)$

   Relation Prior $P(R) = \frac{\#\ occurrences\ of\ this\ relation\ type\ in\ question\ set}{\#\ occurrences\ of\ this\ relation\ type\ in\ treebank}$

   For calculation of the relation prior P(R), we counted the number of occurrences of each relation type in the complete RST Treebank. We also counted the number of occurrences to which at least one question in our data collection refers. The proportion between these numbers, the relation prior, is an indication of the relevance of the relation type for *why*-question and answer pairs.
   For convenience, we take the logarithm of the likelihood. This avoids underflow problems with very small probabilities. Thus, since the range of the likelihood is [0..1], the range of the logarithm of the likelihood is [-∞..0];

4. Save all nuclei with a likelihood greater than the predefined threshold (see Section 3.4.2);

5. Rank the nuclei according to their likelihood;

6. For each of the nuclei saved, print the corresponding answer satellite and the calculated likelihood.

We measured the performance of our implementation by comparing its output to the output of the manual analysis described in Section 3.3.1.

3.4 Results

In this section, we will first present the outcome of the manual analysis, which gives an indication of the performance that can be achieved by a discourse-based system answer extraction module for *why*-QA (Section 3.4.1).

Then we present the performance of the current version of the answer extraction module. When presenting the results, we can distinguish two types of measurements. First, we can measure the module's absolute quality in terms of recall and mean reciprocal rank (MRR). Second, we can measure its performance relative to the results we obtained from the manual analysis. In Section 3.4.2, we do both.

3.4.1 Results of the manual analysis

As described in Section 3.3.1, our manual analysis procedure consists of four steps: (1) identification of the question topic, (2) matching the question topic to a span of text, (3) checking whether this span is the nucleus of an RST relation (or satellite, in case of a cause relation), and (4) selecting its satellite as answer. Below, we will discuss the outcome of each of these subtasks.

The first step succeeds for all questions, since each *why*-question has a topic. For the second step, we were able to identify a text span in the source document that represents the question topic for 279 of the 336 questions that we analyzed (83.0%). We found that not every question corresponds to a unique text span in the source document. For these 279 questions, we identified 84 different text spans. This means that on average, each text span that represents at least one question topic is referred to by 3.3 questions. For the other 57 questions, we were not able to identify a text span in the source document that represents the topic. These question topics are not explicitly mentioned in the text but inferred by the reader using world knowledge. We will come back to this in Section 3.5.1.

For 207 of the 279 questions that have a topic in the text (61.6% of all questions), the question topic participates in a relation of one of the types in Table 3.1 (step III).

Evaluation of the fourth step, answer extraction, needs some more explanation. For each question, we selected as an answer the satellite that is connected to the nucleus corresponding to the question topic. For the purpose of evaluating the answers found using this procedure, we compared them to the user-formulated answers. If the answer found matches at least one of the answers formulated by native speakers in meaning (not necessarily in form), then we judged the answer found as correct. For example, for the question "Why did researchers analyze the changes in concentration of two forms of oxygen?", two native speakers gave as an answer "To compare temperatures over the last 10,000 years", which is exactly the answer that we found following our procedure. Therefore, we judged our answer as correct, even though eight subjects gave a different answer to this question. Evaluating the answer that we found to the question "Why does Christopher Whittle think that Channel One will have no difficulties in reaching its target?" (example 3 in Section 3.3.1) is slightly more difficult, since it is longer than any of the answers formulated by the native speakers. We got the following user-formulated answers for this question:

1. Because schools are subscribing at the rate of 25 a day.

2. Because agents are currently signing up 25 schools per day.

3. He thinks he will succeed because of what he has been able to do so far.

4. Because of the success of the previous 8 weeks.

Answers 1 and 2 refer to leaf 24 in the RST tree (see Figure 3.2); answers 3 and 4 refer to leaf 18 in the tree (see Figure 3.1). None of these answers correspond exactly to the span that we found as answer using the answer extraction procedure ("He said his sales force ... in getting the programs."). However, since some of the user-formulated answers are part of the answer span found, and because the answer is still relatively short, we judged the answer found as correct.

We found that for 195 questions, the satellite that is connected to the nucleus corresponding to the topic is a correct answer. This is 58.0% of all questions. The above figures are summarized in Table 3.2.

In Section 3.5.1, we will come back to the set of questions (42%) for which our procedure did not succeed.

Table 3.2: Outcome of manual analysis

Question	# questions	% of questions
Questions analyzed	336	100
Questions for which we identified a text span corresponding to the topic	279	83.0
Questions for which the topic corresponds to the nucleus of a relation (or satellite in case of a cause relation)	207	61.6
Questions for which the satellite of this relation is a correct answer	195	58.0

3.4.2 Evaluation of the implementation

We evaluate our answer extraction module using the outcome of our manual analysis as reference. We used the answer that we found during manual analysis as reference answer. We measured recall (the proportion of questions for which the system gives at least the reference answer) and MRR (1/rank of the reference answer, averaged over all questions for which the reference answer is retrieved.) We also measured recall as proportion of the percentage of questions for which the manual analysis led to the correct answer (58%, see Table 3.2 above).

We tested a number of configurations of our answer extraction module, in which we varied the following variables:

1. Applying a stop list to the indexed nuclei, i.e. removing occurrences of 251 high-frequent words, mainly function words;

2. Applying lemmatization, i.e. replacing each word by its lemma if it is in the CELEX lemma lexicon [3]. If it is not, the word itself is kept;

3. Expanding the indexed nuclei with synonym information from WordNet [27], i.e. for each content word in the nucleus (nouns, verbs and adjectives), searching the word in WordNet and adding to the index all lemmas from its synonym set;

4. Changing weights between stop words and non-stop words.

We found that best performing is the configuration in which stop words are not removed, lemmatization is applied, no synonyms are added, and stop words and non-stop words are weighted 0.1/1.9. Moreover, in order to reduce the number of answers per question, we added a threshold to the probability of the nuclei found. For

deciding on this threshold, we investigated what the log probability is that the answer extraction module calculates for each of the correct (reference) answers in our data collection. As threshold, we chose a probability that is slightly lower than the probabilities of these reference answers.

We ran the answer extraction module on the 336 questions from our data collection and compared the answer spans found by the system to the answer in the reference table that was manually created (see Section 3.3.1)

With the optimal configuration as described above, the extraction module found the reference answer for 179 questions. So, it obtains a recall of 53.3% (179/336). This is 91.8% of the questions for which the RST structure led to the correct answer in the manual analysis (179/195). The average number of answers that the extraction module gives per question is 16.7. The mean reciprocal rank for the reference answer is fairly high: 0.662. For 29.5% of all questions, the reference answer is ranked in first position. This is 55.3% of the questions for which the extraction module retrieved the reference answer.

An overview of the results is given in Tables 3.3 and 3.4 below. We should note here that recall will go up if we add synonyms to the index for all nuclei, but this lowers MRR and heavily slows down the question-nucleus matching process.

Table 3.3: Main results for optimal configuration

Recall (%)	53.3
Recall as proportion of questions for which the RST structure can lead to a correct answer (%)	91.8
Average number of answers per question	16.7
Mean reciprocal rank (MRR)	0.662

Table 3.4: Ranking of reference answer

Answer rank	# questions	% of questions
Reference answer found	179	53.3
Reference answer ranked in 1st position	99	55.3
Reference answer ranked in 2nd–10th position	60	33.5
Reference answer ranked in other position	20	11.2
Reference answer not found	157	46.7

3.5 Discussion of the results

In the discussion of the results that we obtained, we will focus on two groups of questions. First, we will discuss the questions for which we could not find an answer

in our manual analysis following the procedure proposed (procedure shortcomings). Second, we will consider the questions for which we found an answer using manual analysis but the answer extraction module could not find this answer (system shortcomings). For both groups of questions, we will study the cases for which we did not succeed, and make recommendations for future improvements of the extraction module. In the last part of this section, we will give an overview of the types of RST relations that were found to play a role in *why*-QA.

3.5.1 Discussion of procedure shortcomings

In this section, we will first provide an error analysis and then compare our results to a baseline.

Error analysis

We reported in Section 3.4.1 that for 195 *why*-questions (58.0% of all questions), the answer could be found after manually matching the question topic to the nucleus of an RST relation and selecting its satellite as answer. This means that for 141 questions (42.0%), our method did not succeed. We distinguish four categories of questions for which we could not extract a correct answer using this method (percentages are given as part of the total of 336 questions):

1. Questions whose topics are not or only implicitly supported by the source text (57 questions, 17.0%). Half of these topics is supported by the text, but only implicitly. The propositions underlying these topics are true according to the text, but we cannot denote a place in the text where this is confirmed explicitly. Therefore, we were not able to select a span corresponding to the topic. For example, the question "Why is cyclosporine dangerous?" refers to a source text that reads "They are also encouraged by the relatively mild side effects of FK-506, compared with cyclosporine, which can cause renal failure, morbidity, nausea and other problems." We can deduce from this text fragment that cyclosporine is dangerous, but we need knowledge of the world "renal failure, morbidity, nausea and other problems are dangerous") to do this. For the other half of these questions, the topic is not supported at all by the text, even not implicitly. For example, "Why is the initiative likely to be a success?", whereas nowhere in the text there is evidence that the initiative is likely to be a success.

2. Questions for which both topic and answer are supported by the source text but there is no RST relation between the span representing the question topic span

and the answer span (55 questions, 16.4%). In some cases, this is because the topic and the answer refer to the same EDU. For example, the question "Why were firefighters hindered? refers to the span "Broken water lines and gas leaks hindered firefighters' efforts," which contains both question topic and answer. In other cases, question topic and answer are embedded in different, non-related spans, which are often remote from each other.

3. Questions for which the correct (i.e. user-formulated) answer is not or only implicitly supported by the text (17 questions, 5.1%). In these cases, the question topic is supported by the text, but we could not find evidence in the text that the answer is true or we are not able to identify the location in the text where it is confirmed explicitly. For example, the topic of the question "Why was Gerry Hogan interviewed?" corresponds to the text span "In an interview, Mr. Hogan said". The native speaker that formulated this question gave as answer "Because he is closer to the activity of the relevant unit than the Chair, Ted Turner, since he has the operational role as President." The source text does read that Mr. Hogan is president and that Ted Turner is chair, but the assumption that Gerry Hogan is closer to the activity than Ted Turner has been made by the reader, not by the text.

4. Questions for which the topic can be identified in the text and matched to the nucleus of a relevant RST relation, but the corresponding satellite is not suitable or incomplete as answer (12 questions, 3.6%). These are the questions that in Table 3.2 make the difference between the last two rows (207-195). Some answers are unsuitable because they are too long. For instance, there are cases where the complete text is an elaboration of the sentence that corresponds to the question topic. In other cases, the answer satellite is incomplete compared to the user-formulated answers. For example, the topic of the question "Why did Harold Smith chain his Sagos to iron stakes?" corresponds to the nucleus of a circumstance relation that has the satellite "After three Sagos were stolen from his home in Garden Grove". Although this satellite gives a possible answer to the question, it is incomplete according to the user-formulated answers, which all mention the goal "To protect his trees from thieves".

Questions of category 1 above cannot be answered by a QA system that expects the topic of an input question to be present and identifiable in a closed document collection. If we are not able to identify the question topic in the text manually, then a retrieval system cannot either.

A comparable problem holds for questions of category 3, where the topic is supported by the source text but the answer is not or only implicitly. If the system searches for an answer that cannot be identified in a text, the system will clearly not find it in that text. In the cases where the answer is implicitly supported by the source text, world knowledge is often needed for deducing the answer from the text, like in the examples of cyclosporine and Gerry Hogan above.

Therefore, we consider the questions of types 1 and 3 as unsolvable by a QA system that searches for the question topic in a closed document collection. Together these categories cover 22.0% of all *why*-questions.

Questions of category 2 (16.4% of all questions) are the cases where both question topic and answer can be identified in the text, but where there is no RST relation between the span representing the question topic span and the answer span. We can search for ways to extent our algorithm so that it can handle some of the cases mentioned. For instance, we can add functionality for managing question–answer relations on sub-EDU level.

We think that in some of these cases, syntactic analysis can help in extracting the relation from the EDU. The example question above, "Why were firefighters hindered?" can be answered by a QA system if it knows that the question can be rephrased by "What hindered firefighters?", and that has syntactic information about the EDU "Broken water lines and gas leaks hindered firefighters' efforts". The risk of adding functionality for cases like this is that the number of possible answers per question will increase, decreasing the MRR. We should investigate to what extent syntactic analysis can help in cases where the answer lies in the same EDU as the question.

For cases where question topic and answer are embedded in non-related spans, we can at the moment not propose smart solutions that will increase recall without heavily decreasing the MRR. The same holds for questions of category 4 (3.3%), where RST leads to an answer that is incomplete or unsuitable.

We can conclude from this analysis that there is a subset of *why*-questions (22.0%) that cannot be answered by a QA system that uses a closed document collection since knowledge of the world is essential for answering these questions. Moreover, there is a further subset of *why*-questions (16.4% + 3.6%) that cannot be answered by a system that uses RST structure only, following the approach that we proposed. Together, this means that 42.0% of *why*-questions cannot be answered following the suggested approach. Thus, the maximum recall that can be achieved with this method is 58.0%. If we discard the 72 (57+15) questions that require world knowledge, maximum recall would be 73.9% (195/(336-72)).

Comparison to baseline

In order to judge the merits of RST structure for *why*-QA, we investigated the possibilities of a method based on textual cues (without discourse structure). To that goal, we analyzed the text fragments related to each question–answer pair in our data collection. For each of these pairs, we identified the item in the text that indicates the answer. For 50% of the questions, we could identify a word or group of words that in the given context is a cue for the answer. Most of these cues, however, are very frequent words that also occur in many non-cue contexts. For example, the subordinator *that* occurs 33 times in our document collection, only 3 of which are referred to by one or more *why*-questions. This means that only in 9% of the cases, the subordinator *that* is a *why*-cue. The only two words for which more than 50% of the occurrences are *why*-cues, are *because* (for 4.5% of questions) and *since* (2.2%). Both are a *why*-cue in 100% of their occurrences. For almost half of the question–answer pairs that do not have an explicit cue in the source text, the answer is represented by the sentence that follows (17.6% of questions) or precedes (2.8%) the sentence that represents the question.

Having this knowledge on the frequency of cues for *why*-questions, we defined the following baseline approach:

1. Identify the topic of the question.
2. In the source document, identify the clause(s) that express(es) the same proposition as the question topic.
3. Does the clause following the matched clause start with *because* or *since*? If it does, go to 4. If it does not, go to 5.
4. Select the clause following the matched clause as answer.
5. Select the sentence following the sentence containing the matched clause as answer.

An answer extraction module that follows this baseline method can obtain a maximum recall of 24.3% (4.5+2.2+17.6). This means that an RST-based method can improve recall by almost 140% compared to a simple cue-based method (58.0% compared to 24.3%).

3.5.2 Discussion of system shortcomings

There are 22 questions for which the manual analysis led to a correct answer, but the extraction module did not retrieve this reference answer. For 17 of them, the nu-

cleus that was matched to the question topic manually, is not retrieved by the system because there is no (or too little, given the threshold) vocabulary overlap between the question and the nucleus that represents its topic. For example, the question "Why are people stealing cycads?" can manually be matched to the span "palm-tree rustling is sprouting up all over Southern California", but there are no overlapping words. If we add synonyms to our index for each nucleus (see Section 3.3.2), then 10 of these questions can be answered by the system, increasing recall.

For three other questions, it is our algorithm that fails: these are cases where the question topic corresponds to the satellite of an elaboration relation, and the answer to the nucleus, instead of vice versa. We implemented this functionality for cause relations (see Section 3.4.2), but implementing it for elaboration relations, where these topic-satellite correspondences are very rare, would increase the number of answers per questions and decrease MRR without increasing recall very much.

3.5.3 RST relations that play a role in why-QA

We counted the number of occurrences of the relation types from Table 3.1 for the 195 questions where the RST relation led to a correct answer. This distribution is presented in Table 3.5. The meaning of the column *Relative frequency* in this context will be explained below.

Table 3.5: Addressed relation types

Relation type	# referring questions	Relative frequency
Means	4	1.000
Consequence	30	1.000
Result	19	1.000
Purpose	28	0.857
Evidence	7	0.750
Reason	19	0.750
Explanation-argumentative	14	0.571
Cause	7	0.500
Condition	1	0.333
Interpretation	7	0.333
Circumstance	1	0.143
Elaboration	53	0.112
Sequence	1	0.091
List	4	0.016
Problem-Solution	0	0.000

As shown in Table 3.5, the relation type with most referring question–answer pairs, is the very general elaboration relation. It seems striking that elaboration is more frequent as a relation between a *why*-question and its answer than reason or cause. However, if we look at the relative frequency of the addressed relation types, we see another pattern: in our collection of seven source texts, elaboration is a very frequent relation type. In the seven texts that we consider, there are 143 occurrences of an elaboration relation. Of the 143 nuclei of these occurrences, 16 were addressed by one or more *why*-questions, which gives a relative frequency of around 0.1. Purpose, on the other hand, has only seven occurrences in our data collection, six of which being addressed by one or more questions, which gives a relative frequency of 0.857. Reason and evidence both have only four occurrences in the collection, three of which have been addressed by one or more questions. Consequence even has a relative frequency of 1.000

The table shows that if we address the problem of answer extraction for *why*-questions as a discourse analysis task, the range of relation types that can lead to an answer is broad and should not be implemented too rigidly.

In Section 3.3.1, we pointed out that our data collection may not be fully representative of a user's information need, due to our elicitation method using a closed document set. The relation types in Table 3.5 confirm that assumption to some extent: the presence of relation types such as means and condition suggests that the subjects in some cases formulated *why*-questions whereas they would have formulated *how*- or *when*-questions in case of an actual information need. A question–answer pair like "Why could FK-506 revolutionize the organ transplantation field?" — "Because it reduces harmful side effects and rejection rates", while the text reads "FK-506 could revolutionize the transplantation field by reducing harmful side effects" exemplifies this.

If we want to know our system's performance on *why*-questions that are representative for a user's information need, we are interested in those questions whose answers can be found through a 'core-*why* relation' like cause and reason.

If we only consider the relation types that have relative frequency higher than or equal to 0.5, we see that these relation types are in general closer to the concept of reason as general answer type of *why*-questions (see Chaper 2) than the relation types with a relative frequency lower than 0.5. We also see that the most frequent answer types that we defined for question analysis (see Section 3.1) come back in this set of relation types. Purpose and reason, as defined by Carlson[17], correspond to our definition of the answer type motivation (see Chapter 2). Carlson's consequence, result and cause relations can, based on their definitions, be grouped together as our answer type cause.

We investigated to what extent the performance of our answer extraction module depends on the type of relation that leads from question topic to the reference answer. For this purpose, we split the relation types found in two categories:

- Relation types that are conceptually close of the general answer type reason ('core-*why* relations'): Purpose, Consequence, Evidence, Reason, Result, Explanation-argumentative and Cause. These relation types all have a relative frequency higher than 0.5 for *why*-questions.

- Relation types that are less applicable to *why*-questions ('non-*why* relations'): Means, Condition, Interpretation, Circumstance, Elaboration, Sequence, List and Problem-Solution.

We considered the set of 207 questions for which the topic corresponds to the nucleus of a relation (thereby excluding the 74 questions whose topic or answer is unsupported, or where the RST relation does not lead to an answer) and measured our system's recall on this set of questions. This is 77.5% — which is higher than the total recall of 51.2% because we excluded the majority of problematic cases. We then split the set of 207 questions into one set of questions whose answers can be found through a core-*why* relation (130 questions), and one set of questions that correspond to a non-*why* relation (77 questions) and ran our answer extraction module on both these sets. For the core-*why* relation types, we found a recall of 88.5% and for the non-*why* relation types a recall of 60.3%. Moreover, we found that the remaining 11.5% for the core-*why* relation types suffer from lexical matching problems (see Section 3.5.2) instead of procedural problems: for 100% of these questions, the satellite of the relation is a correct answer. For the non-*why* relation types, this is 85.9%.

Another problem of our data collection method, is that the questions formulated by the readers of the text (in particular the questions relating to core-*why* relations) will probably be influenced by the same linguistic cues that are used by the annotators that built the RST structures: cue phrases (like *because* denoting an explanation relation) and syntactic constructions (like infinite clauses denoting a purpose relation). This is an unwelcome correlation, since in a working QA system users will not have access to the documents. In the following chapters, we will therefore work with a set of questions formulated by users of a QA system instead of elicited questions.

3.6 Conclusions

We created a method for *why*-QA that is based on discourse structure and relations. The main idea of our approach is that the propositions of a question topic and its answer are both represented by a text span in the source text, and that an RST relation holds between these spans. A *why*-question can then be answered by matching its topic to a span in the RST tree and selecting the related span as answer.

We first investigated the possible contribution of our RST approach to *why*-QA by performing a manual analysis of our set of 336 questions and answers collected through elicitation from native speakers pertaining to seven RST-annotated texts. From the evaluation of our manual analysis, we concluded that for 58.0% of our *why*-questions, an RST relation holds between the text span corresponding to the question topic and the text span corresponding to the answer.

We implemented this method for discourse-based *why*-QA using the RST Treebank as document collection. Our answer extraction module obtains a recall of 53.3% (91.8% of the manual score) with an MRR of 0.662.

In Section 3.5.1, we conclude from the analysis of procedure shortcomings that there is a subset of *why*-questions (22.0%) that cannot be answered by a QA system that expects the topic of an input question to be present and identifiable in a closed document collection. For these questions, either the topic or the user-formulated answer is not or only implicitly supported by the corresponding source text, which means that world knowledge is necessary for answering these questions. Furthermore, there is a further subset of *why*-questions (16.4%) that cannot be answered by a system that uses RST structure following the approach we proposed. For these questions, there is no RST relation between the span corresponding to the question topic and the span corresponding to its answer. A third subset (3.6%) of problematic questions contains those questions for which RST leads to an unsuitable or incomplete answer. Together, this means that 42.0% of *why*-questions cannot be answered following the suggested approach. Thus, the maximum recall that can be achieved with this method is 58.0%. If we discard the questions that require world knowledge, maximum recall would be 73.9%. An even higher performance can be achieved if we would only consider those questions that refer to core-*why* relations in the text such as cause and reason.

In the near future we will focus our research on three topics.

First, we aim to create and annotate a test corpus connected to *why*-questions that originate from real users' information needs, based on the *why*-questions collected for the Webclopedia project [39]. With this set, we will investigate first to what extent questions representing real information needs refer to *why*-relations in a document's

RST structure and second what the performance of our method is on such a set of questions.

Secondly, we should note that in a future application of *why*-QA using RST, the system will not have access to a manually annotated corpus—it has to deal with automatically annotated data. We assume that automatic RST annotations will be less complete and less precise than the manual annotations are. As a result of that, performance would decline if we were to use automatically created annotations. Some work has been done on automatically annotating text with discourse structure. Promising is the done work by [61], [87] and [41]. We plan to investigate to what extent we can achieve partial automatic discourse annotations that are specifically equipped to finding answers to *why*-questions. However, these investigations are not a part of this thesis. Instead, we refer to [91] for our findings on this topic.

Thirdly, we will in the next chapter focus on the task of passage retrieval for *why*-QA (the second step in the QA process): how can we improve an off-the-shelf passage retrieval module with knowledge of the structure of *why*-questions and their answers?

4

What is not in the Bag of Words for Why-QA?

Edited from: Suzan Verberne, Lou Boves, Nelleke Oostdijk, Peter-Arno Coppen. What is not in the Bag of Words for Why-QA?. To appear in *Computational Linguistics*, 36(2), 2010.

Abstract

In this chapter, we extend a passage retrieval module that uses off-the-shelf retrieval technology with a re-ranking step incorporating structural information. We get significantly higher scores in terms of MRR@150 (from 0.25 to 0.34) and Success@10. The 23% improvement that we reach in terms of MRR is comparable to the improvement reached on different QA tasks by other researchers in the field, although our re-ranking approach is based on relatively light-weight overlap measures incorporating syntactic constituents, cue words and document structure.

4.1 Introduction

In preliminary experiments on answering *why*-questions on the basis of Wikipedia (not included in this thesis), we found that the answers to most *why*-questions are passages of text that are at least one sentence and at most one paragraph in length [104]. Therefore, we aim at developing a system that takes as input a *why*-question and gives as output a ranked list of candidate answer passages.

In this chapter, we propose a three-step setup for a *why*-QA system: (1) a question processing module that transforms the input question to a query; (2) an off-the-shelf retrieval module that retrieves and ranks passages of text that share content with the input query; and (3) a re-ranking module that adapts the scores of the retrieved passages using structural information from the input question and the retrieved passages.[1]

In the first part of this chapter, we focus on step 2, viz. passage retrieval. The classic approach to finding passages in a text collection that share content with an input query is retrieval using a bag-of-words (BOW) model [82]. BOW models are based on the assumption that text can be represented as an unordered collection of words, disregarding grammatical structure. Most BOW-based models use statistical weights based on term frequency, document frequency, passage length and term density [90].

Since BOW approaches disregard grammatical structure, systems that rely on a BOW model have their limitations in solving problems where the syntactic relation between words or word groups is crucial. The importance of syntax for QA is sometimes illustrated by the sentence "Ruby killed Oswald", which is not an answer to the question "Who did Oswald kill?"[8]. Therefore, a number of researchers in the field investigated the use of structural information on top of a BOW approach for answer retrieval and ranking [92, 80, 88]. These studies show that although the BOW model makes the largest contribution to the QA system results, adding structural (syntactic information) can give a significant improvement.

We hypothesize that for the relatively complex problem of *why*-QA, a significant improvement — at least comparable to the improvement gained for factoid QA —

[1] We should note here that in the previous two chapters, we have assumed a four-step set-up for *why*-QA. Our current three-step proposal takes together steps 3 and 4: analysis and selection of text fragments, and answer generation. This is because for explanation-type answers, which are at least one sentence in length themselves, a passage of text that provides additional context to the answer is to be preferred above the exact answer alone. This is also confirmed by the literature on user preferences, which reads that users of QA systems prefer paragraph-sized chunks texts over just the exact answer [54].

can be gained from the addition of structural information to the ranking component of the QA system. We first evaluate a passage retrieval system for *why*-QA based on standard BOW ranking (step 1 and 2 in our setup). Then we perform an analysis of the strengths and weaknesses of the BOW model for retrieving and ranking candidate answers. In view of the observed weaknesses of the BOW model, we choose our feature set to be applied to the set of candidate answer passages in the re-ranking module (step 3 in our set-up).

The structural features that we propose are based on the idea that some parts of the question and the answer passage are more important for relevance ranking than other parts. Therefore, our re-ranking features are overlap-based: they tell us which parts of a *why*-question and its candidate answers are the most salient for ranking the answers. We evaluate our initial and adapted ranking strategies using a set of *why*-questions and a corpus of Wikipedia documents, and we analyze the contribution of both the BOW model and the structural features.

This chapter is organized as follows. In Section 4.2, related work is discussed. Section 4.3 presents the BOW-based passage retrieval method for *why*-QA, followed by a discussion of the strengths and weaknesses of the approach in Section 4.4. In Section 4.5, we extend our system with a re-ranking component based on structural overlap features. A discussion of the results and our conclusions are presented in Sections 4.6 and 4.7 respectively.

4.2 Related work

We distinguish related work in two directions: research into the development of systems for *why*-QA (Section 4.2.1), and research into combining structural and BOW features for QA (Section 4.2.2).

4.2.1 Research into why-QA

In Chapter 3, we focused on selecting and ranking explanatory passages for *why*-QA with the use of rhetorical structures. We developed a system that employs the discourse relations in a manually annotated document collection: the RST Treebank [18]. This system matches the input question to a text span in the discourse tree of the document and it retrieves as answer the text span that has a specific discourse relation to this question span. We evaluated our method on a set of 336 *why*-questions formulated to seven texts from the WSJ corpus. We concluded that discourse structure can play an important role in *why*-QA, but that systems relying on these structures can only work if candidate answer passages have been annotated

with discourse structure. Automatic parsers for creating full rhetorical structures are currently unavailable. Therefore, a more practical approach appears to be necessary for work in *why*-QA, viz. one which is based on automatically created annotations.

Higashinaka and Isozaki [37] focus on the problem of ranking candidate answer paragraphs for Japanese *why*-questions. They assume that a document retrieval module has returned the top 20 documents for a given question. They extract features for content similarity, causal expressions and causal relations from two annotated corpora and a dictionary. Higashinaka and Isozaki evaluate their ranking method using a set of 1000 *why*-questions that were formulated to a newspaper corpus by a text analysis expert. 70.3% of the reference answers for these questions is ranked in the top 10 by their system, and MRR[2] was 0.328.

Although the approach of Higashinaka and Isozaki is very interesting, their evaluation collection has the same flaw as the one that we used in Chapters 2 and 3: both collections consist of questions formulated to a pre-selected answer text. Questions elicited in response to newspaper texts tend to be unrepresentative for questions asked in a real QA setting. In the current work, therefore, we work with a set of questions formulated by users of an online QA system (see Section 4.3.1).

4.2.2 Combining structural and bag-of-words features for QA

Tiedemann [92] investigates syntactic information from dependency structures in passage retrieval for Dutch factoid QA. He indexes his corpus at different text layers (BOW, part-of-speech, dependency relations) and uses the same layers for question analysis and query creation. He optimizes the query parameters for the passage retrieval task by having a genetic algorithm apply the weights to the query terms. Tiedemann finds that the largest weights are assigned to the keywords from the BOW layer and to the keywords related to the predicted answer type (such as 'person'). The baseline approach, using only the BOW layer gives an MRR of 0.342. Using the optimized IR settings with additional layers, MRR improves to 0.406.

Quarteroni et al. [80] consider the problem of answering definition questions. They use predicate–argument structures (PAS) for improved answer ranking. They find that PAS as a stand alone representation is inferior to parse tree representations, but that together with the BOW it yields higher accuracy. Their results show a significant improvement of PAS–BOW compared to parse trees (F-scores 70.7% vs. 59.6%) but PAS makes only a very small contribution compared to BOW only (which gives an F-score of 69.3%).

[2]The reciprocal rank (RR) for a question is 1 divided by the rank ordinal of the highest ranked relevant answer. The Mean RR is obtained by averaging RR over all questions.

Recent work by Surdeanu, Ciaramita, and Zaragoza [88] addresses the problem of answer ranking for *how-to*-questions. From Yahoo! Answers[3], they extract a corpus of 140,000 answers with 40,000 questions. They investigate the usefulness of a large set of question and answer features in the ranking task. They conclude that the linguistic features "yield a small, yet statistically significant performance increase on top of the traditional BOW and *n*-gram representation" (page 726).

All above-mentioned authors conclude that the addition of structural information in QA gives a small but significant improvement compared to using a BOW-model only. For *why*-questions, we also expect to gain improvement from the addition of structural information.

4.3 Passage retrieval for why-QA using a BOW model

As explained in Section 4.1, our system comprises three modules: *question2query*, passage retrieval, and re-ranking. In the current section, we present the first two system modules, while the re-ranking module, including a description of the structural features that we consider, is presented in Section 4.5. First, however, we describe our data collection and evaluation method.

4.3.1 Data and evaluation setup

For our experiments, we use the Wikipedia INEX corpus [24]. This corpus consists of all 659,388 articles from the online Wikipedia in the summer of 2006 in XML format. We pre-processed the corpus by segmenting it in half-overlapping passages with an average length of 428 characters; these passages formed the candidate answers.

For development and testing purposes, we exploit the Webclopedia question set [39], which contains questions asked to the online QA system `answers.com`. Of these questions, 805 (5% of the total set) are *why*-questions. For 700 randomly-selected *why*-questions, we manually searched for an answer in the Wikipedia XML corpus, saving the remaining 105 questions for future testing purposes. 186 of these 700 questions have an answer in the corpus.[4] Extraction of one relevant answer for

[3] See `http://answers.yahoo.com/`

[4] Thus, about 25% of our questions have an answer in the Wikipedia corpus. The other questions are either too specific ("Why do ceiling fans turn counter-clockwise but table fans turn clockwise?") or too trivial ("Why do hotdogs come in packages of 10 and hotdog buns in packages of 8?") for the coverage of Wikipedia in 2006.

each of these questions resulted in a set of 186 *why*-questions and their reference answer.[5] Two examples illustrate the type of data we are working with:

1. "Why didn't Socrates leave Athens after he was convicted?" — "Socrates considered it hypocrisy to escape the prison: he had knowingly agreed to live under the city's laws, and this meant the possibility of being judged guilty of crimes by a large jury."

2. "Why do most cereals crackle when you add milk?" — "They are made of a sugary rice mixture which is shaped into the form of rice kernels and toasted. These kernels bubble and rise in a manner which forms very thin walls. When the cereal is exposed to milk or juices, these walls tend to collapse suddenly, creating the famous 'Snap, crackle and pop' sounds."

To be able to do fast evaluation without elaborate manual assessments, we manually created one answer pattern for each of the questions in our set. The answer pattern is a regular expression that defines which of the retrieved passages are considered a relevant answer to the input question. The first version of the answer patterns was directly based on the corresponding reference answer, but in the course of the development and evaluation process, we extended the patterns in order to cover as many as possible of the the Wikipedia passages that contain an answer. For example, for question 1 above, we developed the following answer pattern based on two variants of the correct answer that occur in the corpus: `/(Socrates.* opportunity.* escape.* Athens.* considered.* hypocrisy | leave.* run.* away.* community.* reputation)/`.[6]

In fact, answer judgment is a complex task due to the presence of multiple answer variants in the corpus. It is a time-consuming process because of the large number of candidate answers that need to be judged when long lists of answers are retrieved per question. In Chapter 6, we will come back to the assessment of relevant and irrelevant answers.

After applying our answer patterns to the passages retrieved, we count the questions that have at least one relevant answer in the top n results. This number divided by the total number of questions in a test set gives the measure *success@n*. In Section 4.3.2, we explain the levels for n that we use for evaluation. For the highest ranked relevant answer per question, we determine the reciprocal rank (RR). Questions for which the system did not retrieve an answer in the list of 150 results get an RR of 0. Over all questions, we calculate the mean reciprocal rank MRR.

[5] Just like to factoids, most *why*-questions generally have one correct answer that can be formulated in different ways.

[6] Note that the vertical bar separates the two alternatives.

4.3.2 Method and results

In the question2query-module of our system we convert the input question to a query by removing stop words[7] and punctuation, and simply list the remaining content words as query terms.

The second module of our system performs passage retrieval using off-the-shelf retrieval technology. In Khalid and Verberne (2008), we compared a number of settings for our passage retrieval task. We considered two different retrieval engines (Lemur[8] and Wumpus[9]), four different ranking models, and two types of passage segmentation: disjoint and sliding passages. In each setting, 150 results were obtained by the retrieval engine and ranked by the retrieval model. We evaluated all retrieval settings in terms of MRR@n[10] and success@n for levels $n = 10$ and $n = 150$. For the evaluation of the retrieval module, we were mainly interested in the scores for success@150 since re-ranking can only be successful if at least one relevant answer was returned by the retrieval module.

We found that the best-scoring passage retrieval setting in terms of success@150 is Lemur on an index of sliding passages with TF-IDF [126] as ranking model. We obtained the following results with this passage retrieval setting: success@150 is 78.5%, success@10 is 45.2% and MRR@150 is 0.25. We do not include the results obtained with the other retrieval settings here because the differences were small.

The results show that for 21.5% of the questions in our set, no answer was retrieved in the top-150 results. We attempted to increase this coverage by retrieving 250 or 500 answers per question but this barely increased the success score at maximum n. The main problems for the questions that we miss are infamous retrieval problems such as the vocabulary gap between a question and its answer. E.g. The answer to "Why do chefs wear funny hats?" contains none of the words from the question.

[7] To this end we use the stop word list that can be found at `http://marlodge.supanet.com/museum/funcword.html` We use all items except the numbers and the word *why*.

[8] Lemur is an open source toolkit for information retrieval that provides flexible support for different types of retrieval models. See `http://www.lemurproject.org`

[9] Wumpus is an information retrieval system mainly geared at XML retrieval. See `http://www.wumpus-search.org/`

[10] Note that MRR is often used without the explicit cut-off point (n). We add it to clarify that RR is 0 for the questions without a correct answer in the top-n.

4.4 The strengths and weaknesses of the BOW model

In order to understand how answer ranking is executed by the passage retrieval module, we first take a closer look at the TF-IDF algorithm as it has been implemented in Lemur. TF-IDF is a pure BOW model: both the query and the passages in the corpus are represented by the term frequencies (numbers of occurrences) for each of the words they contain. The terms are weighted using their inverse document frequency (IDF), which puts a higher weight on terms that occur in few passages than on terms that occur in many passages. The term frequency (TF) functions for the query and the document, and the parameter values chosen for these functions in Lemur can be found in Zhai [126].

As explained in the previous section, we consider success@150 to be the most important measure for the retrieval module of our system. However, for the system as a whole, success@10 is a more important evaluation measure. This is because users tend to pay much more attention to the top 10 results of a retrieval system than to results that are ranked lower [45]. Therefore, it is interesting to investigate which questions are answered in the top 150 and not in the top 10 by our passage retrieval module. This is the set of questions for which the BOW model is not effective enough and additional (more specific) overlap information is needed for ranking a relevant answer in the top 10.

We analyzed the set of questions that get a relevant answer at a rank between 10 and 150 (62 questions), which below we will refer to as our *focus set*. We compared our focus set to the questions for which a relevant answer is in the top 10 (84 questions). Although these numbers are too small to do a quantitative error analysis, a qualitative analysis provides valuable insights into the strengths and weaknesses of a BOW representation such as TF-IDF. Below (Sections 4.4.1 to 4.4.4), we discuss four different aspects of *why*-questions that present problems to the BOW model.

4.4.1 Short questions

Ten questions in our focus set contain only one or two content words. We can see the effect of short queries if we compare three questions that contain only one semantically rich content word.[11] The rank of the highest ranked relevant answer is given between parentheses; the last of these three questions is in our focus set.

1. Why do people hiccup? (2)
2. Why do people sneeze? (4)
3. Why do we dream? (76)

[11] The word *people* in subject position is a semantically poor content word.

We found that the rank of the relevant answer is related to the corpus frequency of the single semantically rich word, which is 64 for *hiccup*, 220 for *sneeze* and 13,458 for *dream*. This means that many passages are retrieved for question 3, making the chances for the relevant answer to be ranked in the top 10 smaller. One way to overcome the problem of long result lists for short queries is by adding words to the query that make it more specific. In the case of *why*-QA, we know that we are not simply searching for information on *dreaming* but for an *explanation* for *dreaming*. Thus, in the ranking process, we can extend the query with explanatory cue words such as *because*.[12] We expect that the addition of explanatory cue phrases will give an improvement in ranking performance.

4.4.2 The document context of the answer

There are many cases where the context of the candidate answer gives useful information. Consider for example the question *Why does a snake flick out its tongue?*, the correct answer to which was ranked 29. A human searcher expects to find the answer in a Wikipedia article about *snakes*. Within the *Snake* article he or she may search for the words *flick* and/or *tongue* in order to find the answer. This suggests that in some cases there is a direct relation between a specific part of the question and the context (document and/or section) of the candidate answer. In cases like this, the answer document and the question apparently share the same topic (*snake*). By analogy with linguistically motivated approaches to factoid QA [28] we introduce the term *question focus* for this topic.

In the example question *flick* is the word with the lowest corpus frequency (556), followed by *tongue* (4925) and *snake* (6809). Using a BOW approach to document title matching, candidate answers from documents with *flick* or *tongue* in their title would be ranked higher than answers from documents with *snake* in their title. Thus, for questions for which there is overlap between the question focus and the title of the answer documents (two thirds of the questions in our set), we can improve the ranking of candidate answers by correctly predicting the question focus. In Section 4.5.1, we make concrete suggestions for achieving this.

4.4.3 Multi-word terms

A very important characteristic of the BOW model is that words are considered separate terms. One of the consequences is that multi-word terms such as multi-word noun phrases (mwNPs) are not treated as a single term. Below, three examples of

[12] The addition of cue words can also be considered to be applied in the retrieval step. We come back to this in Section 4.6.3.

questions are shown in which the subject is realized by a mwNP (underlined in the examples; the rank of the relevant answer between brackets).

1. Why are hush puppies called hush puppies? (1)
2. Why is the coral reef disappearing? (29)
3. Why is a black hole black? (31)

We investigated the corpus frequencies for the separate parts of each mwNP. We found that these are quite high for *coral* (3316) and *reef* (2597) compared to the corpus frequency of the phrase *coral reef* (365). The numbers are even more extreme for *black* (103550) and *hole* (9734) versus *black hole* (1913). On the other hand, the answer to the *hush puppies* question can more easily be ranked because the corpus frequencies for the separate terms *hush* (594) and *puppies* (361) are relatively low. This shows that multi-word terms do not necessarily give problems for the BOW model as long as the document frequencies for the constituent words are relatively low. If (one of) the words in the phrase is/are frequent, it is very difficult to rank the relevant answer high in the result list with use of word overlap only.

36 of the 62 questions in our focus set contain a mwNP. For these questions, we can expect improved ranking from the addition of NPs to our feature set.

4.4.4 Syntactic structure

The BOW model does not take into account sentence structure. The potential importance of sentence structure for improved ranking can be exemplified by the following two questions from our set. Note that both examples contain a subordinate clause (finite or non-finite):

1. Why do baking soda and vinegar explode when you mix them together? (4)
2. Why are there 72 points to the inch when discussing fonts and printing? (36)

In both cases, the contents of the subordinate clause is less important to the goal of the question than the contents of the main clause. In the first example, this is (coincidentally) reflected by the corpus frequencies of the words in both clauses: *mix* (12724) and *together* (83677) have high corpus frequencies compared to *baking* (832), *soda* (1620), *vinegar* (871) and *explode* (1285). As a result, the reference answer containing these terms is ranked in the top-10 by TF-IDF. In the second example however, the corpus frequencies do not reflect the importance of the terms. *Fonts* and *printing* have lower corpus frequencies (1243 and 6978 respectively) than *points* (43280) and *inch* (10046). Thus, *Fonts* and *printing* are weighted heavier by TF-IDF while these terms are only peripheral to the goal of the query, the core of which is *Why are there 72 points to the inch?* This cannot be derived from the corpus

frequencies, but can only be inferred from the syntactic function (adverbial) of *when discussing fonts and printing* in the question.

Thus, the lack of information about sentence structure in the BOW model does not necessarily give rise to problems as long as the importance of the question terms is reflected by their frequency counts. If term importance does not align with corpus frequency, grammatical structure becomes potentially useful. Therefore, we expect that syntactic structure can make a contribution to cases where the importance of the terms is not reflected by their corpus frequencies but can be derived from their syntactic function.

4.4.5 What can we expect from structural information?

In Sections 4.4.1 to 4.4.4 we discussed four aspects of *why*-questions that are problematic to the BOW model. We expect contributions from the inclusion of information on cue phrases, question focus and the document context of the answer, noun phrases, and the syntactic structure of the question. We think that it is possible to achieve improved ranking performance if features based on structural overlap are taken into account instead of global overlap information.

4.5 Adding overlap-based structural information

From our analyses in Section 4.4, we found a number of question and answer aspects that are potentially useful for improving the ranking performance of our system. In this section, we present the re-ranking module of our system. We define a feature set that is inspired by the findings from Section 4.4 and aims to find out which structural features of a question–answer pair contribute the most to better answer ranking. We aim to weigh these features in such a way that we can optimize ranking performance. The input data for our re-ranking experiments is the output of the passage retrieval module. A success@150 score of 78.5% for passage retrieval (see Section 4.3.2) means that the maximum success@10 score that we can achieve by re-ranking is 78.5%.

4.5.1 Features for re-ranking

The first feature in our re-ranking method is the score that was assigned to a candidate answer by Lemur/TF-IDF in the retrieval module (f0). In the following subsections we introduce the other features that we implemented. Each feature represents the

overlap between two item bags[13]: a bag of question items (for example: all question's noun phrases, or the question's main verb) and a bag of answer items (for example: all answer words, or all verbs in the answer). The value that is assigned to a feature is a function of the overlap between these two bags. We used the following overlap function:

$$S(Q,A) = \frac{Q_A + A_Q}{Q + A} \tag{4.1}$$

in which Q_A is the number of question items that occur at least once in the bag of answer items, A_Q is the number of answer items that occur at least once in the bag of question items, and $Q + A$ is the number of items in both bags of items joined together.

The syntactic structure of the question

In Section 4.4.4, we argued that some syntactic parts of the question may be more important for answer ranking than other. Since we have no quantitative evidence yet which syntactic parts of the question are the most important, we created overlap features for each of the following question parts: phrase heads (f1), phrase modifiers (f2); the subject (f3), main verb (f4), nominal predicate (f5) and direct object (f6) of the main clause; and all noun phrases (f11). For each of these question parts, we calculated its word overlap with the bag of all answer words. For the features f3 to f6, we added a variant where as answer items only words/phrases with the same syntactic function as the question token were included (f7, f8, f9, f10).

Consider for example question 1 from Section 4.3.1: "Why didn't Socrates leave Athens after he was convicted?", and the reference answer as the candidate answer for which we are determining the feature values: "Socrates considered it hypocrisy to escape the prison: he had knowingly agreed to live under the city's laws, and this meant the possibility of being judged guilty of crimes by a large jury."

From the parser output, our feature extraction script extracts *Socrates* as subject, *leave* as main verb and *Athens* as direct object. Neither *leave* nor *Athens* occur in the answer passage, thus f4, f6, f8 and f10 are all given a value of 0. So are f5 and f9, because the question has no nominal predicate. For the subject *Socrates*, our script finds that it occurs once in the bag of answer words. The overlap count for the feature f3 is thus calculated as $\frac{1+1}{1+18} = 0.105$.[14] For the feature f7, our script extracts

[13] Note that a 'bag' is a set in which duplicates are counted as distinct items.

[14] The bag of question subjects contains one item (*Socrates*, the 1 in the denominator) and one item from this bag occurs in the bag of answer words (the left 1 in the numerator). Without stopwords, the bag of all answer words contains 18 items, one of which occurs in the bag of question subjects (the right 1 in the numerator).

the grammatical subjects *Socrates*, *he* and *this* from the parser's representation of the answer passage. Since the bag of answer subjects for f7 contains three items, the overlap is calculated as $\frac{1+1}{1+3} = 0.5$.

The semantic structure of the question

In Section 4.4.2, we saw that often there is a link between the question focus and the title of the document in which the reference answer is found. In those cases, the answer document and the question share the same topic. For most questions, the focus is the syntactic subject: "Why do cats sleep so much?". Judging from our data, there are two exceptions to this general rule: (1) If the subject is semantically poor, the question focus is the (verbal or nominal) predicate: "Why do people sneeze?", and (2) in case of etymology questions (which cover about 10% of *why*-questions), the focus is the subject complement of the passive sentence: "Why are chicken wings called Buffalo Wings?"

We included a feature (f12) for matching words from the question focus to words from the document title and a feature (f13) for the relation between question focus words and all answer words. We also include a feature (f14) for the other, non-focus question words.

The document context of the answer

Not only the document title in relation to the question focus is potentially useful for answer ranking, but also other aspects of the answer context. We include four answer context features in our feature set: overlap between the question words and the title of the Wikipedia document (f15), overlap between question words and the heading of the answer section (f16), the relative position of the answer passage in the document (f17), and overlap between a fixed set of words that we selected as explanatory cues when they occur in a section heading and the set of words that occur in the section heading of the passage (f18).[15]

Synonyms

For each of the features f1 to f10 and f12 to f16 we add an alternative feature (f19 to f34) covering the set of all WordNet synonyms for all question terms in the original

[15] We found these section heading cues by extracting all section headings from the Wikipedia corpus, sorting them by frequency, and then manually marking those section heading words that we expect to occur with explanatory sections. The result is a small set of heading cues (*history*, *origin*, *origins*, *background*, *etymology*, *name*, *source*, *sources*) that is independent of the test set we work with.

feature. For synonyms, we apply a variant of Equation 1 in which Q_A is interpreted as the number of question items that have at least one synonym in the bag of answer items and A_Q as the number of answer items that occur in at least one of the synonym sets of the question items.

WordNet Relatedness

Additionally, we included a feature representing the relatedness between the question and the candidate answer using the WordNet Relatedness tool [76] (f35). As measure of relatedness, we choose the Lesk measure, which incorporates information from WordNet glosses.

Cue phrases

Finally, as proposed in Section 4.4.1, we added a closed set of cue phrases that are used to introduce an explanation (f36). We found these explanatory phrases in a way that is commonly used for finding answer cues and that is independent of the our own set of question–answer pairs. We queried the key answer words to the most frequent *why*-question on the web "Why is the sky blue?"("blue sky rayleigh scattering") to the MSN Search engine[16] and crawled the first 250 answer fragments that are retrieved by the engine. From these, we manually extracted all phrases that introduce the explanation. This led to a set of 47 cue phrases such as *because*, *as a result of*, *which explains why*, etc.

4.5.2 Extracting feature values from the data

For the majority of features we needed the syntactic structure of the input question, and for some of the features also of the answer. We experimented with two different syntactic parsers for these tasks: the Charniak parser [21] and a development version of the Pelican parser.[17] Of these, Pelican has a more detailed descriptive model and gives better accuracy but Charniak is at present more robust for parsing long sentences and large amounts of text. We parsed the questions with Pelican because we need accurate parsings in order to correctly extract all constituents. We parsed all answers (186 times 150 passages) with Charniak because of its speed and robustness.

For feature extraction, we used the following external components: A stop word list,[18] the sets of cue phrases as described in Sections 4.5.1 and 4.5.1, the CELEX

[16] See `http://www.live.com`

[17] See `http://lands.let.ru.nl/projects/pelican/`

[18] See Section 4.3.1

Lemma lexicon [3], the WordNet synonym sets [27], the WordNet Similarity tool [76] and a list of pronouns and semantically poor nouns.[19] We used a Perl script for extracting feature values for each question–answer pair. For each feature, the script composes the required bags of question items and answer items. All words are lowercased and punctuation is removed. For terms in the question set that consist of multiple words (for example, a multi-word subject), spaces are replaced by underscores before stop words are removed from the question and the answer. Then the script calculates the similarity between the two sets for each feature following Equation 1.[20]

Whether or not to lemmatize the terms before matching them is open to debate. In the literature, there is some discussion on the benefit of lemmatization for question answering [7]. Lemmatization can especially be problematic in the case of proper names (which are not always recognizable by capitalization). Therefore, we decided only to lemmatize verbs (for features f4 and f8) in the current version of our system.

4.5.3 Re-ranking method

Feature extraction led to a vector consisting of 37 feature values for each of the 27,900 items in the data set. We normalized the feature values over all 150 answer candidates for the same question to a number between 0 and 1 using the L1 vector norm. Each instance (representing one question–answer pair) was automatically labeled 1 if the candidate answer matched the answer pattern for the question and 0 if it did not. On average, a *why*-question had 1.6 correct answers among the set of 150 candidate answers.

In the process of training our re-ranking module, we aim at combining the 37 features in a ranking function that is used for re-ordering the set of candidate answers. The task of finding the optimal ranking function for ranking a set of items is referred to as 'learning to rank' in the information retrieval literature [56]. In Chapter 5, we compare several machine learning techniques for our learning-to-rank problem. We evaluated the results using 5-fold cross validation on the question set.

4.5.4 Results from re-ranking

The results for the complete system compared with passage retrieval with Lemur/TF-IDF only are in Table 4.1. We show the results in terms of success@10, success@150 and MRR@150. We only present the results obtained using one of the

[19] Semantically poor nouns that we came across in our data set are the nouns *humans* and *people*.

[20] A multi-word term from the question is counted as one item.

best-performing learning-to-rank techniques: logistic regression.[21] A more detailed description of our machine learning method and a discussion of the results obtained with other learning techniques can be found in Chapter 5.

Table 4.1: Results for the *why*-QA system: the complete system including re-ranking compared against plain Lemur/TF-IDF for 187 *why*-questions

	Success@10	Success@150	MRR@150
Lemur/TF-IDF–sliding	45.2%	78.5%	0.25
TF-IDF + Re-ranking w/ 37 struct. feats	57.0%	78.5%	0.34

After applying our re-ranking module, we found a significant improvement over bare TF-IDF in terms of success@10 and MRR@150 ($z = -4.29, P < 0.0001$ using the Wilcoxon Signed-Rank test for paired reciprocal ranks).

4.5.5 Which features made the improvement?

In order to evaluate the importance of our features, we rank them according to the coefficient that was assigned to them in the logistic regression model (See Table 4.2). We only consider features that are significant at the $P = 0.05$ level. We find that all eight significant features are among the top nine of features with the highest coefficient.

Table 4.2: Features that significantly contribute to the re-ranking score ($P < 0.05$), ranked by their coefficient in the logistic regression model (representing their importance). Asterisks on coefficients denote the level of significance for the feature: ** means $P < 0.001$, * means $0.001 < P < 0.01$, no asterisk means $0.01 < P < 0.05$.

Feature	Coefficient
TF-IDF (f0)	0.39**
Overlap between question focus synonyms and document title (f30)	0.25**
Overlap between question object synonyms and answer words (f28)	0.22
Overlap between question object and answer objects (f10)	0.18*
Overlap between question words and document title synonyms (f33)	0.17
Overlap between question verb synonyms and answer words (f24)	0.16
WordNet Relatedness (f35)	0.16*
Cue phrases (f36)	0.15*

The feature ranking is discussed in Section 4.6.1.

[21] We used the *lrm* function from the Design package in *R* (`http://cran.r-project.org/web/packages/Design`) for training and evaluating models based on logistic regression.

4.6 Discussion

In the following subsections, we discuss the feature ranking (Section 4.6.1), make a comparison to other re-ranking approaches (Section 4.6.2) and explain the attempts that we made for solving the remaining problems (Section 4.6.3).

4.6.1 Discussion of the feature ranking

Table 4.2 shows that only a small subset (eight) of our 37 features significantly contribute to the re-ranking score. The highest ranked feature is TF-IDF (the bag of words), which is not surprising since TF-IDF alone already reaches an MRR@150 of 0.25 (see Section 4.3.2). In Section 4.4.5, we predicted a valuable contribution from the addition of cue phrases, question focus, noun phrases and the document context of the answer. This is partly confirmed by Table 4.2, which shows that among the significant features are the feature that links question focus to document title and the cue phrases feature. The noun phrases feature (f11) is actually in the top nine features with the highest coefficient but its contribution was not significant at the 0.05 level ($P = 0.068$).

The importance of question focus for *why*-QA is especially interesting because it is a question feature that is specific to *why*-questions and does not similarly apply to factoids or other question types. Moreover, the link from the question focus to the document title shows that Wikipedia as an answer source can provide QA systems with more information than a collection of plain texts with less discriminative document titles does.

The significance of cue phrases is also an important finding. In fact, including cue phrases in the *why*-QA process is the only feasible way of specifying which passages are likely to contain an explanation (i.e. an answer to a *why*-question). In Chapter 3, we pointed out that higher-level annotation such as discourse structure can give useful information in the *why*-answer selection process. However, the development of systems that incorporate discourse structure suffers from the lack of tools for automated annotation. The current results show that surface patterns (the literal presence of items from a fixed set of cue words) are a step in the direction of answer selection.

The significant features in Table 4.2 also show us which question constituents are the most salient for answer ranking: focus, main verb and direct object. We think that features incorporating the question's subject are not found to be significant because in a subset of the questions, the subject is semantically poor. Moreover, since for most questions the subject is the question focus, the subject features and

the focus features are correlated. In our data, the question focus apparently is the more powerful predictor.

4.6.2 Comparison to other approaches

The 23% improvement that we reach in terms of MRR@150 (from 0.25 to 0.34) is comparable to that reached by Tiedemann in his work on improving factoid QA with use of structural information.

In order to see whether the improvement that we achieved with re-ranking is on the account of structural information or just the benefit of using word sequences, we experimented with a set of re-ranking features based on sequences of question words that are not syntactically defined. In this re-ranking experiment, we included TF-IDF, word bigrams and word trigrams as features. The resulting performance was around baseline level (MRR=0.25), significantly worse than re-ranking with structural overlap features. This is still true if we add the cue word feature (which, in isolation, only gives a small improvement to baseline performance) to the n-gram features.

4.6.3 Solving the remaining problems

Although the results in terms of success@10 and MRR@150 are satisfactory, there is still a substantial proportion of *why*-questions that is not answered in the top 10 result list. In this section, we discuss a number of attempts that we made to further improve our system.

First, after we found that for some question parts synonym expansion leads to improvement (especially the main verb and direct object), we experimented with the addition of synonyms for these constituents in the retrieval step of our system (Lemur). We found, however, that it does not improve the results due to the large synonym sets of many verbs and nouns which add much noise and lead to very long queries. The same holds for the addition of cue words in the retrieval step.

Second, although our re-ranking module incorporates expansion to synonym sets, there are many question–answer pairs where the vocabulary gap between the question and the answer is still a problem. There are cases where semantically related terms in the question and the answer are of different word classes (e.g. *hibernate—hibernation*), and there are cases of proper nouns that are not covered by WordNet (e.g. *B.B. King*). We considered using dynamic stemming for verb–noun relations such as the *hibernation*-case but research has shown that stemming hurts as many queries as it helps [7]. Therefore, we experimented with a number of different

semantic resources, i.e. the nominalization dictionary Nomlex [63] and the wikiOntology by Ponzetto and Strube [77]. However, in their current state of development these semantic resources cannot improve our system because their coverage is too low to make a contribution to our re-ranking module. Moreover, the present version of the wikiOntology is very noisy and requires a large amount of cleaning up and filtering.

Third, we considered that the use of cue phrases may not be sophisticated enough for finding explanatory relations between question and answer. Therefore, we experimented with the addition of cause–effect pairs from the English version of the EDR Concept Dictionary [123] — as suggested by Higashinaka and Isozaki [37]. Unfortunately, the list appeared to be extremely noisy, proving it not useful as source for answer ranking.

4.7 Conclusions and directions for future research

In this chapter, we extended a passage retrieval system for *why*-QA using off-the-shelf retrieval technology (Lemur/TF-IDF) with a re-ranking step incorporating structural information. We get significantly higher scores in terms of MRR@150 (from 0.25 to 0.34) and Success@10. The 23% improvement that we reach in terms of MRR is comparable to that reached on various other QA tasks by other researchers in the field (see Section 4.6.3). This confirms our hypothesis in Section 4.1 that for the relatively complex problem of *why*-QA, a significant improvement can be gained by the addition of structural information to the ranking component of the QA system.

Most of the features that we implemented for answer re-ranking are based on word overlap between part of the question and part of the answer. As a result of this set-up, our features identify the parts of *why*-questions and their candidate answers that are the most powerful/effective for ranking the answers. The question constituents that appear to be the most important are the question focus, the main verb and the direct object. On the answer side, most important are the title of the document in which the candidate answer is embedded and knowledge on the presence of cue phrases.

Since our features are overlap-based, they are relatively easy to implement. For implementation of some of the significant features, a form of syntactic parsing is needed that can identify subject, verb and direct object from the question and sentences in the candidate answers. An additional set of rules is needed for finding the question focus. Finally, we need a fixed list for identifying cue phrases. Exploiting the title of answer documents in the feature set is only feasible if the documents that may contain the answers have titles and section headings similar to Wikipedia.

In conclusion, we developed a method for significantly improving a BOW-based approach to *why*-QA that can be implemented without extensive semantic knowledge sources. Our series of experiments suggest that we have reached a maximum performance that can be obtained using a knowledge-poor approach. Experiments with more complex types of information — discourse structure, cause–effect relations — show that these information sources have not as yet developed sufficiently to be exploited in a QA system.

In the next chapter, we optimize the ranking component of our system by evaluating a number of techniques for the learning-to-rank task.

5

Learning to Rank Answers to Why-Questions

Edited from: Suzan Verberne, Hans van Halteren, Daphne Theijssen, Stephan Raaijmakers, Lou Boves. Learning to Rank for *Why*-Question Answering. Submitted to *Information Retrieval*.

Abstract

In this chapter, we evaluate a number of machine learning techniques for the task of ranking answers to *why*-questions. We use a set of 37 linguistically motivated features that characterize questions and answers. We experiment with a number of machine learning techniques (among which several classifiers and regression techniques, Ranking SVM and SVM^{map}) in various settings. The purpose of the experiments is to assess how the different machine learning approaches can cope with our highly imbalanced binary relevance data, with and without hyperparameter tuning. We find that with all machine learning techniques, we can obtain an MRR score that is significantly above the TF-IDF baseline of 0.25 and not significantly lower than the best score of 0.35. Because of their relatively light-weight implementation, regression techniques seem the best option for our learning problem.

5.1 Introduction

In Chapter 4, we described a system for *why*-QA that consists of an off-the-shelf passage retrieval engine (Lemur[1]), and a ranking module that uses a set of features extracted from the question and each of the candidate answers. Until now, we have mainly focused on improving the ranking performance of our system by adapting and expanding the feature set used for ranking. This has led to a set of 37, mostly linguistically motivated, features representing the degree of overlap between a question and each of its candidate answers. We have experimented with a genetic algorithm and with logistic regression for finding the optimal weights for combining the 37 features [106, 107].

In this chapter, we aim at finding the optimal ranking function for our feature set to be applied in the ranking module to the set of candidate answers. We evaluate a number of learning-to-rank techniques [56] in their ability of ranking the answers in our data set. The problem of answering *why*-questions is of interest because the length and the complexity of the answers make it an interesting case study for answer ranking with the use of linguistically motivated features.

The problem of learning to rank has gained attention in the field of Information Retrieval (IR) since 2005. It has been boosted by the ongoing development of the LETOR benchmark data set [57]. Until now, most learning-to-rank research has been directed at developing new techniques and evaluating them on the LETOR data collections. This has resulted in a good understanding of the performance of a range of ranking techniques for this specific data set. However, it is not yet known to what extent their performances will change for other data sets. This work is a step towards understanding to what extent the results do generalize to other data and applications.

Learning-to-rank experiments are meaningful for applications that produce a ranked list of items (documents, entities, answers, etc.) that are described by a set of features and an class label according to which they can be ranked. In IR applications, the class label refers to the item's relevance. In the case of QA, relevance is generally defined as a binary variable [117]. On the other hand, all operational QA systems still present a ranked list of answer candidates for each individual input question [85]. For our system for *why*-QA, we also use binary relevance labeling while aiming at a ranked result list. Although we found that it is to some extent possible to label the answers to *why*-questions on a multi-level relevance scale, we decided to treat answer relevance as a binary variable (see Section 5.3.3). This means that our ranking

[1] See `http://www.lemurproject.org`

function needs to induce a ranked list from binary relevance judgments.[2]

A second challenge that we face in learning to rank our data is the imbalance between positive and negative instances in the training set: There tend to be much more incorrect than correct answers [95]. This is not unique for QA data (in document retrieval, for example, the number of irrelevant documents is also much larger than that of relevant ones) but we will see that this imbalance plays a role in ranking the answers in our data collection.

We evaluate the following techniques for the task of learning a ranking for *why*-answers: Naive Bayes, Support Vector Classification, Support Vector Regression, Logistic Regression, Ranking SVM, SVM^{map} and a Genetic Algorithm. Following the learning-to-rank literature [56], we consider three different approaches to learning to rank: (1) the so-called pointwise approach, in which candidate answers are classified individually (over all questions), (2) the pairwise approach, in pairs of two candidate answers to the same question are classified, and (3) the listwise approach, in which the complete ranking of all candidate answers to the same question is optimized.[3] We will discuss the performance of each of these three approaches on our data while evaluating the machine learning techniques mentioned above. Some of these techniques require tuning of hyperparameters, others do not. We will pay special attention to the effects of data imbalance and hyperparameter tuning in the performance of the techniques.

This chapter is organized as follows: in Section 5.2, we discuss related work on QA research, learning to rank and the problem of imbalanced data. In Section 5.3 we describe the resources that we use for our experiments and we specify the characteristics of the data used in our experiments. An overview of the experiments that we conducted is in Section 5.4. The results are presented and discussed in Section 5.5. Section 5.6 contains our conclusions.

5.2 Related work

In Section 5.2.1, we give a summary of QA research, the role of *why*-questions in QA and learning-to-rank experiments for the purpose of QA. In Section 5.2.2, we present a brief overview of the learning-to-rank approaches in the literature. In Section 5.2.3, we discuss the challenge of classifying imbalanced data, which is an

[2] In ranking binary relevance data, the goal is to rank the correct answers higher than the incorrect answers. There is no evaluation of the ranking among the (in)correct answers themselves.

[3] We will use the term 'answer cluster' to refer to the set of candidate answers to one question. Here, we assume a set-up in which a list of candidate answers is retrieved by a retrieval engine. Learning to rank is the task of learning the optimal ranking for the answers within each cluster.

important aspect of our learning problem.

5.2.1 Question answering and the position of why-questions

QA research emerged in the field of Information Retrieval in the mid-1990. From 1999 to 2007, the TREC-QA track[4] has encouraged the development and evaluation of open-domain QA systems, with the use of common evaluation measures. In the first years of the TREC-QA track, different question types were included in one and the same task. The 1999 QA track contained 200 questions, only two of which were *why*-questions; all other questions were factoids (asking after *who*, *where*, *how many*, etc.) [110]. From 2002 onwards, *why*-questions were no longer included in the track's main task [113].

According to the 2002 overview paper by Maybury [62], *why*-questions are one of the most complex question types. This is mainly because the answers to *why*-questions are not named entities (which are in general clearly identifiable), but text passages giving a (possibly implicit) explanation [104]. Recently, research by Higashinaka and Isozaki has been directed at developing and evaluating QA systems for answering Japanese *why*-questions (*why*-QA) [37]. For English *why*-questions, we have developed an approach that combines bag-of-words retrieval techniques with linguistic and structural knowledge (see Chapter 4). This chapter will continue this work with learning-to-rank experiments for our set of structural and linguistic features.

Until now, not much research has been directed at learning-to-rank experiments for the purpose of optimizing QA systems. Usunier et al. [95] are the first to apply learning-to-rank techniques to QA data: they experiment with AdaBoost and RankBoost on a set of 250 questions. Surdeanu et al. [88] adopted the Ranking Perceptron approach of Shen and Joshi [85] for learning to rank in the context of a large QA collection.

5.2.2 Learning to Rank approaches

Most approaches to learning to rank consider the problem as a case of supervised learning. All instances (the items to be ranked) are assigned a (binary or ordinal) score representing their relevance as determined by an independent judgment process; this score is considered as the ground truth. In the training stage, a ranking function is learned based on the set of feature vectors with their ground truth labels. In the testing stage, the function is applied to new sets of items in order to gener-

[4] See `http://trec.nist.gov/data/qamain.html`

ate a ranked order. Learning a ranking function is not a trivial task. In [56], many approaches to learning to rank are discussed.

Approaches to learning a ranking from a set of labeled instances can be divided in three categories: (1) learning to classify the instances according to their (binary or ordinal) label irrespective of the clustering of the answers[5] (pointwise approach), (2) classifying pairs of correct and incorrect answers for their mutual order and optimize the proportion of correctly ordered pairs (pairwise approach), or (3) optimizing a cost function for the ordering of answers within one answer cluster (listwise approach). In the remainder of this section, we discuss the three approaches in more detail.

The pointwise approach

In the pointwise approach, the clustering of answers per question is ignored in the training stage: The task of the classifier is to learn whether an instance should be classified as relevant or irrelevant, irrespective of the answer cluster it belongs to. Relations between the candidate answers to the same question are ignored. A ranked answer list can then be induced by letting the classifier assign a score to each instance in the test set, expressing the probability that it should be classified as relevant, and then ordering the answers per question according to these scores (ordinal sort) [14]. Techniques that can be applied in this approach are classifiers (such as Naive Bayes and Support Vector Classification) and regression techniques (such as Logistic Regression and Support Vector Regression) [22, 36].

In the literature, the pointwise approach is considered the weakest of the three learning-to-rank approaches, because it ignores the clustering of instances per query. This especially leads to problems in situations where the number of answers varies largely for different queries.[6] Moreover, pointwise approaches do not take into account the position of each answer in a ranked list. As a result of that, candidate answers that are in the bottom part of the result list receive the same attention as the top-ranked candidates while they are relatively less important for the overall system performance [56].

The pairwise approach

An alternative way of learning a ranking for a list of answers is to classify pairs of relevant and irrelevant answers within one cluster for their mutual order and optimizing the proportion of correctly ordered answers. This learning principle is called

[5] We use the word 'answers' here because that is the type of data we are working with in our QA experiments. In learning-to-rank experiments, 'answers' can be any type of items to be ranked.

[6] This type of imbalance was not relevant in our study since we ranked 150 answers for all questions.

'pairwise preference learning', and was introduced by Joachims [44], who proposed the learning algorithm Ranking SVM based on this principle. Other pairwise algorithms are RankNet [14] and RankBoost [31]. Pairwise preference learning is studied in more detail in [32] and is applied to several ranking problems such as combining rankings from multiple retrieval systems in [19].

Pairwise approaches are considered more powerful than pointwise approaches because they consider pairs of instances from the same cluster and do not take into account unrelated instances (answers to other queries). Furthermore, pairwise approaches tend to give better results on the LETOR benchmark data than pointwise approaches, although the differences are small and not always significant [56].[7]

The listwise approach

The third, more recently developed approach to learning to rank answers is the listwise approach, in which a cost function for the ordering of answers within one answer cluster is optimized [120]. There are two subtypes of listwise approaches [56]. The first takes into account the relevance labels of all instances within one cluster and optimizes the instance order for an IR evaluation measure such as Mean Average Precision (MAP), Mean Reciprocal Rank (MRR) or Normalized Discount Cumulative Gain (nDCG). Examples of techniques that optimize for IR evaluation measures are SVM^{map} [124] and AdaRank [121]. The second type of listwise learning (e.g. ListNet [16]) takes sublists of ranked items as training data, and optimizes for the difference between this ground truth ranking and the hypothesized ranking. Listwise techniques are considered promising because they already reach scores similar to or better than pairwise techniques while they have been developed more recently [56].

In [106] we implemented a listwise ranking approach using a genetic algorithm that optimizes answer ranking for MRR. Genetic algorithms have been applied to learning-to-rank problems and other retrieval optimization problems by several researchers in the field [94, 122, 93]. The ranking performance can be defined and implemented in the so-called fitness function in different ways. In [26], a number of fitness functions that are derived from ranking evaluation measures (such as MAP) are compared for their effectiveness.

5.2.3 The problem of imbalanced data

As mentioned in Section 5.1, class imbalance is a challenge in many learning-to-rank tasks, also in ranking QA data [95]. This especially holds for pointwise techniques

[7] All experimental results on the LETOR data can be downloaded from `http://research.microsoft.com/en-us/um/beijing/projects/letor/`

because the imbalance hampers the optimization process: If 98% of the instances in the training set have been labeled incorrect, then classifying all instances as incorrect gives an accuracy of 98%.

This problem has been acknowledged by many researchers in the machine learning field [42, 95, 1, 89]. Because SVMs are very popular for all sorts of classification tasks, much work on tackling the problem of imbalanced data is focused on making SVMs robust to imbalance. In the literature, three solutions for curing problematic class imbalances for classifiers are discussed: undersampling the majority class, oversampling the minority class and cost-modifying according to the same ratio as the class balance. In general, the latter approach gives the best results for various classifiers [42, 89]. In Section 5.4.2, we explain our attempts for curing the class imbalance in our data.

Class imbalance causes fewer problems for regression techniques than for classifiers. In regression models, the so-called 'intercept' value moves the outcome of the regression function towards the bias in the data. If the class imbalance is not too extreme, the intercept can be adapted so that the regression function is robust against it [73]. In Section 5.5.1, we come back to the effect of class imbalance on regression techniques.

Pairwise approaches are less sensitive to class imbalance than pointwise approaches. The reason is that they classify pairs of correct and incorrect answers from the same cluster, thereby balancing the training data. For listwise approaches, data imbalance is not an issue since these techniques are not classification-based but they optimize for instance order directly.

5.3 Data and system set-up

In this section, we present the background of our learning to rank task in terms of the resources that we use for development and evaluation (5.3.1), the set-up of our QA system (5.3.2), the ground truth labeling that we apply to the data (5.3.3) the features that we extract from the data (5.3.4), and our evaluation set-up (5.3.5).

5.3.1 Resources

For our experiments, we used the Wikipedia INEX 2006 corpus [24]. This corpus consists of all 659,388 articles extracted from the online Wikipedia in the summer of 2006, converted to XML format. Before indexing the corpus, we segmented all Wikipedia articles into passages. We used a semi-fixed passage size of 500 to 600 characters (excluding all XML markup) with an overflow to 800 for the purpose of

completing sentences.[8] We created passage overlap by starting each new passage at a sentence boundary halfway the previous passage. For Wikipedia articles that contain fewer than 500 characters in total, we included the complete text as one passage. Our segmentation process produced an index of 6,365,890 passages. We separately saved the document title and section heading as metadata for each passage because they were used in our feature set.

For our experiments we exploited the Webclopedia question set by Hovy et al. [39]. This set contains questions that were asked to the online QA system `answers.com`. Of these questions, 805 (5% of the total set) are *why*-questions. For development and testing purposes, we needed a set of questions for which we knew that they had an answer in the corpus. For 700 randomly selected *why*-questions from this set we therefore searched for an answer in the Wikipedia XML corpus by manually formulating queries and browsing through documents. Three examples illustrate the type of data we are working with:

1. "Why do most cereals crackle when you add milk?" — "They are made of a sugary rice mixture which is shaped into the form of rice kernels and toasted. These kernels bubble and rise in a manner which forms very thin walls. When the cereal is exposed to milk or juices, these walls tend to collapse suddenly, creating the famous 'Snap, crackle and pop' sounds."

2. "Why didn't Socrates leave Athens after he was convicted?" — "Socrates considered it hypocrisy to escape the prison: he had knowingly agreed to live under the city's laws, and this meant the possibility of being judged guilty of crimes by a large jury."

3. "Why was cobalt named cobalt?" — "The word cobalt comes from the German kobalt or kobold, meaning evil spirit, the metal being so called by miners, because it was poisonous and troublesome (it polluted and degraded the other mined elements, like nickel)."

For 186 of the 700 *why*-questions, we were able to find at least one correct the answer in the Wikipedia corpus.[9] Thus, our data collections consists of 186 *why*-questions.

[8] We assume that answer passages ending in an unfinished sentence are undesirable. However, if the hard maximum of 800 characters is reached, the passage is cut off between two words to prevent non-sentence contexts like tables to result in extremely long passages.

[9] Thus, about 25% of our questions have an answer in the Wikipedia corpus. For the majority of the other questions (except for some questions that seem to be jokes rather than a serious information need), the coverage of Wikipedia in 2006 appeared not to be sufficient. Chapter 6 gives a detailed analysis of Wikipedia's shortcomings for our data.

This is not very large for machine learning experiments but comparable to the data collections that are contained in the LETOR benchmark data set [79].

5.3.2 System set-up

Our system consists of three modules that are run in sequence:

1. A question processing module that transforms the input question to a query by removing stop words and punctuation.

2. An off-the-shelf retrieval module that retrieves passages of text that share content with the input query. Here, we use Lemur to retrieve 150[10] answer passages per question. The selection of these 150 passages was done on the basis of the TF-IDF weighting scheme as it has been built in in Lemur [126]. This gives us a set of 186 questions with 150 candidate answers per question.

3. A ranking module that ranks the retrieved passages using features extracted from the question and each of the 150 candidate answers (see Section 5.3.4 below): 27,900 ($186 * 150$) question-answer pairs (instances) in total. One of the features that we use is the TF-IDF score that was assigned to each candidate answer by Lemur. Finding the optimal ranking function for these data is our goal for this chapter.

5.3.3 Ground truth labeling

For training and testing machine learning techniques, each instance in the data has to be assigned a label. In IR research, these labels are relevance assessments. In the case of learning-to-rank experiments, researchers are often faced with large amounts of data that need to be labeled: a relatively small set of 100 queries with 100 results per query already gives a set of 10,000 instances. Since it is often too costly to label all instances by hand, estimations of relevance are generally used instead of complete manual judgments. These estimations can come from the aggregation of rankings by multiple systems for the same data, or by sampling a number of instances per cluster, in which case all un-annotated instances are considered irrelevant. A number of aggregation and sampling options for the estimation of relevance assessments are discussed in [2].

Thus, the large amounts of training instances force IR researchers to use estimations instead of complete manual judgments for labeling the instances. Moreover,

[10] We experimented with a higher number of answer candidates but coverage was hardly improved when increasing this number to 500.

in the case of *why*-QA, it is not possible to apply fully automatic ground truth labeling (which is often applied for evaluation in factoid-QA) because the answer to a *why*-question can have several textual variants. Therefore, we performed a form of sampling in which an assessor judged answers for each question, starting with the answer that is ranked first and stopping at the first correct answer found. We did this for several system settings, which gave different rankings and therefore different samples of assessments.

Although it is possible to judge the quality of answers to *why*-questions on a multi-level scale (ranging from partly relevant to highly relevant), we found that multi-level judgments are very subjective. Therefore, we decided to use binary relevance assessments: "Does this passage answer the question, or not?" Judgments by a second assessor on a sample of the data showed that the annotation task was relatively difficult: the two assessors agreed in 97% of the cases, but taking into account the chance agreement we reached only a moderate κ value of 0.48 (due to the highly imbalanced data). Since the second assessor only judged a sample of the data that had been annotated by the first assessor, it was not sensible to try and reach consensus on these data with two annotators. Therefore, we used the ground truth annotations of the first assessor.[11]

Because we judged a sample of all instances, we supported our manual judgments with a set of answer patterns similar to the answer patterns used in TREC: a regular expression for each question that defines which answers should be labeled as correct. These answer patterns allowed us to not simply consider all unlabeled instances as incorrect (which is generally done in the case of judgments for a sample of the data [2]) but to label some of the unlabeled instances as correct because they matched the answer pattern for the question. For example, for question 2 above ("Why didn't Socrates leave Athens after he was convicted?"), we developed the following answer pattern after assessing a sample of the candidate answers in our set: */(Socrates.* opportunity.* escape.* Athens.* considered.* hypocrisy | leave.* run.* away.* community.* reputation)/*. The pattern is based on two variants of the correct answer that we found in the set of candidate answers.[12]

[11] In Chapter 6, we will address the difficulties of manual human judgments for the evaluation of *why*-QA.

[12] Note that the vertical bar separates the two alternative formulations.

Table 5.1: Set of 37 features used in our ranking module

TF-IDF	The score that is assigned to a candidate answer by Lemur/TF-IDF in the retrieval module
14 Syntactic features	Overlap between question and answer constituents (e.g. subject, verb, question focus)
14 WordNet expansion features	Overlap between the WordNet synsets of question and answer constituents
1 Cue phrase feature	Overlap between candidate answer and a predefined set of explanatory cue phrases
6 Document structure features	Overlap between question (focus) words and document title and section heading
1 WordNet Relatedness feature	Relatedness between question and answer according to the WordNet similarity tool [76]

5.3.4 Feature extraction

In Chapter 4, we compiled a set of 37 features that are summarized in Table 5.1. We syntactically parsed the questions with the Pelican parser[13] and the candidate answers with the Charniak parser [21]. Then we used a Perl script to extract all feature values from the question, the answer candidate and both their parse trees.

Each feature represents the overlap between two item bags[14]: a bag of question items (for example: all question's noun phrases, or the question's main verb) and a bag of answer items (for example: all answer words, or all verbs in the answer). The value that is assigned to a feature is a function of the overlap between these two bags. We used the following overlap function:

$$S(Q,A) = \frac{Q_A + A_Q}{Q + A} \tag{5.1}$$

in which Q_A is the number of question items that occur at least once in the bag of answer items, A_Q is the number of answer items that occur at least once in the bag of question items, and $Q + A$ is the number of items in both bags of items joined together.

Description of the features

Below we give a summary description of the 37 features that we used for ranking (cf. Table 5.1).

[13] See `http://lands.let.ru.nl/projects/pelican/`

[14] Note that a 'bag' is a set in which duplicates are counted as distinct items.

- Syntactic features. These are features that describe the overlap between a syntactically defined question part (such as subject, verb or direct object) and the answer passage or parts of the answer passage (e.g. matching the question's verb to all verbs in the answer). The syntactic features that deserve some extra attention here, are the features related to question focus (e.g. overlap between the question focus and the title of the answer document). We introduced the term question focus in analogy to linguistically motivated approaches to factoid QA for the topic of the question ("What is the question about?"). We defined three rules for determining the focus of a *why*-question: If the subject is semantically poor (*people*, *human* or a pronoun), the question focus is the (verbal or nominal) predicate: "Why do people sneeze?". In case of etymology questions, the focus is the subject complement of the passive sentence: "Why are chicken wings called Buffalo wings?". In all other cases, the focus is the syntactic subject of the question, e.g. "Why are flamingos pink?" (see also Chapter 4).

- WordNet expansion features. For each of the syntactic overlap features, we included an additional feature that describes the overlap between the WordNet synonym set [27] of a syntactically defined question part and the answer. This allowed us to investigate the importance of WordNet expansions for specific parts of the question, instead of for all question words indistinguishably.

- Cue phrase feature. The cue phrase feature is the overlap between the bag of answer words and a fixed set of words that suggest some kind of explanation. We found the cue phrases in a way that is commonly used for finding answer templates: we queried the key answer words to the most frequent *why*-question on the web ("blue sky rayleigh scattering" for "Why is the sky blue?") to MSN's Live Search[15] and crawled the first 250 answer fragments that are retrieved by the engine. From these, we manually extracted all phrases that introduce the explanation. This led to 47 cue phrases such as *because*, *as a result of*, *which explains why*, etc.

- Document structure features. The six document structure features cover information about the document context of a candidate answer passage, such as: the overlap between the question and the title of the Wikipedia document, the overlap between the question and the title of the section in which the candidate answer occurs, and the relative position of the candidate answer in the document.

[15] See `http://www.live.com`

- WordNet Relatedness feature. We defined the relatedness between a question and an answer as the weighted average of the relatedness of each question word with all words in the answer:

$$REL(Q,A) = \frac{\sum_{q=1}^{m} \sum_{a=1}^{n} REL(w_q, w_a)}{m} \tag{5.2}$$

  in which Q,A is the question-answer pair under consideration, w_q represents the question words, w_a the answer words, m is the number of question words, and n is the number of answer words. As a measure of word relatedness ($REL(w_q, w_a)$), we chose the Lesk measure, which finds overlaps between the glosses of two words, also if they belong to different word classes [76]. We used the version of Lesk that was adapted for WordNet by Banerjee and Pedersen [4].

In Chapter 4, we found that the most valuable features for ranking candidate answers to *why*-questions in addition to TF-IDF are the overlap between the question focus and the topic of the answer document, the overlap between words in the answer and synonyms of the question's main verb and the question's direct object, the WordNet Relatedness score for the question-answer pair, and the presence of cue phrases in the answer.

Resulting feature vectors and normalization

Feature extraction led to a vector comprising 37 feature values for each of the 27,900 items in the data set. For feature value normalization, we performed a form of clusterwise normalization that is comparable to the approach by Liu et al. [57] ('QueryLevelNorm' in LETOR).

Assume a question Q_i with the candidate answers $A_j (j = 1..150)$. For each feature $F_k (k = 1..37)$, its value x_{ijk} is normalized by transforming it to its z-score:

$$x'_{ijk} = (x_{ijk} - \mu_{ik}) / \sigma_{ik} \tag{5.3}$$

in which μ_{ik} is the mean of all values of feature F_k for the candidate answers to Q_i and σ_{ik} is the standard deviation of all values of feature F_k for the candidate answers to Q_i.

Normalizing feature values to a relative value within a cluster makes our data more suitable for pointwise learning approaches. Moreover, this approach makes it possible to normalize the scores independently of the answers to other questions: It can be performed for the set of candidate answers to each new input question.

5.3.5 Evaluation set-up

Each instance in our data was labeled correct if the candidate answer was deemed a correct answer to the question and incorrect if it was not (see Section 5.3.3). On average, a *why*-question had 1.6 correct answers among the set of 150 candidate answers retrieved by Lemur. This means that the incorrect/correct ratio in our data collection is 71 to 1 (98.6% of the instances in the training set was labeled incorrect). Akbani et al [1] consider a data set to be 'highly imbalanced' for the use of classification techniques if the ratio of negative against positive instances is bigger than 50 to 1.

For evaluation, we counted the questions in the test set that have at least one correct answer in the top n ($n \in 10, 150$) of the results. This number divided by the total number of questions in our test collection gave the measure *Success@n*. For the highest ranked correct answer per question, we determined its reciprocal rank ($RR = 1/rank$). If there was no correct answer retrieved by the system at $n = 150$, the RR was 0. Over all questions, we calculated the mean RR: $MRR@150$.

We performed 5-fold cross validation on the question set. We kept the 150 answers to each question together in one fold so that we did not train and test on answers to the same question. For techniques that require tuning of hyperparameters, we used a development set (see Section 5.4.1). In the training stage, we excluded the 40 questions (21.5%) for which none of the 150 candidate answers was correct. The test set on the other hand did contain these questions, for which RR would naturally be 0.

5.4 Experiments

In this section, we describe the machine learning techniques we evaluated and how we applied each of them to our learning problem. In all cases, we used the 37-feature set with clusterwise normalization that we described in Section 5.3.4.

As a baseline, we used the system setting in which the answers are retrieved and ranked according to TF-IDF only.

5.4.1 Matrix of techniques

We compared the three learning-to-rank approaches introduced in Section 5.2.2: the pointwise approach (see Section 5.4.2), the pairwise approach (Section 5.4.3) and the listwise approach (Section 5.4.4). In the pointwise approach, we evaluated the following classification and regression techniques: Naive Bayes, Support Vector Classification, Support Vector Regression and Logistic Regression. In the pairwise ap-

proach, we evaluated the same classification and regression techniques, and Ranking SVM. For the listwise approach, we evaluated SVM^{map} and a Genetic Algorithm that optimizes for MRR.

Hyperparameter tuning

For techniques that require hyperparameter values, we not only evaluated the default hyperparameter setting but we also tried to find optimal values for the hyperparameters using a grid search over a large range of values (see Section 5.4.2 for a description of the grid we used). For hyperparameter tuning, it is necessary to use development data that is held out from the training set. We searched for hyperparameter values that give the best results in terms of MRR on the development set. Given the small number of questions in our training set,[16] we decided to hold out 10 questions with their 150 answers from each training set. Because development sets of 10 questions are quite small, we selected three (non-overlapping) development sets for each fold.

As a further measure to prevent overfitting on the development sets (in order to make the optimization process more robust), we selected three (near-)optimal hyperparameter settings for each development set, instead of simply taking the one leading to the best MRR. The three hyperparameter settings were selected as follows: The first was always the one leading to the best MRR on the development set. The second and third were the highest local optima that are further than five steps in the grid away from the first chosen point and from each other (see the descriptions of the used grids in 5.4.2).

During testing, the outputs of the nine models that were created for the three development sets (three models per development set) were combined by addition, after scaling them to a comparable range.

5.4.2 The pointwise approach

We first investigated the pointwise approach of applying classification and regression techniques to our learning problem. In the training phase, the classifier or regressor learns to classify each instance (question answer pair) as either correct or incorrect, irrespective of the cluster it belongs to.[17] In the test phase, we let the model assign a score to each instance in the data representing the likelihood that this instance should

[16] Around 120 because, as explained in Section 5.3.5, we excluded the 21% questions without correct answers and 20% for each fold to test on.

[17] Recall that we did normalize the feature values per cluster, which made our data more suitable for pointwise learning approaches.

be classified as correct. The actual ranking is done by a script that sorts the instances per cluster by the output score of the classifier.

As discussed in Section 5.3.5, our data show a strong imbalance between positive and negative instance, with a incorrect/correct ratio in the training set of 71. This may cause problems for machine learning techniques that are designed for classification. Therefore, we applied a balancing strategy to all classification and regression techniques that we evaluated. As observed in the literature [42, 89], application of a cost factor is the preferred approach to counter imbalance. If a system did not allow for this, we applied oversampling of the positive instances in such a way that each training set included approximately as many positive as negative instances.

In the pointwise approach, we trained and tested each machine learning technique both on the original (imbalanced) data and on the data that was balanced first (by applying a cost factor or oversampling). We performed hyperparameter optimization for both these data variants. This led to four different settings per machine learning technique: original default, original tuned, balanced default, and balanced tuned.

Naive Bayes classifier (NB)

For experiments with Naive Bayes (NB), we used the `e1071` package in R.[18] This package does not allow for tuning of hyperparameters for Naive Bayes so we only ran the Naive Bayes classifier in its default setting, on both the original and the oversampled data.

Support Vector Classification (SVC) and Support Vector Regression (SVR)

For standard support vector methods, we used LIBSVM.[19] As proposed by the authors of LIBSVM, we first scaled our data using *svm-scale*. We experimented with support vector classification (C-SVC) and support vector regression (ε-SVR). For both, we used the RBF kernel [40].

The RBF kernel expects two hyperparameters: c — the trade-off between training error and margin, and γ — a multiplication factor determining the range of kernel space vector norms. Their default values are $c = 1$ and $\gamma = 1/k$ (with k being the number of features, giving a γ of 0.027 for our data). For the grid search, we

[18] See `http://cran.r-project.org/web/packages/e1071/index.html`

[19] See `http://www.csie.ntu.edu.tw/~cjlin/libsvm`

followed the suggestion in [40] to use exponentially growing sequences of c and γ. We varied c from 2^{-13} to 2^{13} and γ from 2^{-13} to 2^{7} in steps of $\times 4$. [20]

SVC allows us to use a cost factor for training errors on positive instances, which we did: During hyperparameter tuning, we kept the cost factor unchanged at 71 ($-w1 = 71$). For SVR (which does not allow for a cost factor), we oversampled the positive instances in the training sets.

Logistic regression (LRM)

We used the `lrm` function from the `Design` package in *R* for training and evaluating models based on logistic regression.[21] LRM uses Maximum Likelihood Estimation (MLE) as optimization function. It has a built-in option for data balancing (applying a weight vector to all instances), of which we found that it has exactly the same effect on the data as oversampling the positive instances in the training set. The other hyperparameters in LRM (a parameter for handling collinearity in stepwise approaches and a penalty parameter for data with many features and relatively few instances) are not relevant for our data. Therefore, we refrained from hyperparameter tuning for LRM: We only trained models using the default parameter settings for both the original and the balanced data.

5.4.3 The pairwise approach

For the pairwise approach, we evaluated Joachim's Ranking SVM algorithm [44]. In addition, we evaluated the same classification and regression techniques as in the pointwise approach. We made this possible by transforming our data into instance pairs that can be handled by these techniques (as explained below).

Ranking SVM

We used version 6 of SVM^{light} for our Ranking SVM experiments.[22] Ranking SVM considers the training data to be a set of instance pairs, each pair consisting of one correct and one incorrect answer. On these instances, the system performs pairwise preference learning [44]. The ranking order of a set of training instances is optimized according to Kendall Tau:

$$\tau = \frac{(N_c - N_d)}{(N_c + N_d)} \tag{5.4}$$

[20] This means that each next value is 4 times as high as the previous, so we go from 2^{-13} to 2^{-11} to 2^{-9} etc.

[21] See `http://cran.r-project.org/web/packages/Design/index.html`

[22] See `http://svmlight.joachims.org`

in which N_c is the number of concordant item pairs (the two items are ordered correctly) and N_d is the number of discordant item pairs (the two items are ordered incorrectly).

Similar to the other SVM techniques, we used the RBF kernel in Ranking SVM, which takes both the hyperparameters c and γ. For tuning these parameters, we searched over the same grid as for SVC.

Classification and regression techniques

To enable the use of classifiers and regression techniques in a pairwise approach, we transformed our data into a set of instance pairs. We presented the answers in pairs of one correct and one incorrect answer to the same question. We kept the number of features constant (at 37), but we transformed each feature value to the difference between the values of the two answers in the pair. In other words, we created feature vectors consisting of 37 difference values.

In the training data, each instance pair is included twice: 'correct minus incorrect' with label 'correctly ordered' and 'incorrect minus correct' with label 'incorrectly ordered'. In the testing phase, we let the classifier assign to each instance pair the probability that it is correctly ordered. Then we transform the data back to normal answer instances by summing the scores for each answer i over all pairs $[i, j]$ in which i is ranked first.

We evaluated the same classifiers and regression techniques in the pairwise approach as we evaluated for the pointwise approach: Naive Bayes, Support Vector Classification, Support Vector Regression and Logistic Regression.

5.4.4 The listwise approach

In the listwise approach there is no classification of instances or instance pairs; instead, the ordering of an answer cluster as a whole is optimized. As explained in Section 5.2.2, the implementation of listwise ranking approaches is a recent development and the results obtained with these techniques are promising [124, 121]. We evaluated two listwise optimization algorithms: SVM^{map}, and our own implementation of a genetic algorithm that optimizes the order of answers per question using MRR as fitness function.

SVMmap

SVM^{map} is a freely available algorithm[23] that takes clusters of instances with binary relevance labels as input and optimizes the instance order within each cluster for Mean Average Precision (MAP) [124]. In SVM^{map}, we again used the RBF kernel. For tuning the parameters c and γ, we searched over the same grid as for SVC.

Genetic algorithm (GA)

We used a Perl implementation of a genetic algorithm (GA) [33] for our experiments.[24] Our aim when training the genetic algorithm was to find the optimal weight vector for our feature vector of 37 feature values (a linear combination of feature values). As weights, we used the integers 0 to 10. In terms of the genetic algorithm, each possible weight vector is an individual.

In each run ('generation'), the GA selects the configurations that give the highest MRR on the training set (the 'fittest individuals') for crossover ('mating'). By default, the crossover rate is 0.95 and the mutation rate 0.05. For the selection of individuals, we chose tournament selection, which is the most efficient strategy. We used uniform crossover because the order of our features in the feature vector is not relevant. In our experiments, we set the generation size to 500 and the number of generations to 50 based on the shape of the learning curve in earlier experiments on the same data.

We did not run a meta-level GA for tuning our GA, because implementing such a procedure proved to be computationally prohibitive.

5.5 Results and discussion

The results that we obtained are in Table 5.2. For all settings, success@150 is 78.5% (This score does not change because there are no new answers retrieved by the ranking module). Success@10 is around 56% for the best-scoring settings (compared to 45% for the TF-IDF baseline).

5.5.1 Discussion of the results

For significance testing, we used the Wilcoxon Signed-Rank test on paired reciprocal ranks (RRs): Per question, we took the RR of the highest ranked correct answer in

[23] See `http://projects.yisongyue.com/svmmap`

[24] See `http://search.cpan.org/~aqumsieh/AI-Genetic-0.04`

Table 5.2: Results for the pointwise, pairwise and listwise approaches in terms of MRR@150. An asterisk (*) on an MRR score indicates a statistically significant improvement ($P < 0.01$ according to the Wilcoxon Signed-Rank test) over the TF-IDF baseline. A dagger (†) indicates that the MRR score is not significantly lower than the highest MRR score (0.35).

	Pointwise approach			
Technique	*Orig. default*	*Orig. tuned*	*Balanced default*	*Balanced tuned*
TF-IDF	0.25			
NB	0.19	-	0.20	-
C-SVC	0.10	0.32*†	0.32*†	0.33*†
ε-SVR	0.34*†	0.30*	0.33*†	0.32*†
LRM	0.34*†	-	0.31*	-
	Pairwise approach			
Technique	*Default*	*Tuned*		
NB	0.32*†	-		
C-SVC	0.32*†	0.34*†		
ε-SVR	0.32*†	**0.35***		
LRM	0.31*	-		
Ranking SVM	0.13	0.33*†		
	Listwise approach			
Technique	*Default*	*Tuned*		
GA-MRR	0.32*†	-		
SVM^{map}	0.33*†	0.34*†		

two system settings. Then we made 186 pairs of RRs for these two settings and calculated the Wilcoxon score over them.

The highest MRR score that we obtained is 0.35 (by SVR for pairwise classification).[25] We will call this the optimum in the remainder of this section.

Comparing pointwise, pairwise and listwise approaches

We obtained good results with techniques following either of the three approaches: pointwise, pairwise and listwise. The results for the pairwise approach much resemble the results for balanced data in the pointwise approach. This finding confirms the

[25]The 21% of questions without a correct answer in the top 150 all have an RR of 0; the MRR for the successful questions only is 0.45. This is quite high considering the Success@10 score of 56%. A further investigation of the results shows us that this is because a large proportion of successful questions has a correct answer at position 1 (Success@1 for all questions including the unsuccessful questions is 24.2%).

results found by other researchers on LETOR data: pairwise approaches are in some cases slightly better than pointwise approaches but pointwise approaches can reach good results if the data are balanced and the hyperparameters are tuned properly. We should also recall here that we applied clusterwise normalization to all feature values to overcome the problem that pointwise approaches ignore the clustering of answers per question (see Section 5.3.4).

For Naive Bayes, however, the results for the pointwise and pairwise approaches are very different. Here we see that presenting the problem as a pairwise classification problem is essential for Naive Bayes to predict the data correctly. We suspect that this is because the simplicity of the Naive Bayes model, which is based on the probability of each feature value given the class of the instance. When presenting the data in pairs, we apply a form of bagging: Each positive answer is included in the data many times, but each time as part of a different instance pair. As a result, all positive instance pairs are different from each other and the algorithm has more more quasi-independent data points available for learning to make the right decision for one answer. Not surprisingly, the Naive Bayes classifier depends on the availability of (quasi-)independent training data for learning a proper classifier.

In the bottom part of Table 5.2, we see that both our listwise approaches (GA-MRR and SVM^{map}) lead to scores that are not significantly lower than the optimum, but also not higher than the results for the pointwise and pairwise techniques. From the literature on listwise techniques one would expect a result that is better than the pointwise and pairwise approaches. We speculate that the failure of our GA approach to outperform pointwise and pairwise approaches is because the linear feature combination with integer weights that we implemented in the Genetic Algorithm is not sophisticated enough for learning the data properly. The results for SVM^{map} may be suboptimal because the optimization is done on a different function (MAP) than the evaluation measure (MRR). We (and others [56]) have found before that the best results with listwise techniques are obtained with a loss function that optimally resembles the evaluation measure. In that respect, it would be interesting to experiment with the lesser known algorithm SVM^{mrr} [20].

The effect of data imbalance

As pointed out in the machine learning literature (see Section 5.2.3), classifiers are in general sensitive to data imbalance. Table 5.2 shows that especially pointwise SVC gives very poor results in its default setting on our imbalanced data. If we balance the data, SVC reaches good results with the default settings of LIBSVM.

As opposed to SVC, the results for Naive Bayes are not improved by balancing

the data. Above, we speculated that this is due to the simplicity of the Naive Bayes model: We assume that oversampling the positive instances will only change the prior probabilities for the classes.

We find that for regression techniques (SVR and LRM), balancing the data by oversampling or applying a cost factor leads to slightly (not significantly) lower MRR scores. In Section 5.2.3, we concluded from the literature that class imbalance causes fewer problems for regression techniques than for classifiers because in the regression model, the intercept value moves the outcome of the regression function towards the bias in the data. Building a regression function on data in which the positive instances have been oversampled apparently leads to slight overfitting.

The effect of hyperparameter tuning

The effects of hyperparameter tuning on the ability of a technique to model our data varies much between the different techniques. The results for pointwise SVC show that with optimal hyperparameter settings SVC is able to reach a good result, even for the highly imbalanced data, on which the default settings performed very poorly. The other technique for which the default settings give a result below baseline (Ranking SVM) also profits much from hyperparameter optimization ($P < .0001$).

On the other hand, for those settings where default hyperparameters already give good results (most pointwise approaches on the balanced data and most pairwise approaches), we see that hyperparameter tuning does not lead to significantly better (or even worse) MRR scores. The only exception is pairwise SVR with $P = 0.01$ according to Wilcoxon.

There are two possible explanations for the finding that for some techniques, optimization of hyperparameters does not give a significant improvement. The first possibility is that our balanced data set very much resembles the type of data for which the developers of LIBSVM created the default hyperparameter settings in their software. More plausible however is that the learning problem we consider is relatively easy: A subset of the correct answers can easily be recognized in the data and is ranked high in all kinds of settings (providing that the data have been balanced), and another subset is that difficult that none of the experimental settings can recognize it as correct answers. As a result, the default settings that are already capable of ranking the relatively easy parts of the data correctly reach the same overall score as the tuned settings, which all have troubles with ranking the difficult parts of the data. Since hyperparameter tuning is expected to give a significant improvement for all techniques, we speculate that our data is in this respect essentially different from most data learned by support vector techniques in the literature. Note that it is

not the nature of the problem of *why*-QA that determines the characteristics of the learning problem, but the nature of the features that we used.

There is one setting where we observe a significant ($P = 0.0003$) degradation in MRR by hyperparameter tuning: pointwise SVR on the original data. After carefully analysing the results, we have to conclude that, despite our efforts to prevent overfitting (see Section 5.4.1), our tuning strategy still suffers from overfitting of the hyperparameters to the tune sets, with highly variable consequences for the test set. A full presentation of this problem, and possible solutions, goes beyond the scope of this thesis and should be the subject of future work.

5.6 Conclusion

In this chapter, we have optimized the re-ranking module of a system for *why*-question answering. The goal of this chapter was to compare a number of machine learning techniques in their performance on the task of learning a ranking for answers that are described by a set of 37 linguistically-motivated overlap features and a binary label representing their correctness. We evaluated learning techniques in pointwise, pairwise and listwise approaches.

We found that with all machine learning techniques, we can get to an MRR score that is significantly above the TF-IDF baseline of 0.25 and not significantly lower than the best score of 0.35.[26]

We are able to obtain good results with all three types of approaches for our data: pointwise, pairwise and listwise. The optimum score was reached by Support Vector Regression for the pairwise representation, but some of the pointwise settings reached scores that were not significantly lower than this optimum. We argue that pointwise approaches can reach good results for learning-to-rank problems if (a) data imbalance is solved before training by applying a cost factor or oversampling the positive instances, (b) feature value normalization is applied per answer cluster (query-level normalization) and/or (c) proper hyperparameter tuning is performed.

We obtained reasonable results with two listwise techniques: SVM^{map} and a Genetic Algorithm optimizing for MRR. Relative to the pairwise and pointwise approaches, our results are somewhat lower than the results reported on the LETOR benchmark data, where listwise approaches outperform pairwise and pointwise approaches. We attribute this to the choice of optimization function (MAP vs. MRR) and our relatively simple implementation with a linear combination of feature values

[26] Our conclusions are based on experimental results. We think that it might be interesting for model developers to use our findings for better understanding the effects of their models on specific data types.

respectively.

We found that for our imbalanced data set, some of the techniques with hyperparameters heavily depend on tuning. However, if we solve the class imbalance by balancing our data or presenting the problem as a pairwise classification task then the default hyperparameter values are well applicable to the data and tuning is less important. The pairwise transformation enables even Naive Bayes to classify and rank the data properly. Since hyperparameter tuning is a process that takes much time and computational power, a technique without hyperparameters, or a technique for which tuning can be done easily without heavy computing, should be preferred if it reaches equal performance to techniques with (more heavy) tuning. In this respect, regression techniques seem the best option for our learning problem: logistic regression reaches a score very close to the optimum (MRR is 0.34) without tuning. Pairwise support vector regression reaches optimal performance (MRR is 0.35) with tuning.

It seems that with the current feature set we have reached a ceiling as far as individual rankers are concerned. We might still achieve an improvement with a combination of rankers or second level classifiers, but the overlap between the results for individual systems is such that this improvement can never be very big.[27]

Moreover, given the experimental result of 0.35 and the theoretical optimum of 0.79 (if for all questions with at least one correct answer a correct answer is ranked at position 1), we can conclude that our features are suboptimal for distinguishing correct from incorrect answers. Since we already invested much time in finding the best features for describing our data (see Chapter 4), we conclude that the problem of distinguishing correct and incorrect answers to *why*-questions is more complex than an approach based on textual (overlap) features can solve.

In Chapter 6, we will review the problem of answering *why*-questions in detail. In automatically answering complex questions such as *why*-questions, human reasoning and world knowledge seem to play an important role. We will investigate the limitations of an Information Retrieval-based approach that relies on word overlap for complex question answering.

[27] In fact, we did some preliminary experiments with a linear combination of answer scores from combinations of rankers. By use of a hill-climbing mechanism we were able to find a set of weights that leads to an MRR on the test set of 0.38. This is significantly better than the best-scoring individual technique ($P = 0.03$). We have not attempted second level classification, since this would mean repeating all our experiments with a nested cross-validation in order to do a proper training and tuning of the second level classifier.

6

Rethinking the Problem of Why-Question Answering

Edited from: Suzan Verberne, Lou Boves, Nelleke Oostdijk, Peter-Arno Coppen. Rethinking the Problem of Why-Question Answering. Submitted to *Artificial Intelligence*.

Abstract

In this chapter, we review the problem of answering *why*-questions. After discussing the history of the use of knowledge in Question Answering research, we provide an in-depth analysis of the questions that our system could not answer. While the most important problems surface in the form of a vocabulary gap, a deeper analysis shows that what we are confronted with is the well-known knowledge acquisition bottleneck that has plagued knowledge-based AI-systems. We then proceed to analyzing the ways in which humans cope with the fact that the vocabulary in a question and its answer may show no overlap, and that explanations need not always be recognizable by virtue of dedicated syntactic or semantic constructions. We conjecture that human readers use previously learned concept associations in order to recognize that an answer passage answers the input question. These concept associations go beyond the classic synonym and hypernym relations which are generally contained in lexico-semantic resources such as WordNet. We then propose two approaches, both based on results of recent research in text mining, that can help to remove the knowledge acquisition bottleneck without the need for taking recourse to hand-crafted rules.

6.1 Introduction

The task of automatically answering questions in natural language has been studied extensively in the fields of computational linguistics and artificial intelligence. Question Answering (QA) research has its roots in the 1960s, with the development of natural language database interfaces and expert systems. In the nineties, knowledge-based QA-research gave way to an approach that heavily relies on statistical text retrieval techniques. At the same time there has been a development away from domain-specific to open-domain questions. This development has been encouraged by the TREC-QA track[1], which also introduced the use of common evaluation measures.

The 1999 QA track contained 200 questions, the large majority of which were factoids: questions asking after *who*, *where*, *how many*, etc. that expect a named entity as answer. Only two of the 200 questions were *why*-questions [110]. From 2002 onwards, *why*-questions were no longer included in the track's main task. This is because *why*-question answering (*why*-QA) is considered a discipline in its own right [62]: the answers to *why*-questions are not named entities (which are in general clearly identifiable), but paragraph-length passages giving a (possibly implicit) explanation [104].

Following the approaches developed in factoid QA we have studied *why*-QA by combining techniques from Information Retrieval (IR) and Natural Language Processing (NLP) (see Chapters 4 and 5). We have focused on open-domain *why*-questions for which it is reasonable to expect that an answer can be found in an encyclopedia. In addition, rather than inventing questions that seemed interesting from some theoretical perspective, we experimented with questions from the Webclopedia question set [39] that have been asked on the Internet by real users. For a randomly selected subset of 700 *why*-questions we manually searched for an answer in the Wikipedia 2006 XML corpus [24]; for only 186 questions we were successful. One of the things we will investigate in this chapter is how it is possible that almost three out of four *why*-questions that are asked on the web do not have an answer in the largest encyclopedia on the web.

We used the 186 questions for which we could find an answer in Wikipedia to understand what an automatic system must be able to accomplish in order to find a satisfactory answer. For each question, our system retrieves a set of 150 candidate answers and ranks them according to their estimated relevance to the question. Our system was able to find an answer for 79% of these questions; for 57% of the ques-

[1] See `http://trec.nist.gov/data/qamain.html`

tions the answer was in the top-10 of the ranked result list. While one might argue that finding an answer for 79% of the questions for which an answer is available is a promising result, one might also take a more pessimistic stand and emphasize that only 57% received a useful answer, since few users are willing to read beyond the tenth passage in a ranked list of answer candidates [45]. The results of our manual search for answers and the evaluation of our system both suggest that there is a need for better understanding the problem of answering *why*-questions.

In this chapter, we describe the history of automatic QA, from database interfaces through knowledge-intensive intelligent systems to the statistical approach in IR. We discuss the shortcomings of our system for *why*-QA, illustrating the limitations of the overlap-based IR approach for *why*-QA. We relate these limitations to the process of human text understanding in order to identify the knowledge gaps from which our system suffers. We then investigate how modern knowledge-based systems that were previously superseded by statistical approaches can fill the gaps by making optimal use of language and knowledge resources that have been vastly extended and improved over the last decennia.

Our research questions are the following:

1. To what extent can *why*-questions that are asked on the web be answered using encyclopedic data that is available online?

2. What are the limitations of an IR approach to *why*-QA?

3. To what extent can models of human text understanding explain the failures of our system for *why*-QA?

4. Can modern (resource-driven) implementations of knowledge-based approaches remedy these system failures given the current state of knowledge sources?

This chapter is structured as follows: First, in Section 6.2, we give an overview of the history of QA research. Then, in Section 6.3 we focus on the functionality and performance of our system for *why*-QA. In that section we also try to explain why we failed to find an answer to 75% of open-domain *why*-questions in Wikipedia. In Section 6.4 we focus on an analysis of the problems that our system experiences in handling the questions for which we knew that an answer was available. Then we relate these system failures to the human competence of finding and recognizing the answers that our system was not able to retrieve (Section 6.5). We conclude with a proposal for a future knowledge-based approach to *why*-QA in Section 6.6.

6.2 History of the use of knowledge in QA research

In this section, we give an overview of the role of knowledge in QA systems over the years. Where possible, we will focus specifically on the problem of *why*-questions.

6.2.1 Database systems: factual knowledge

In a survey paper published in 1965, Simmons [86] discusses fifteen experimental English QA systems that had been developed since the advent of mainframe computers. Most of these systems aimed at offering a natural language interface to some specific database. All operated within a restricted domain. The QA systems BASEBALL [34] and LUNAR [119], for example, used domain-specific knowledge bases that were handcrafted by experts. The LUNAR system was demonstrated at a lunar science convention in 1971 and it was able to answer 90% of the questions posed by people untrained on the system.

These early QA systems already made a distinction between different question types based on the question word (*who*, *when*, *where*, etc.). Since the main focus of these systems was on factual knowledge, they were mainly suited for answering factoid questions by retrieving answers from databases. Simmons mentions a few text-based systems, which attempted to find answers in unstructured text. These systems all used syntactic parsing of both the question and the answer text, and made use of synonym expansion. Although in the 1960s the systems could only deal with small amounts of text, they were designed with much larger amounts of texts in mind.

Despite the general optimism about the future of QA systems, Simmons concluded that there were four big challenges for QA system developers: (1) the interpretation of sentence meaning, (2) dealing with ambiguities, (3) making inferences between sentences, and (4) increasing the scope of the domains covered by the systems.

6.2.2 Intelligent systems: Knowledge combined with reasoning

In the 1970s, QA research was mainly aimed at the development of expert systems, i.e. systems that can perform complex tasks within a specific domain by modeling the behavior of human experts. For that purpose, comprehensive knowledge representation theories and decision-making and inference procedures were developed and implemented in operational expert systems.

One approach to building knowledge representations were the scripts proposed by Schank and his collaborators [83]. Handcrafted scripts were combined with in-

ference rules to implement systems that could understand stories. This was based on the idea that the human capability of understanding stories also originates from previously learned situation scripts.

Lehnert was the first to apply the script approach to automatically answering questions starting with *why* in her text comprehension system SAM [51]. She distinguishes three answer types for *why*-questions: motivation, goal orientation, and physical causation. She describes the answer to a *why*-question as a causal chain: two situations that are linked by causality. SAM saves all inferences and resolutions that it encounters in the story. When a *why*-question is asked to test its understanding of a story, the system examines all inference–resolution pairs to see if the question statement matches any of the resolutions. If it finds one, the corresponding inference will explain the resolution.

Manually creating scripts and providing a system with all the knowledge that is needed to create inference–resolution pairs for arbitrary stories seems at best feasible for a limited number of restricted and well-understood domains. In its classic, rule-based form, this approach cannot be generalized to open-domain text comprehension. In their 1993 book, Engelmore and Feigenbaum [25] stress the importance of knowledge in expert systems and state that "much of the future of expert systems depends on breaking the knowledge acquisition bottleneck and in codifying and representing a large knowledge infrastructure".

Orkin and Roy [72] go one step further: They claim that handcrafted scripts and inference rules suffer from the fundamental limitation that rule-based systems are not able to cope with the many unforseen situations that happen in real life. This explains why the QA-approach in expert systems eventually had limited success.

6.2.3 Adding linguistic knowledge to statistical IR-based systems

At the same time that developers of expert systems experienced the limitations of handcrafted knowledge bases, the field of Information Retrieval (IR) was making significant advances using approaches based on word overlap. For QA this raised the question whether it may be possible to bypass the knowledge engineering bottleneck by using information that can be collected by means of a somewhat superficial skimming of large amounts of text. The availability of text retrieval techniques resulted in the emergence of open-domain QA systems in the IR field in the mid 1990s, boosted by the NIST evaluation campaigns TREC-QA (for QA in English) and CLEF-QA[2] (for multi-lingual QA).

[2]See `www.clef-campaign.org`

In the first QA track, TREC-8 in 1999, the best QA system already reached an MRR[3] of 0.66 on a test collection mainly consisting of factoid questions [111]. All successful systems in TREC-8 combined standard bag-of-words retrieval techniques (ignoring the order of the words and the corresponding linguistic structure) with named entity recognition for retrieving entities of the type asked for in the question (e.g., a person for *who*, a time designation for *when*). The best systems were able to answer two out of three factoid questions correctly, which was considered very successful.

In the years following TREC-8, more difficult question types (e.g., questions that expect a list of items as answer) and more realistic questions were added to the test collection [110]. Some developers tried to include knowledge about question structure in their system, but from 2001 onward a steadily increasing preference for a shallow, data-driven approach can be observed [112]. This approach to answering factoid questions remained unchanged over the years [114, 113, 115, 116], keeping the overall system performance for the task stable, despite the fact that the difficulty of the task increased.

Nevertheless, attempts were made to improve the retrieval-based QA systems with linguistic knowledge. Most of these experiments focus on adding syntactic information to a baseline system that represent both the query and the response documents as bags of words. Systems using NLP techniques often show a small but significant improvement over the bag-of-words baseline [92, 80, 88].

Buchholz [13] describes an approach to QA that combines basic text retrieval with an NLP component exploiting a set of grammatical/discourse-type relations [13]. One of the relations that she considers is purpose/reason, which implies that her system would in principle be able to answer *why*-questions. An evaluation on the TREC-QA question set however shows that none of the (only two) *why*-questions in this set was answered. The performance that her system reaches on the total set of questions is satisfying but not impressive.

Narayanan and Harabagiu [69] investigate the value of deep semantic representations combined with predicate–argument relations. They show that a QA system can profit from this information, especially for question types pertaining to causal and temporal relations [69]. However, the costs incurred by computationally intensive syntactic and semantic processing hardly justify the gains in performance. Consequently, most QA systems developed in the last decade use bag-of-words passage retrieval techniques, combined with knowledge of the expected answer type, thereby

[3]MRR is based on the rank of the highest ranked correct answer for each question. Reciprocal rank (RR) is 1 divided by this rank (RR=0 if no correct answer is retrieved); MRR is the mean RR over all questions.

mainly relying on word overlap.

Although overlap-based techniques have proven to solve a significant part of the QA problem, the ceiling that has been reached in the performance of open-domain systems seems unsatisfactory. In their summary of the CLEF 2008 QA track, Forner et al. state: "In six years of QA experimentation, a lot of resources and know-how have been accumulated, nevertheless systems do not show a brilliant overall performance, even those that have participated to most QA campaigns, and still seem not to manage suitably the different challenges proposed" [29].

6.2.4 Why-QA

It is generally assumed that *why*-questions require a different approach than factoids because their answers are longer and more complex than named entities. This was the starting point of our research.

In Chapter 2, we presented a classification of answer types for *why*-questions, in analogy to the answer type classification proposed by Lehnert [51] and the classifications that substantially contribute to the performance of factoid QA. We distinguished the following answer types, based on the classification of adverbial clauses by Quirk et al. [81] (section 15.45): cause, motivation, circumstance, and purpose. In open-domain questions the latter two types are very rare and by far the largest proportion of *why*-questions is causal. Moreover, we encountered one other relatively frequent answer type in our data: etymology (e.g. "Why are Hush Puppies called Hush Puppies?"). We will come back to this in Section 6.3.

All types of answers to *why*-questions entail some kind of explanatory or causal relation. In Chapter 3 we focused on discovering explanatory passages with the use of discourse structures in the framework of Rhetorical Structure Theory (RST) [60, 61]. We found that 42% of the answers were completely implicit, i.e., they did not have the form of a relevant rhetorical relation. Moreover, experiments with adding RST annotations to Wall Street Journal texts and Wikipedia articles showed that it is difficult to reach agreement between annotators about rhetorical relations, even on the sentence level [18, 99]. Therefore, an approach based on more superficial features of candidate answer paragraphs appeared to be necessary.

Higashinaka and Isozaki developed an approach for answering Japanese *why*-questions [37]. They extract features for causal expressions and causal relations from two annotated corpora and a dictionary and apply these features to rank candidate answer paragraphs extracted from the top-20 documents obtained with a document retrieval system. They evaluate their ranking method using a set of 1,000 Japanese *why*-questions that were formulated to a newspaper corpus by a text analysis expert.

70% of the reference answers that were retrieved in the top-20 documents is ranked in the top-10 by their system. However, this result may be over-optimistic; in Chapter 2 we found that asking readers to formulate (*why*)-questions for which the answer is in a text creates a strong bias towards relations and inferences that are explicitly expressed in the text.

6.3 Data and approach for why-QA

In this section we present the data we collected and the set-up of our system for *why*-QA. We will explain the issues that we encountered in developing a system aimed at handling *why*-questions that have actually been asked to an online QA system.

6.3.1 Characteristics of the data we collected

For our research we used the Webclopedia question set [39], which contains questions asked to the QA system `answers.com`. For 700 randomly selected *why*-questions (from the complete set of 805), we manually searched for an answer in the Wikipedia XML corpus from INEX 2006 [24], which contains the complete English Wikipedia from the summer of 2006. We pre-processed the corpus by segmenting it in half-overlapping passages with an average length of 428 characters; these passages formed the candidate answers.

As explained in Section 6.2.3, we found three semantic types of *why*-questions in the Webclopedia data: causal questions, motivation questions and etymology questions. The following three examples illustrate the type of data we are working with.

1. "Why did the Globe Theatre burn down?" (causal question) — "The first Globe burned to the ground in 1613, apparently by flaming material expelled from a cannon used for special effects during a performance of Henry VIII that ignited the thatched roof of the gallery."

2. "Why didn't Socrates leave Athens after he was convicted?" (motivation question) — "Socrates considered it hypocrisy to escape the prison: he had knowingly agreed to live under the city's laws, and this meant the possibility of being judged guilty of crimes by a large jury."

3. "Why was cobalt named cobalt?" (etymology question) — "The word cobalt comes from the German kobalt or kobold, meaning evil spirit, the metal being so called by miners, because it was poisonous and troublesome (it polluted and degraded the other mined elements, like nickel). Other sources cite the

origin as stemming from silver miners' belief that cobalt had been placed by kobolds who had stolen the silver. Some also think the name may derive from Greek kobalos, which means 'mine', and which may have common roots with kobold, goblin, and cobalt."

We were able to find a satisfactory answer to 186 of the 700 Webclopedia *why*-questions.[4] Thus, about 75% of the *why*-questions asked to `answers.com` did not have an answer in the Wikipedia 2006 corpus. We analyzed a random sample of 50 of these questions to find out why they did not have an answer in the largest online encyclopedia. We performed a manual search for the answers, first in the most recent online version of Wikipedia (July 2009) and if we could not find the answer there, we used Google on the complete web. By 'manual search' we mean that we manually formulated queries based on the question terms that we expected to be the most informative. If the first query did not yield a satisfactory result, we adapted the query until we felt that all options had been tried in vain. If a potentially relevant document was retrieved, we searched through it for the answer. If necessary, we clicked through to other documents.

We identified the following types of missing answers:

- For twelve of the 50 questions (24%) we were able to find the answer in the most recent version of Wikipedia. On these subjects, the coverage of Wikipedia apparently has extended since 2006. E.g. "Why do we smile when we're happy?" and "Why can't an American give a clock to a Chinese colleague?".

- A further three questions could be answered using Wikipedia, but their answers are very complex and therefore too long to be explained satisfactorily in a paragraph-sized passage or even in a document the size of an article in an encyclopedia. E.g. "Why did America lose the Vietnam War?".

- For thirteen questions, we could not find the answer in Wikipedia, but we were able to find the answer somewhere else on the web using Google. E.g. "Why do computer screens/TVs flicker when seen on a TV program?", "Why do some people wear a sprig of Rosemary on Anzac Day?" and "Why is my reflection upside down in a spoon?".

- For 22 of the 50 questions (i.e., almost 50%) we were not able to find the answer at all. Three of these questions seem to be jokes rather than representing a serious information need (e.g. "Why do hotdogs come in packages

[4] By 'finding a satisfactory' answer we mean that we found a passage of text of which we felt that it was a correct and complete answer to the question.

of 10 and hotdog buns in packages of 8?"). Three other questions could not be answered because their propositions are false. False propositions are not always prohibitive: With 'common misconceptions' such as "Why did Marie-Antoinette say 'Let them eat cake?"' (which is a wrong translation and erroneously attributed to Marie-Antoinette), the corrective answer can often still be found. However, if the misconception is not common, it is unlikely that it is discussed in a serious web document, e.g. "Why do the sunrise times vary greatly by latitude, but sunset times vary only slightly?". Two questions were meta-questions in the sense that they ask about the (performance of the) QA system itself: "Why don't you give more room to ask questions?".
However, 14 out of the 22 questions for which we could not find an answer on the web were serious questions for which an explanation is likely to be present in encyclopedia-type data. E.g. "Why do utility wires hung from poles sometimes hum?" and "Why do some aspen trees turn red or orange in the fall and not yellow?". We still think that for some of these questions an answer can be found on the web, but we failed to create effective queries. It is interesting to note that some of the propositions in these unanswered questions are frequently mentioned on the web without anyone explaining the phenomenon. E.g. "Why do some people call soft drinks pop?" and "Why is weird spelled w-e-i-r-d and not w-i-e-r-d?".

What does this mean for the issue of *why*-questions in general and for QA system developers in particular? First of all, Wikipedia is a reliable answer source that provides well-structured documents about a wide range of subjects, but its coverage is (even for encyclopedia-type questions) limited compared to the web as a whole. Good additional answer sources are Wiki Answers[5] and Yahoo! Answers[6] [118]. Second, the web keeps growing and questions that are asked by users of search engines are likely to be answered by the content of user-generated web sites over time. Therefore, the coverage of future web-based QA systems is likely to increase, but this may go at the cost of the credibility of the answers retrieved from less authoritative sources.

Most of the time, human searchers recognize an answer to a question they have asked when they read it. However, this does not imply that it is always easy to find the answer when it can be deduced from some web document. It may happen that the explanation is implicitly given in a document that can only be retrieved with a query that uses very different terms than the original question. For example, the answer to the question "Why do USA fax machines not work in the UK?" (the proposition

[5] http://wiki.answers.com
[6] http://answers.yahoo.com

of which is probably only partly true) can be deduced from documents that explain the difference between the ISDN-D channel in Europe and the USA. In cases such as this one may wonder if a non-specialist user will understand that the answer is provided in the retrieved document.

Thus, the success of answering *why*-questions using data available on the web is not only determined by the coverage of web resources and the query formulation skills of the user, but also by the user's capability of understanding the line of reasoning in the answer text.

6.3.2 Our system for why-QA

QA systems developed in an IR context almost invariably consist of a number of modules connected in a pipeline. The always present document or passage retrieval module heavily relies on a bag-of-words approach. Our system also has a pipeline setup. The first module is *question2query*, which transforms the input question to a query by removing punctuation and stop words. The second module is the Lemur retrieval engine[7]: we indexed the passages with the stemming option turned off and set Lemur to retrieve 150 answer passages from the Wikipedia passage index for the input query and initially rank them with the TF-IDF ranking function as built in in Lemur.[8] The third module is a re-ranking module that takes as input the 150 candidate answers that were ranked by TF-IDF and re-orders them according to a set of 37 features describing linguistic and structural aspects of the *why*-question and its candidate answers. The output of the re-ranking module is then presented as a ranked list of answer passages to the input question.

6.3.3 Knowledge exploited in our QA system

Our system uses linguistic knowledge in the re-ranking of candidate answers. The way in which the linguistic features are used and combined to obtain the best possible ranking is explained in detail in Chapter 5. In this section we briefly motivate and explain the linguistic knowledge that we employed in our re-ranking module. For a more detailed description of the selection of the linguistic features we refer to Chapter 4.

The *why*-questions in our data collection are complete sentences that consist of at least a subject and a predicate. The subject is in most cases realized by a semantically

[7] See `http://www.lemurproject.org/`

[8] The TF-IDF ranking function is based on the TF-IDF term weight measure that determines the importance of a term for a text by taking into account the frequency of the term in that text (Term Frequency) and the inverse of its frequency in the complete corpus (Inverse Document Frequency).

rich noun phrase (*Socrates*, *the Globe Theatre*), but in some cases by a pronoun (*I*, *we*) or a semantically poor noun (*people*, *humans*). The predicate is realized by either a verbal predicate with possibly one or more objects (e.g. *did't leave Athens*, *burn down*), or by a nominal predicate (e.g. *Buffalo wings* in "Why are chicken wings called Buffalo wings?"). Some questions contain an additional subordinate clause (e.g. *after he was convicted*). We captured the information contained in the syntactic structure of the question by defining a number of features that measure the overlap between a specific part of the question and (a specific part of) the candidate answer.[9] For example, we included a feature for the overlap between the question's main verb and all verbs in the candidate answer. We used automatic syntactic parsing [21] and a Perl script for automatically extracting all feature values.

Moreover, we employed discourse characteristics of our answer corpus (Wikipedia) in our set of structural features. Wikipedia articles always have an informative title capturing the topic of the document. We hypothesized that the topic of the answer document often overlaps with the topic of the question. We defined the question topic as follows[10]: For etymology questions, the topic is the subject complement: "Why are chicken wings called Buffalo Wings?". For questions with a semantically poor subject, the question topic is its (verbal or nominal) predicate: "Why do people sneeze?". For all other questions, the question topic is the grammatical subject: "Why did the Globe Theatre burn down?". These rules of thumb made it possible to extract the topic from the parser output of each question automatically. We included a number of features describing the overlap between the (topic of the) question and the (topic of the) answer document.

A third important type of knowledge that we included in our re-ranking module is information on specific phrases that are used to introduce an explanation. For this purpose we collected a set of 47 cue words and phrases such as *because*, *as a result of*, *which explains why*, etc.

Finally, we added a number of WordNet [27] expansion features that are derived from the syntactic and structural features mentioned above: In addition to the overlap between the question verb and verbs from a candidate answer, we expanded the question's verb with its WordNet synonym set. We calculated the overlap between this set and the verbs in the answer, and included this overlap as an additional feature. This allowed us to investigate the value of WordNet expansions for specific parts of the question compared to other parts. As a more general semantic similarity feature, we included the WordNet Lesk Relatedness score [76] for the question–answer pair,

[9] For details on how we calculated the overlap, we refer to Chapter 5.

[10] In previous work, we referred to the question topic as the question focus. We will come back to this in Section 6.5.1.

which is based on word overlap between the WordNet glossaries of the question words on the one hand and WordNet glossaries of the answer words on the other hand (cf. Section 6.6.1).

In the literature, there is some discussion on the benefit of lemmatization for question answering [7]. While we did not lemmatize the Wikipedia corpus before indexing it, we did apply lemmatization to some parts of the questions and the candidate answers for the extraction of features in our re-ranking module (see also Section 6.4.2). Since lemmatization is especially problematic in the case of proper nouns, we decided only to lemmatize verbs in the re-ranking module of our system.

In Chapter 4, we found that the most important question constituents appeared to be the question focus, the main verb and the direct object. On the answer side, most important were the title of the document in which the candidate answer was embedded and knowledge on the presence of cue phrases. In addition, the WordNet Relatedness score for the question–answer pair made a valuable contribution to the ranking performance.

6.3.4 Evaluation

For evaluation and training purposes, we had to assess which of the candidate answer passages were satisfactory answers to each question. Since the set of answers was quite large ($186 * 150 = 27,900$ instances), we manually judged a sample of the data and used estimations of relevance for the answers that were not judged. This is common practice for relevance assessments in IR research, where the large amounts of training instances force researchers to use estimations instead of complete manual judgments for labeling the instances [2].

We performed a form of sampling in which an assessor judged answers for each question, starting with the answer that is ranked first and stopping at the first satisfactory answer found. We did this for several system settings, which gave different rankings and therefore different samples of assessments. We used binary judgments, in which 'satisfactory' means "this passage answers the question", even if there is also some irrelevant information contained in the passage. 'Unsatisfactory' means "This passage does not (completely) answer the question". This way, we excluded obviously incomplete answers from the set of correct answers. Judgments by a second assessor on a sample of the annotated data showed that this task is relatively difficult: the two assessors agreed in 97% of the cases, but taking into account the chance agreement due to the highly imbalanced data (most answers are unsatisfactory), we reached only a moderate Cohen's κ value of 0.48.

Analysis of the differences showed that the judgments are somewhat subjective:

One assessor can be satisfied with an answer while another one is not. To understand why the task is so difficult we can refer to the question "Why was cobalt named cobalt?" (cf. Section 6.3.1). An assessor who has seen the list of possible explanations might deem a passage that includes only one of the options as 'incomplete', while an assessor who is not aware of the full list of options might deem that passage as satisfactory. It is worth noting that the ambiguity in deciding whether an answer to an open-domain *why*-question is 'correct' is yet another difference with previous work in expert systems and questions triggered by specific texts, where the unique 'correct' answer is always obvious. This is why we prefer to use terms such as 'satisfactory' or 'acceptable', rather than '(in)correct' in our evaluation.

As estimations for the relevance of the instances that were not part of the sample that was manually judged, we used a set of TREC-style answer patterns: a regular expression for each question that defines which answers should be labeled as acceptable. These answer patterns allowed us to not simply consider all unlabeled instances as incorrect (which is generally done in the case of judgments for a sample of the data [2]) but to label some of the unlabeled instances as correct because they matched the answer pattern for the question. For example, for question 2 in Section 6.3.1 ("Why didn't Socrates leave Athens after he was convicted?"), we developed the following answer pattern after assessing a sample of the candidate answers in our set: */(Socrates.* opportunity.* escape.* Athens.* considered.* hypocrisy | leave.* run.* away.* community.* reputation)/*. The pattern is based on two variants of the correct answer that we found in the set of candidate answers.[11]

We used the data assessed by the first annotator together with the set of answer patterns for the evaluation of our system. This evaluation showed that 79% of the 186 *why*-questions for which we had manually found an answer in Wikipedia had at least one satisfactory answer in the list of 150 candidates. Ranking by Lemur/TF-IDF only (without re-ranking) led to an MRR of 0.25. By applying the re-ranking module to the data (after optimizing the ranking function in the third module of the system), we reached an MRR of 0.35. 57% of the questions get at least one satisfactory answer in the top-10 (Success@10 is 57%) (see Chapter 5).

Our knowledge-based re-ranking module improved the MRR score for our system from 0.25 to 0.35 in a setting where 150 candidate answers were retrieved. One might wonder whether the performance could have been improved if the re-ranking module would have had access to a larger number of candidate answers. To answer this question we investigated success@n for values of n up to 2000. The results are shown in Figure 6.1. From this figure it appears that for $n = 150$ the function

[11] Note that the vertical bar separates the two alternative formulations.

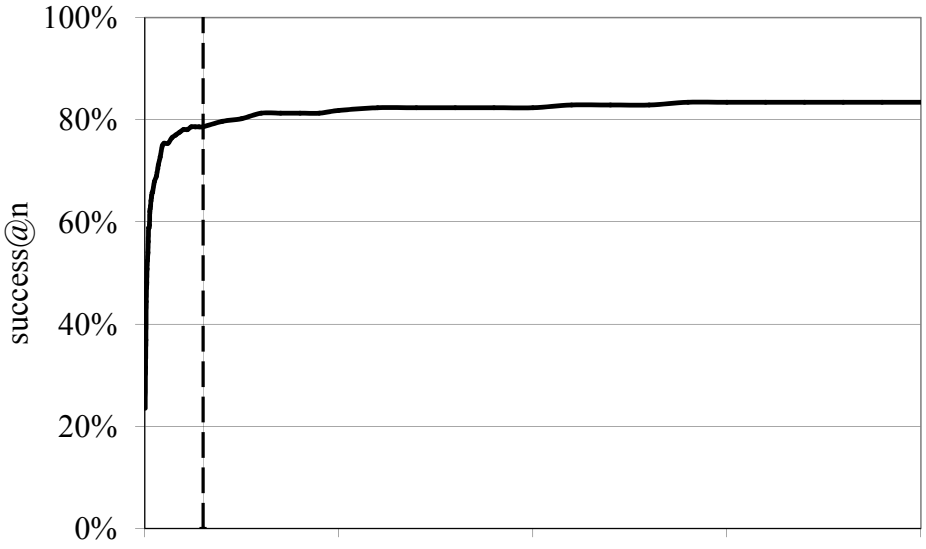

Figure 6.1: Success@n scores for $n = \{1, ..., 2000\}$, in which n is the number of candidate answers retrieved per question. The dotted line indicates the $n = 150$ point.

success@n is very close to the point where it approximates its asymptotical value. Therefore, it is safe to say that increasing n will only result in a large increase of the computational requirements for the language processing, without yielding a significant improvement in MRR and success@n.

6.4 Analysis of system failures

In this section we focus on the questions for which our system failed to find an answer, despite the fact that a satisfactory answer passage was present in the corpus. The main goal of the analysis is to establish whether the problems that appeared too hard for a largely overlap-based approach to QA could be solved by adding linguistic and world knowledge, not in the form of handcrafted rules, but using the resource-driven machine learning procedures that have shown their power in other information extraction tasks.

We did not encounter serious problems that were due to failures of the syntactic parser in finding the main verb, the subject and the direct object in the questions and the sentences in candidate answer passages. Therefore, we do not believe that efforts to improve the parser will be cost-effective. While improved parsing will certainly

not hurt, the return on investment will be minimal. This confirms the findings for the contribution of structural linguistic knowledge to improving statistical IR (cf. section 6.2.3).

In section 6.3.1 we suggested that the major cause for our own failure to find answers to many of the serious questions was our inability to formulate queries containing terms that are present in the answer passage. We found that the 40 questions (21% of the set of 186) for which our system could not find the proper passage in the Wikipedia corpus suffered from the same kind of problem: the terms in the question did not match the terms in the document that contains the answer. Therefore, there seems to be a serious vocabulary gap between the question and its answer. In an approach that completely relies on overlap such a gap is fatal.

6.4.1 The vocabulary gap problem in why-QA

The vocabulary gap problem is infamous in IR applications because it affects all systems that rely on lexical similarity (word overlap) [6]. For QA, the vocabulary gap problem is perhaps even more acute than for document retrieval because the answer to a natural language question does not necessarily repeat the words in the question.[12]

We identified three different types of vocabulary gap in our data: (1) the answer passage lacks one or more important terms from the question that happen to be present in irrelevant passages (term mismatch, 71%), (2) the question is short and underspecified; it contains frequent terms that occur in a very large proportion of the passages in the corpus (16%) and (3) the question is expressed by a lay person on a very different level than the answer formulated by a domain expert (13%).

Term mismatch

A question term missing in the answer passage is especially problematic if the question is relatively short. In most of these cases, the missing term is (part of) the core proposition of the question: generally the predicate or subject. For example, the answer to the question "Why is fire hot?" does not contain the word *hot* (but *heat* instead), and the answer to the question "Why don't women go bald just like men?" does not contain the words *women* and *men* (but *female* and *male*). In other cases, the missing question term modifies or disambiguates a question term that is highly frequent in the answer corpus. For example, in the question "Why is English the lan-

[12]The natural answer to "What is the capital of the Netherlands?" is "Amsterdam", rather than "The capital of the Netherlands is Amsterdam."

guage of the USA?", the term *USA* specifies the term *language*. In the bag-of-words approach this syntactic/semantic relation is lost and the retrieval system returns passages with the words *English*, *language* and *USA* in any relation. The correct answer passage does not contain *USA* (but *North America*) and consequently it is ranked lower than passages that do contain the query terms.

When investigating the question–answer pairs that suffer from a question term missing in the correct answer passage, we often found that the answer contains a term that is related to the missing question term. However, most of these relations are not pure synonymy. In our set of question–answer pairs with a missing question term, we find cases of spelling variants (*mosquitos–mosquitoes*, *double jointed–double-jointed*), near-synonymy (*USA–North America*) and hyperonymy (*WWI–war*), but most of the term pairs have a different, associative, relation (*north pole–high latitudes*, *hot–heat*, *peaked roof–roof shapes*).

Underspecified questions

Some questions, such as "Why are leaves green?", only contain words with a high frequency in the Wikipedia corpus. In the example question, the least frequent term *leaves* occurs in 19,925 passages. All 150 passages that are retrieved by Lemur contain both the terms *leaves* and *green* (or twice the word *leaves*), but none of these explains why leaves are green. Part of the problem is due to the fact that the words are not disambiguated for their parts of speech (*leaves* as noun versus *leaves* as verb) or capitalization (*green* as color versus *Green* as name). However, we conjecture that the fundamental problem is that the bag-of-words approach fails to capture the essential syntactic/semantic relation between *leaves* and *green*.

There are several possibilities for making a query that is generated from the input question more specific. One is to take into consideration multi-word phrases ("ice cream" in the question "Why are ice cream sundaes called sundaes?") or complete propositions ("leaves are green"). However, it is difficult to decide which multi-word sequences from the question (all bigrams? all noun phrases? all head-modifier pairs?) should be added to the query. Adding all possible n-grams will lead to very long queries for which it would again be difficult to predict the high-scoring passages. The use of head-modifier pairs (dependency relations) can be valuable [49] but requires proper syntactic annotation of the answer corpus, which is time consuming and not (yet) feasible for huge web corpora. A second option would be to require that candidate answers contain explicit explanations in the form of cue phrases such as *because*. However, in Chapter 4 we found that adding cue phrases to the Lemur query led to very long queries and introduced much noise. As a result, the overall

retrieval performance was not improved.[13]

Underspecified questions should not be viewed as anomalies. In human–human conversation it happens all the time that a question does not elicit a factual answer, but a request for clarification. In asking the first question the speaker might have relied on non-existing shared ground with the hearer. It might also be that the speaker was simply not able to formulate the question in a manner that allows for an adequate answer.

Question expressed on a different level than the answer

In our set of unanswered questions we found five cases in which the real question topic is not explicitly expressed: These questions contain a description of a concept for which the questioner searches an explanation without knowing the correct term. For example, the answer to "Why do our ears ring?" is an explanation of the phenomenon *tinnitus* and the answer to "Why does a woman have to be a virgin to be a nun?" is an explanation of the concept *chastity*.

If these questions are rephrased to the form "why [phenomenon]" (e.g. "why tinnitus", "why chastity"), chances are bigger that the answer is retrieved. We confirmed this by manually rephrasing the five unsuccessful questions as "why [phenomenon]" and running our pipeline system to them. In this form, four of these questions had an answer in the top-10. The fifth question ("Why does your temperature go up so many degrees when you have a cold?", reformulated as "Why fever?") suffered from the high frequency of the term *fever* in Wikipedia, as a result of which the answer was still not retrieved in the top-150.

Although rephrasing appears to be a solution for these problematic questions, it is not trivial to (automatically) decide which questions need rephrasing and which do not. We expect that some form of interaction with the user is needed here and this is beyond the scope of the current thesis.

6.4.2 Possible solutions to the vocabulary gap problem

A fraction of the problems related to the vocabulary gap might seem technical, rather than fundamental. This is especially true for mismatches due to spelling variants and morphological variations. However, things are not so easy as they might seem on the face of it. For example, we did not perform stemming on the queries and the answer corpus for the retrieval step, because earlier experiments showed that for

[13] Recall that cue phrases did contribute positively to the system performance when included as a feature in our re-ranking module.

the Webclopedia questions and Wikipedia answer passage corpus, Lemur's built-in stemming function did not improve retrieval performance. Spelling normalization will definitely help, but these problems are rare.

It is interesting to note that the vocabulary gap is not just an artefact of an automatic QA system that relies on word overlap. Humans also appear to suffer from the problem, be it to a lesser extent. In human–human communication, clarification questions (and misunderstandings) are also frequent. Furthermore, when a person who really needs an answer to some question interacts with a conventional search engine, and the first attempts at finding an answer are not satisfactory, (s)he will formulate multiple queries and each new query will be informed by the results of the previous ones. However, QA as it is usually conceived of in NLP and IR is a non-interactive process in which a question must be answered by a single context-free query. If a real user with a real information need would be constrained to typing a single question, it may happen that (s)he will fail to recognize an answer when it is returned.

Using lexical resources to close the vocabulary gap

The problem that *North America* is not treated as equivalent to *USA* (see Section 6.4.1) might be solved by exploiting semantic networks such as WordNet [27]. In Chapter 4 we found that semantic expansions only improve QA applications if they are applied to restricted (syntactic) contexts. For *why*-questions, synonyms of the question's main verb and direct object significantly contribute to the performance of our re-ranking module. However, expanding all words in the question with their WordNet synonyms introduced so much noise that performance did not improve.

WordNet only contains semantic relations between words of the same word class: Nouns are related to nouns and verbs to verbs. We found that for our data the lack of verb–noun relations such as *hibernate–hibernation* in WordNet caused failures. Also, the coverage of WordNet for proper names is limited: Only 9% of the nouns in WordNet are proper nouns [59]. One third of the questions in our Webclopedia set contains at least one proper noun (e.g. *B.B. King*, *Rice Krispies*, *Miletus*), most of which are not covered by WordNet. The relations in WordNet are limited to synonymy and *is-a* relations. However, concepts can be related in many ways: they may share a hypernym (e.g. *American–European*) or they tend to co-occur in texts (e.g. *steering–car*). Moreover, since we are working with *why*-questions and their answers, we are especially interested in causal or explanatory relations between concepts such as *fire–burn* and *flu–fever*.

To understand whether the limitations of WordNet can be remedied, we exper-

imented with additional semantic resources, viz., WikiOntology hypernyms [77], Nomlex [58] and Cause–effect relations (EDR Concept Dictionary [123]). We found that none of these additional resources improved the performance of our system significantly. However, this does not necessarily imply that this type of resources is fundamentally unable to improve performance. The resources that we used are not yet fully mature and open to substantial improvements, which eventually might have an impact on QA-performance. The WikiOntology Hypernyms and the Cause–effect pairs from the EDR Concept Dictionary were both very noisy. Since the WikiOntology was still under development, it is still possible that future versions of this resource may have a larger impact. The cause–effect pairs from EDR need a large amount of human clean-up before they could properly be applied in information extraction applications.[14] The verb–noun relations from Nomlex were clean and edited, but had a negligible impact because of the limited size of the resource.

6.5 Relating the system failures to human text understanding

In this section, we relate the vocabulary gap problem to human understanding of written language. We consider the situation in which a person is asked to judge whether a passage is a satisfactory answer to a question. We ask the question how this person solves the problems that were too difficult for our system.

In the process of human text understanding, concepts are activated based on the context in which words appear in a text. Semantically related words in the preceding context of the target word have a facilitating effect on the interpretation of a target word [70], but the semantic interpretation of the remainder of the sentence may also be needed for understanding a target word [67]. Connectionist models of word understanding represent the mental lexicon as a network of concepts with connections between associated concepts, and links between the lexical and conceptual processing levels [84]. In Latent Semantic Analysis (LSA) the meaning of a word is represented by the frequency with which it occurs in the context of other words [23]. For all we know, word meanings form a closely connected associative network in the human cognitive system, rather than the neat hierarchical organization that might be suggested by the conventional semantic relations of synonymy, hyperonymy, etc. Also, relations between words easily cross the borders raised by POS tags in lexico-

[14] As a result of automatic extraction from a corpus, the set of cause–effect relations contains a large amount of word pairs that are not useful. Especially frequent verbs such as *get* or *go* are labeled as an 'effect' of many (up to 500) 'causes'.

semantic resources such as WordNet.

In the following subsections we discuss aspects of human text understanding in relation to the system failures presented in Section 6.4. We distinguish two moments in the QA process where human text understanding can be related to challenges that were faced by our system: question interpretation (Section 6.5.1) and answer understanding (Section 6.5.2).

6.5.1 Question interpretation

Concept associations

In Section 6.4 we explained that the problem of the vocabulary gap in QA systems is due to the fact that these systems depend on word overlap. For humans, the vocabulary gap problem is less of an issue. We conjecture that this is because people use a combination of lexical associations and inferences to relate words from the query to words in a candidate answer passage. Thus, we could say that our system not so much suffers from a vocabulary gap but more from a 'knowledge gap': It cannot recognize associative concept relations between question words and answer words, while human readers can and actually do so while processing words in context.

We illustrate the difference between human text understanding and automatic processing of word overlap with the question "Why is English the language of the USA?" The answer passage that our system failed to find reads:

> "English was inherited from British colonization. The first wave of English-speaking immigrants was settled in North America in the 17th century."

The system failure is due to the fact that the answer does not contain the words *USA* and *language*, while there are many other passages in Wikipedia that combine the terms *English*, *USA* and *language*. For a human reader, who has encountered the words *USA* and *North America* many times in highly similar contexts, replacing *USA* by *North America* is effortless.[15] The same holds for *language* and *speaking*. While reading the question, concepts that are related to the words in the question are activated in the mental lexicon. Thus, by reading the words *English* and *USA*, the concept *North America* was already activated (together with many other words),

[15]This word pair is in fact an additional example of the shortcomings of WordNet as semantic resource: The two words do not co-occur in a synonym set. *North America* is contained in one synonym set together with the word *continent* and in two odd synonym sets with words such as *collection*, *generally accepted accounting practices* and *French West Indies*. USA occurs in two synonym sets: *United States* and *United States Army*.

thereby facilitating the understanding of the answer passage and recognizing its relevance to the question.

Question focus and the semantics of why-questions

The literature suggests that one of the most important features for answering natural language questions is the notion of question focus [35, 28, 96, 51]. In the literature the focus of *why*-questions has traditionally been defined as the part of the question that determines the question's 'contrast class': the set of alternative answers. For example, the question "Why did Adam eat the apple?" can — at least in theory — mean "Why did *Adam*, rather than someone else, eat the apple?" or "Why did Adam *eat* the apple, and not leave it on the tree?". Van Fraassen [96] observes that the contrast class is often not expressed explicitly if one looks at the question in isolation. However, it may well be that the contrast class is implied in the context in which the question was asked. This was already pointed out by Lehnert [51] who states that "understanding the point of a question often requires consideration of the question context as well as the use of general world knowledge". We already discussed the somewhat unnatural status of context-free questions in section 6.4.1.

We found that although in theory many *why*-questions have more than one possible focus, in practice it is not useful to identify this focus and its contrast class. There are three reasons for this. First, in many questions the focus is the complete predicate of the question. For example, a person asking the question "Why did Britain enter WWI?" is probably looking for a general explanation, not for an explanation why Britain, rather than some other country, engaged in WWI, or why Britain entered WWI, rather than another war. Second, the contrast class is often infinite and very diffuse. It is not only left tacit in most *why*-questions but also in (the context of) the answer. Third, if more that one focus is possible, most of the time only one of the interpretations is discussed in the text collection. Since it is not possible to find the answer to one of the other interpretations, knowing the question focus will not improve performance.

In conclusion, the assumed importance of question focus for *why*-questions in the literature does not apply to the practical task of retrieving answer passages in a non-interactive QA-system. Instead, it is more sensible to use a definition of question focus that has additional value for the data at hand. In previous work, we used the concept of question focus to refer to the topic of the question, in analogy to some approaches to factoid QA [28]. It appeared that the topic of a question often figured as the title of (or a heading in) the Wikipedia document in which the answer is contained. E.g. the answer to the question "Why did the Globe Theatre burn down?"

can be found in the document with the title (and the topic) *Globe Theatre* (see Section 6.3.3). Using this knowledge in the re-ranking module improved performance significantly.

6.5.2 Answer understanding

Cues for explanation

In our discussion of the problem that questions are sometimes underspecified (cf. Section 6.4.1) we already noted that expanding a query with cue phrases that signal an explanation did not improve the answer retrieval performance because much noise was added: Cue phrases often occur in a context where they do not function as a cue for explanation.

It is safe to assume that humans use complex and remote inferences as cues to recognize that some text offers an explanation. For example, in the answer "English was inherited from British colonization ... the 17th century", a human reader may interpret *was inherited* as a cue for a cause–effect relation. But there are also examples where humans can infer that the passage offers an explanation in the absence of specific lexical cues. This makes it extremely difficult to design processes that rely (almost) exclusively on lexical, syntactic or semantic overlap that can recognize that a text passage contains an explanation. This is in line with the finding that it is very difficult to devise automatic methods for annotating texts with discourse relations [61, 91], or to automatically mine explanation-type relations from texts [75], and with the fact that human annotators need substantial training to reach a reasonable level of agreement on the discourse annotation task [18, 99].

States of not knowing

Bromberger [11, 12] proposed a formal representation of *why*-questions and the conditions that define correct answers. He introduced two states of not knowing the answer to a question: P-predicament and B-predicament. "[In case of a B-predicament,] the answer is beyond what the person mentioned can conceive, can think of, can imagine, that is, is something that person cannot remember, cannot excogitate, cannot compose." [12]. This state of not knowing is reminiscent of the class of questions that are expressed on a different level than the answer (See Section 6.4.1).

Ideally, a QA system for complex questions should be able to handle questions of the type "Why do our ears ring?", because one cannot expect that the modal user will know all technical terms for phenomena that may become of interest. The example of

tinnitus can illustrate the similarities and differences between our system's approach and the human approach for finding the answer. The answer to the question "Why do our ears ring?" is:

> "The mechanisms of subjective tinnitus are often obscure. While it's not surprising that direct trauma to the inner ear can cause tinnitus, other apparent causes (e.g., TMJ and dental disorders) are difficult to explain. Recent research has proposed that there are two distinct categories of subjective tinnitus, otic tinnitus caused by disorders of the inner ear or the acoustic nerve, and somatic tinnitus caused by disorders outside the ear and nerve, but still within the head or neck. It is further hypothesized that somatic tinnitus may be due to 'central crosstalk' within the brain, as certain head and neck nerves enter the brain near regions known to be involved in hearing."

Our system did not retrieve this answer passage because the passage does not contain the words *ears*[16] and *ring*. A human reader who is not familiar with the term *tinnitus* might also have trouble recognizing this passage as an answer. He will, however, be able to understand that the passage is about some hearing disorder and therefore in some way related to the question.

Both an unknowledgeable human reader and our system would have been helped by reading the introduction of the Wikipedia document containing the answer passage: "Tinnitus ([...] from the Latin word *tinnitus* meaning "ringing") is the perception of sound within the human ear in the absence of corresponding external sound." Thus, a human searcher would be helped in recognizing the passage as the answer by reading the context of the passage.

The process of first skimming a complete document, and then use novel knowledge gleaned from the document for zooming in on the answer passage, can be imitated in a two-step retrieval process in which document retrieval precedes answer passage retrieval. We investigated the feasibility of such a two-step implementation. We found that although it gives good results for a subset of the questions (the questions that literally share their topic with the correct answer document), for other questions the document retrieval step has a negative filtering effect: The answer is not found because the relevant document was not ranked high enough in the first retrieval step. In the set-up of our system, it is simpler and more effective to add context information in the re-ranking module. In fact, we already included the title

[16]The passage does contain the singular *ear*, but remember that we did not perform stemming. Even if we had, chances that the passage would be recognized as an answer ro the question are slight.

of the document and the section heading in which each candidate answer is contained as features in our re-ranking module.

6.5.3 Concluding remarks on system failures

It is interesting to note that although virtually all problems that we encountered in answering *why*-questions are related to text understanding, we have encountered few, if any, cases in which model theoretic semantics may come to rescue (cf. the discussion of question focus in section 6.5.1). Rather, we have seen an urgent need to come to grips with representations of world knowledge that can be invoked to recognize that passages containing non-overlapping vocabulary still are related, perhaps even in the form that one passage provides an explanation of some state-of-affairs mentioned in the other. In conclusion, it is fair to say that we have come full circle: we can repeat the Engelmore and Feigenbaum's quote [25], replacing 'expert systems' by 'QA-systems': "much of the future of *QA-systems* depends on breaking the knowledge acquisition bottleneck and in codifying and representing a large knowledge infrastructure". In the remainder of this chapter we will explore two directions in which this challenge can be taken without the need to rely on hand-crafted rules.

6.6 Suggestions for bridging the knowledge gap

In Sections 6.4 and 6.5 we suggested that human readers understand written text by applying widely and tightly connected concept associations and inferences that they learned from previous experience. Our QA system, on the other hand, relies on word overlap. Existing lexico-semantic resources do not help very much in improving the performance of *why*-QA because they lack context sensitivity (see Section 6.4.2). For example, the word *language* occurs in 12 synonym sets with a total of 76 synonyms. Synonyms for different meanings of a term in a question may promote irrelevant answer passages, adding even stronger competition for the correct answer passage. Moreover, despite the large number of synonyms that most words have in WordNet, the sets often do not cover the data at hand. For example, the word *hat* in "Why do chefs wear funny hats?" occurs in 9 noun synonym sets comprising a total of 58 synonyms. But the answer term *toque* is not one of these synonyms.

In this section, we discuss two possible solutions for solving the knowledge gap in QA. The first is extending lexico-semantic resources with contextual information (Section 6.6.1). The second is an intelligent system approach based on models extracted from Wikipedia's link structure (Section 6.6.2).

6.6.1 Making lexical resources context sensitive

If existing lexical resources could be made more context-sensitive, they would benefit from local associations between words and concepts. Research in this direction has been done by Pantel and Lin [53, 74]. They acknowledge the importance of using word senses instead of word forms in IR and QA applications and the disadvantages of (manually constructed) lexical resources. They extract word similarities from text corpora using the principle that similar contexts define similar meanings (the distributional hypothesis), and employ the extracted similarities for word sense clustering. Pantel and Lin's intrinsic evaluations are promising [74] but it would especially be interesting for future work to perform an extrinsic evaluation of their resources in the context of a QA system.

Banerjee and Pedersen [5] have experimented with semantic relatedness based on the overlap between the glossaries of two words in WordNet. A glossary defines a word and illustrates its usage. As a bonus, this overlap yields semantic similarity measures for words with different word classes, which we have found to be one of the most important limitations of WordNet synonym sets (see Section 6.4.2). Glossary overlap was implemented as the Lesk relatedness measure in the WordNet Similarity package [76]. We experimented with Lesk relatedness as one of our re-ranking features and found a small but significant contribution (see Section 6.3.3). In-depth analysis of the output confirmed our assumption that Lesk relatedness adds less noise than direct use of synonyms and hypernyms in WordNet.

The mining of associative relations between words as done in [74] and [5] can lead to lexical resources containing contextual information for terms. Since a term can refer to several different concepts[17], and one concept can be expressed by different terms, we propose that contextual information should be modeled on two levels: a lexical layer containing word forms, and a conceptual level containing the concepts that ambiguous words can refer to. Contextual information on the co-occurrence of concepts can be modeled as associative relations between concepts. The frequency of co-occurrence is represented by the strength of the concept relations, and the different concepts one word can refer to are modeled by links between the lexical and conceptual level. Both layers should contain a combination of domain-independent statistics and domain-dependent data.

Resources as suggested in the previous paragraph could be constructed by means of Latent Semantic Indexing [23], but other techniques might be equally powerful. The mix of domain-dependent and -independent representations can be obtained by mining general and domain-specific corpora.

[17] Note that without context, almost all English content words are ambiguous to some extent.

6.6.2 Wikipedia's link structure as contextual information

The link structure of Wikipedia might prove to be another device for harvesting contextual information at the concept level. Recently, the value of Wikipedia's link structure for mining semantic relations has been studied by several researchers in the text mining field [64, 46, 68, 125].

Each Wikipedia document can be considered as representing a (potentially complex) concept. Associative relations between concepts can be learned by extracting for each Wikipedia document the links to other Wikipedia documents. All concepts addressed in a question would return their most related Wikipedia document. Links from these pages to other pages would then create a much larger set of potentially relevant documents, after which the cross-links between the pages could be used to prune that set so as to yield a subset comprising the most promising candidates.

The different concepts that one word can refer to can be extracted by mining all disambiguation pages from Wikipedia. In June 2009, the English Wikipedia contained over 110,000 disambiguation pages. A disambiguation page could be considered to represent an ambiguous word, with the list of linked entities on this page as the different concepts the word can refer to. In order to estimate the prior probability of occurrence of these different concepts, the number of incoming links to the pages of each of these concepts can be counted. For example, the word *USA* can refer to 26 different concepts according to its Wikipedia disambiguation page. Of these, the page United States has by far the most incoming links (over 50,000).

An implementation of the associative activation process that we propose here is beyond the scope of this thesis. We should also note that the actual implementation is far from trivial, as is shown by recent research on link structure mining [68]. The index process relies on robust normalization (capitalization and lemmatization issues) and properly trained frequency weights.

6.7 Conclusions and suggestions for future research

Following our analysis of system failures (Section 6.4), the relation between these system failures and human text understanding processes (Section 6.5), and our suggestions for bridging the knowledge gap (Section 6.6), we can now answer the questions posed in Section 6.1.

To what extent can *why*-questions that are asked on the web be answered using encyclopedic data that is available online? We found that for only 25% of the *why*-questions that were asked to an online QA service in 2001, Wikipedia contained the answer in 2006. In July 2009, the coverage of Wikipedia had grown, but still we

could not find the answer in this encyclopedia to the majority of *why*-questions. We expect the coverage of user-generated answer sources (both encyclopedic and QA repositories such as Yahoo! Answers) to grow over time, increasing the potential success of future web-based QA systems. Moreover, the success of answering *why*-questions online is not only determined by the coverage and reliability of web-based resources, but also by the query formulation skills of the user, and the user's capability of understanding the answer text and recognizing it as the answer.

It should be noted that the common procedure for evaluating the performance of QA-systems is somewhat unnatural, in that it only considers the answers returned to a single context-free query. A person who has a real need for an answer can be expected to engage in an interaction (be it with a system or a human) to rephrase the original question if no satisfactory answer is returned. There is an urgent need for developing an alternative research paradigm that provides room for interactive clarifications.

What are the limitations of an IR approach to *why*-QA? Retrieval approaches that are based on lexical similarity suffer from a vocabulary gap. For 21% of the *why*-questions in our data set, our system based on text retrieval was not able to find the answer despite the fact that it was available in the corpus. For all these questions this was due to the use of different words in the question and the answer. We discussed a number of specific cases, such as the problems of underspecified questions and users searching for explanations for phenomena for which they do not know the term. We found that most of the semantic relations between question words and answer words in our data are not synonymic or hypernymic but more vaguely associative, as a result of which conventional lexico-semantic resources do not provide us with sufficient information to solve the vocabulary gap.

To what extent can models of human text understanding explain the failures of our system for *why*-QA? We were able to explain to a large extent why humans are capable of recognizing answer passages as the answers to the problematic questions for which our system was not able to retrieve these answer passages. The most important characteristic of human text understanding in this respect is the implicit use of associative concept relations. This not only helps the activation of concepts in the mental lexicon based on the textual context, but also the mapping of the concepts represented by question words to concepts represented by answer words. Moreover, human readers are capable of recognizing and understanding explanations based on very weak cues on the lexical, syntactic, discourse and world knowledge level. We therefore conclude that the infamous vocabulary gap in QA is actually a knowledge gap: Retrieval systems are not able to make concept associations based on input words that are implicitly made by human readers.

Can modern (resource-driven) implementations of knowledge-based approaches remedy these system failures given the current state of knowledge sources? We proposed two alternatives for the current, limited use of lexical resources such as WordNet as sources for lexico-semantic relations. The first of these is to extend lexico-semantic resources with context-sensitive information that approximates the associative processes in human text understanding. This could be implemented in the framework of Latent Semantic Analysis. The second is an intelligent system approach based on models extracted from Wikipedia's link structure. Promising research is currently being conducted in the text mining field on mining Wikipedia's link structure for the exploitation of its semantic knowledge. Both approaches suggested above hold the promise of capturing substantial amounts of world knowledge in a manner that fits naturally in the statistical approach to Information Retrieval and Question Answering. Therefore, they hold the promise of bringing closer a solution of the knowledge acquisition bottleneck that has haunted automatic QA since its inception.

We can make a number of important recommendations for system developers that aim to bridge the knowledge gap in future QA systems: (1) Do not rely exclusively on specific semantic relations such as synonymy and hyperonymy but implement a source of more general associative relations. (2) Investigate methods (such Latent Semantic Analysis) that save concepts instead of words in the lexicon. If we make it possible in the future to represent the question and the answer corpus as bags of concepts instead of bags of words, then many implicit word relations will be more explicit. (3) Approach the concept matching problem as a task of concept activation. Words from the question and the answer need not necessarily overlap, as long as they activate (partly) the same set of concepts. It remains to be seen what future developments in syntactic and semantic theory, implemented in improved syntactic parsing and formal automatic semantic interpretations, can contribute to removing the knowledge acquisition bottleneck.

In conclusion, we can say that the main challenge of answering *why*-questions is to bridge the knowledge gap that exists between questions and answers. The manifestation of this gap is different for humans and QA-systems. Humans may not be able to find the proper terms or expressing a query and they may have difficulty in understanding that a piece of text does contain an answer to the question, again because they do not recognize the terms. For QA systems, the knowledge gap is also present as a vocabulary gap between question and answer. In addition, automatic systems may have difficulty recognizing implicit reasoning and explanation in the answer passage. Since this capability is problematic for machines but very natural

for human readers, the process of *why*-QA deserves renewed attention from the field of artificial intelligence.

7

Summary and Conclusions

In this thesis, we studied the problem of automatically answering open-domain *why*-questions. The general set-up of Question Answering (QA) systems consists of at least three steps: (1) question analysis, (2) retrieval of documents or passages that possibly contain the answer, and (3) answer extraction.

In Chapter 1, we stated that *why*-questions require a different approach than factoid questions because their answers tend to be longer and more complex. Moreover, because of the complexity of the problem, *why*-QA is an interesting case study for a linguistically motivated approach to information retrieval. We have investigated several levels of linguistic information (lexico-semantic, syntactic, discourse) in this thesis in order to find out which of these contribute most to improving the performance of our *why*-QA system.

In Chapter 2 and 3 we investigated steps 1 and 3 of the QA process using several levels of linguistic analysis. In Chapter 4 and 5, we developed a system in which step 2 (passage retrieval) is the central component. In this system we combine techniques from Information Retrieval (IR) and Natural Language Processing (NLP).

For evaluation purposes, we developed three different data sets. Two of these are collections of *why*-questions that were formulated for a collection of newspaper articles. For Chapter 2 we used unannotated texts and for Chapter 3 we exploited texts with annotations on the discourse level. The third data set, which we used in Chapter 4 and further, is a set of *why*-questions that were formulated by users of a QA system, and answers that we have found in Wikipedia.

We will first discuss our findings for each of the five subquestions formulated in Chapter 1 before answering our main research question: "What are the possibilities and limitations of an approach to *why*-QA that uses linguistic information in addition

to text retrieval techniques?"

Chapter 2: Linguistic information for question analysis

In Question Answering systems, question analysis is the first step to be performed before document/passage retrieval and answer extraction. In Chapter 2, we investigated question analysis for *why*-questions using different types of linguistic information.

RQ I. What types of linguistic information can play a role in question analysis for *why*-QA?

First, we collected a set of 395 *why*-questions and 769 corresponding answers that we elicited from native speakers who read a number of newspaper texts from Reuters and Guardian.

Following approaches to factoid QA, we created a classification of semantic answer types for *why*-questions. Knowing the answer type of a question facilitates the answer extraction process if the system succeeds to distinguish between different answer types in the source document. We found that the most frequent — and therefore, the most important — answer types for *why*-questions are *cause* and *motivation*.

We exploited information on the syntactic structure of the question in order to predict the semantic answer type. Our question analysis module first assigned to each input question a syntactic category such as 'action question', 'process question' or 'existential *there* question' with the use of a number of linguistic resources: constituency parse trees [71], a set of hand-written rules, verb classes from VerbNet [48] and the Levin verb index [52], and noun classes from WordNet [27]. From these syntactic categories, the module deduced the semantic answer type.

We found that the distinction between cause and motivation questions can largely be predicted from (1) the distinction between action verbs (e.g. *write*) and process verbs (e.g. *grow*), (2) the modality of the auxiliary (e.g. *did* vs. *can*) and (3) the agency of the subject (e.g. *Mr. Bocuse* vs. *class sizes*). Thus, for a question analysis module that predicts the semantic answer type of *why*-questions, external lexico-semantic resources are indispensable.

In order to look-up lexico-semantic information for subjects and verbs, the syntactic constituents need to be extracted from the question. This can be done with a set of rules applied to the output of a parser. In Chapter 2, we manually selected the best parse tree from the parse trees suggested by the TOSCA parser. In Chapter 4 we will see that it is also possible to use fully automatic syntactic parses without a considerable performance loss.

For evaluation, we manually classified 130 *why*-questions from our development set with respect to their semantic answer type. Our question analysis module succeeded in assigning the correct answer type to 62.2% of these questions, the wrong answer type to 2.4%, and no answer type to the other 35.4%. This means that the precision of our question analysis module is high and recall reasonable. However, in our data, the distribution over the answer types was skewed: the causal questions formed the large majority. This means that the relative improvement that can be gained from answer type prediction is limited.

Chapter 3: Discourse analysis for answer extraction

As explained above, the first two steps in the question answering process are question analysis and document/passage retrieval. In Chapter 3, we addressed the third step: answer extraction. Answers to *why*-questions take the form of text passages that give a (possibly implicit) explanation, motivation, elaboration, etc. These types of discourse relations in texts may be made explicit by annotations using the framework of Rhetorical Structure Theory (RST). In Chapter 3 we aimed to answer the following question:

RQ II. To what extent can annotations on the level of rhetorical structure (discourse analysis) help in extracting answers to *why*-questions?

In order to answer this question, we elicited a set of *why*-questions for Wall Street Journal texts that had been manually annotated in the RST Treebank [18]. We implemented an answer extraction module that uses manual annotations for extracting answer passages from these texts. In this set-up we assumed that the document containing the answer was already retrieved from the corpus using a retrieval engine.

Our answer extraction method hypothesized that both the question and its answer correspond to a span of text in the source document. For matching the question to spans of text in the RST annotation, we implemented a language modeling approach that selects text spans with a large proportion of words from the question. Furthermore, the model takes into account the RST relation connected to each text span and the prior probability of this relation type for *why*-answers.

The method assumes that a relevant RST relation holds between the text span representing the question and the text span representing the answer. We selected a number of relation types from the RST relation set which we believed might be relevant for *why*-QA, such as *motivation*, *cause* and *elaboration*. We assumed that these relation types are also related to the distinction between answer types in Chapter 2.

Our answer extraction module then extracted the spans of text connected to the candidate question spans as candidate answers — the candidate answers were ranked by the probability assigned by the language model to the question span. On average, 16.7 candidate answers per question got assigned a probability above a pre-defined threshold.

We evaluated the answer extraction module on 336 *why*-questions formulated to documents from the RST Treebank. A manual analysis of the data showed that the maximum recall that we could reach with our extraction approach was 58.0%: 42.0% of *why*-questions could not be answered following the suggested RST approach. Our answer extraction module reached 91.8% of the maximum recall: for 53.3% of the 336 questions, the module extracted the reference answer from the RST tree of the answer document.

Although these results were promising, the proposed method relied on annotated data. We investigated to what extent we could achieve partial automatic discourse annotations that were specifically equipped to finding answers to *why*-questions. Unfortunately, automatic annotations on the discourse level appeared problematic [91]. Moreover, we found that existing techniques for automatic RST annotation rely on shallow cue phrase matching [87].

Therefore, we shifted our focus in the second half of the thesis to an approach for which shallow text processing techniques suffice.

Chapter 4: Linguistic features for passage ranking

After we had investigated the use of linguistic information for question analysis (Chapter 2) and answer extraction (Chapter 3), we decribed an approach to *why*-QA in Chapter 4 that is centered around the second step of the QA process: passage retrieval.

In the system that we presented in Chapters 4 and 5, we combine off-the-shelf text retrieval technology with linguistic and other structural knowledge of *why*-questions and their answers. For these experiments, we collected a new data set: 186 user-generated *why*-questions from the Webclopedia question set [39], for which we manually found reference answers in the Wikipedia XML corpus [24]. Using these data for development and evaluation purposes, we aimed to answer the following question:

RQ III. To what extent can we improve answer ranking by adding structural linguistic information to a passage retrieval module for *why*-QA and which information from the question and its candidate answers is the most important?

While Chapters 2 and 3 were largely linguistic in nature, Chapters 4 and 5 are more based on Natural Language Processing (NLP). We implemented a pipelined system in which we exploited the Lemur retrieval engine for passage retrieval. In the baseline setting, we transformed the input question to a Lemur query by removing stop words and punctuation and set Lemur to retrieve 150 passages from Wikipedia for each question. The ranking as performed by the TF-IDF ranking model gave us the baseline results: success@150 was 78.5%, success@10 was 45.2% and Mean Reciprocal Rank (MRR) was 0.25.

Since we found that most answers to *why*-questions are paragraph-length passages [104], we refrained from an additional answer extraction step after passage retrieval. Instead, we considered the passages of 500 to 800 characters from Wikipedia to be potential answers, and decided to use knowledge on the structure of *why*-questions and their answers to improve the initial TF-IDF-ranking of the answer passages.

In Chapter 4, we analyzed the output of our baseline system in order to learn the strengths and weaknesses of a bag-of-words approach for *why*-QA. To that end, we studied the questions that were answered in the top 150 by our passage retrieval module but not in the top 10. Based on this analysis, we decided on a set of 37 question and answer features that might distinguish answers from non-answers. We implemented these features as overlap scores for specific question and answer parts. We extracted the feature values from our data using a Perl script and a number of linguistic resources.

By refraining from an additional answer extraction step, we largely disregarded our method for answer type prediction from Chapter 2 and the RST-based extraction approach from Chapter 3. However, we implemented the knowledge that we gained in these chapters as a number of features describing the syntactic structure of the question and a feature for the presence of cue phrases such as *because* in the answer passage.

After having decided on our feature set, we trained a logistic regression model (LRM) for the distinction between answers and non-answers to the *why*-questions in our collection. In the test stage, we used the log odds assigned to each question-answer pair by the model to sort the answers per question. By applying this re-ranking module to our set of candidate answers we got significantly higher scores in terms of MRR (from 0.25 to 0.34) and Success@10 (from 45% to 57%). The output of the LRM showed that only a small subset (eight) of our 37 features significantly contributed to the re-ranking score.

The question constituents that appeared to be the most important were the question focus, the main verb and the direct object. On the answer side, most important

were the title of the document in which the candidate answer was embedded and knowledge on the presence of cue phrases. Since our features were overlap-based, they were relatively light-weight to implement. Moreover, no manual intervention was needed for the extraction of the feature values. For implementation of some of the significant features, a form of syntactic parsing was needed that could identify subject, verb and direct object from the question and sentences in the candidate answers. These constituents could be extracted from the automatically generated output of a syntactic parser. An additional set of rules was needed for finding the question focus. Finally, we needed a fixed list for identifying cue phrases.

Chapter 5: Optimizing answer ranking

In Chapter 5, we optimized our re-ranking module by comparing a number of machine learning techniques for learning to rank the answers, still using the set of 37 features that we developed in Chapter 4.

RQ IV. Which machine learning techniques are the most suitable for learning the ranking of *why*-answers, using a set of linguistically motivated overlap features?

In evaluating machine learning techniques for the task of learning a ranking function, we faced two challenges. First, in the manual annotation of our data we had decided to treat answer relevance as a binary variable, which made the annotation task more feasible. As a result, our ranking function needed to induce a ranked list from binary relevance judgments. The second challenge was the imbalance between positive and negative instances in our training set: 98.6% of the answers in our data set was labeled as incorrect.

We evaluated the following techniques for the task of learning a ranking for *why*-answers: Naive Bayes, Support Vector Classification, Support Vector Regression, Logistic Regression, Ranking SVM, SVM^{map} and a Genetic Algorithm. Following the learning-to-rank literature [56], we considered three different approaches to learning to rank: (1) the so-called pointwise approach, in which candidate answers are classified individually (over all questions), (2) the pairwise approach, in which pairs of two candidate answers to the same question are classified, and (3) the listwise approach, in which the complete ranking of all candidate answers to the same question is optimized.

We found that with all machine learning techniques, we could get to an MRR score that was significantly above the TF-IDF baseline of 0.25 and not significantly

lower than the best score of 0.35. We were able to obtain good results with all three types of approaches for our data: pointwise, pairwise and listwise. The optimum score was reached by Support Vector Regression for the pairwise representation, but some of the pointwise settings reached scores that were not significantly lower than this optimum. We also found that for our imbalanced data set, some of the techniques with hyperparameters heavily depended on tuning. However, if we solved the class imbalance by oversampling the positive instances or presenting the problem as a pairwise classification task then the default hyperparameter values were well applicable to the data and tuning was less important.

Given the experimental result of 0.35 and the theoretical optimum of 0.79 (if for all questions with at least one correct answer a correct answer would be ranked at position 1), we concluded that our features were suboptimal for distinguishing correct from incorrect answers. Since we already invested much time in finding the best features for describing our data (see Chapter 4), we concluded that the problem of distinguishing between answers and non-answers to *why*-questions was more complex than an approach based on textual (overlap) features could solve.

Chapter 6: Limitations of the IR approach to why-QA

In Chapter 6, we reviewed the problem of *why*-QA in detail. We discussed the history of the use of knowledge in QA research, starting with the database systems that were developed in the 1960s, via the expert systems from the 1970s and 1980s to the IR-based approaches that emerged in the 1990s. From Chapter 5 we had concluded that the problem of finding answers to *why*-questions is more complex than an approach based on textual (overlap) features can solve.

RQ V. What are the limitations of an IR approach to *why*-QA and to what extent can models of human text understanding explain the shortcomings of state-of-the art QA systems?

In Chapter 6 we performed an extensive error analysis of our approach to *why*-QA, in which we identified a number of problematic question categories. We found that by far the most important limitation of the IR approach to QA seemed to be the vocabulary gap between question and answer. We related this vocabulary gap problem to human understanding of written language. We considered the situation in which a person is asked to judge whether a passage is a satisfactory answer to a question. From this analysis, we concluded that our system not so much suffered from a *vocabulary gap* but more from a *knowledge gap*. A lexical overlap-based method

cannot recognize associative concept relations between question words and answer words, while human readers can and actually (unconsciously) do exploit associative relations.

In trying to bridge the knowledge gap, we found that most of the semantic relations between question words and answer words in our data were not synonymic or hypernymic but more vaguely associative, as a result of which conventional lexico-semantic resources could not provide us with sufficient information to solve the vocabulary gap.

We made a number of recommendations for system developers that aim to bridge the knowledge gap in future QA systems: First, do not rely exclusively on specific semantic relations such as synonymy and hyperonymy but implement a source of more general associative relations. Second, investigate methods (such Latent Semantic Analysis) that save concepts instead of words in the lexicon. If we make it possible in the future to represent the question and the answer corpus as bags of concepts instead of bags of words, then many implicit word relations will be more explicit. And finally, approach the concept matching problem as a task of concept activation. Words from the question and the answer need not necessarily overlap, as long as they activate (partly) the same set of concepts.

Answering the main research question

As explained in the beginning of this chapter, the general approach to automatic Question Answering (QA) is a pipelined setup with at least three modules: (1) question analysis and query generation, (2) document/passage retrieval and (3) answer extraction. For *why*-questions, we first investigated linguistic aspects of both the questions and their answers, and then chose a set-up in which we used off-the-shelf retrieval and ranking technology. Since the answers to *why*-questions are paragraph-length passages, we replaced the answer extraction task by a re-ranking module that uses linguistic information for improving the answer ranking. We evaluated our method using a set of user-generated open-domain *why*-questions against Wikipedia.

Main question What are the possibilities and limitations of an approach to *why*-QA that uses linguistic information in addition to text retrieval techniques?

Text retrieval techniques with modern ranking models are very powerful in finding and ranking passages that share content with an input query. These techniques represent the query and the answer passage as a bag of words weighted by term frequencies. We found that without taking into account the structure of the questions

and the answers, a state-of-the art retrieval engine is able to retrieve a correct answer in the top-10 of the result list for 45% of the open domain *why*-questions.

Bag-of-words retrieval and ranking has a number of known limitations. For the task of *why*-QA, we found that the most important limitation is that the structure of the questions and the candidate answers is not taken into account. We studied a number of levels of linguistic information on both the side of the question and the side of the answer in order to find out which type of information is the most important for answering *why*-questions.

For the question, we found that predicting the semantic answer type (cause or motivation) is to some extent possible with the use of syntactic annotation. However, this information is only useful if we have a robust method of distinguishing the different answer types in candidate answer passages. Therefore, we investigated the use of discourse-level annotations for answer extraction: rhetorical relations such as explanation, purpose and cause provide useful information for extracting answers to *why*-questions. Although we found the potential value of discourse structure to be reasonable, it appeared that automatically annotating texts with rhetorical relations is still an unsolved problem.

We found that a more powerful approach to *why*-QA is to use an off-the-shelf passage retrieval component and improve over its ranking performance with the use of knowledge-driven, but relatively shallow, overlap features. We implemented a re-ranking module that incorporates the knowledge that we gained about the syntactic structure of *why*-questions and the surface form and the document context of the answers. With this module, we were able to improve significantly over the already quite reasonable bag-of-words baseline. After we optimized the feature combination in a learning-to-rank set-up, our system reached an MRR of 0.35 with a success@10 score of 57%. These scores were reached with only eight overlap features, one of which was the baseline ranker TF-IDF and the others were based on linguistic information and document structure.

Although we reached a significant improvement, the power of our retrieval and ranking system for *why*-questions is still limited: for 43% of the questions that have an answer in the Wikipedia corpus, this answer was not retrieved in the top-10 results. We argued that this seems to be caused by the infamous vocabulary gap between natural language questions and their answers. On the other hand, human readers are perfectly capable of recognizing the correct answers to *why*-questions even if different words are used and there are no explicit explanation-type cue phrases (such as *because*) in the passage. This is because NLP systems can only link words to their synonyms or hypernyms from lexical resources while human readers can recognize all sorts of associative concept relations, even if two concepts are only remotely

related to each other.

In conclusion, high-performance question answering for *why*-questions is still a challenge. The main reason is that the knowledge sources that are currently available for NLP research are too limited to capture the text understanding power that is needed for recognizing the answer to an open-domain *why*-question. Since this capability is problematic for machines but very natural for human readers, the process of *why*-QA deserves renewed attention from the field of artificial intelligence. At the end of chapter 6 we presented several suggestions for the directions of future research.

Nederlandse samenvatting

In dit proefschrift hebben we het automatisch beantwoorden van Engelstalige *waarom*-vragen onderzocht. De gebruikelijke opzet van vraag-antwoord-systemen bestaat uit minimaal drie stappen: (1) analyse van de vraag, (2) retrieval van documenten of passages die mogelijk het antwoord bevatten en (3) antwoord-extractie.

Het uitgangspunt van dit proefschrift, zoals omschreven in Hoofdstuk 1, is dat voor *waarom*-vragen een andere benadering nodig is dan voor zogenaamde *factoid*-vragen omdat de antwoorden op *waarom*-vragen langer en complexer zijn. Daarnaast is het automatisch beantwoorden van *waarom*-vragen vanwege die complexiteit een interessante case voor een taalkundig gemotiveerde benadering van *Information Retrieval*. We hebben in dit proefschrift verschillende niveaus van taalkundige informatie onderzocht (het lexico-semantische, het syntactische en het *discourse*-niveau) om uit te vinden welke hiervan het meest bijdragen aan het beantwoorden van *waarom*-vragen.

In Hoofdstuk 2 en 3 hebben we gewerkt aan respectievelijk stap 1 en 3 van het vraag-antwoord-proces met behulp van verschillende niveaus van taalkundige analyse. In Hoofdstuk 4 en 5 hebben we gewerkt aan een systeem waarin stap 2 (*passage retrieval*) centraal staat. In dit systeem combineren we IR-technieken met *Natural Language Processing (NLP)*.

Voor evaluatiedoeleinden hebben we gewerkt met drie verschillende dataverzamelingen. De eerste twee zijn collecties van *waarom*-vragen die geformuleerd zijn voor een verzameling krantenartikelen. Voor Hoofdstuk 2 waren dit ongeannoteerde teksten en voor Hoofdstuk 3 teksten met annotaties op het discourse-niveau. De derde dataverzameling, die we gebruikt hebben in Hoofdstuk 4 en verder, is een set van *waarom*-vragen die door gebruikers van een vraag-antwoord-systeem zijn geformuleerd, en antwoorden die we hebben gevonden in Wikipedia.

In deze samenvatting bespreken we eerst onze bevindingen met betrekking tot elk van de vijf subvragen uit Hoofdstuk 1, waarna we de hoofdvraag beantwoorden: "Wat zijn de mogelijkheden en beperkingen van een benadering van het beantwoorden van *waarom*-vragen die taalkundige informatie gebruikt ter verbetering van de resultaten van conventionele tekst-retrieval-technieken?"

Hoofdstuk 2: Taalkundige informatie voor vraaganalyse

In vraag-antwoordsystemen is vraaganalyse de eerste stap die wordt uitgevoerd, vóór *document/passage retrieval* en antwoordextractie. In hoofdstuk 2 staat de vraaganalyse voor waarom-vragen centraal. We hebben ons gericht op de volgende vraag:

Vraag I. Welke soorten taalkundige informatie kunnen een rol spelen in de vraaganalyse voor het beantwoorden van *waarom*-vragen?

Om deze vraag te kunnen beantwoorden hebben we eerst een aantal proefpersonen gevraagd om bij een aantal krantenartikelen (geselecteerd uit het Reuters en het Guardian corpus) *waarom*-vragen te formuleren met een antwoord in het artikel. Dit resulteerde in een set van 395 geëliciteerde *waarom*-vragen en 769 corresponderende antwoorden.

Analoog aan bestaande methoden voor het beantwoorden van factoid-vragen (vragen die beginnen met *wie*, *wat*, *waar* of *wanneer*) hebben we een classificatie gemaakt van semantische antwoordtypen voor *waarom*-vragen. In onze dataset bleken *oorzaak* en *motivatie* de meest frequente — en dus de belangrijkste — antwoordtypen voor *waarom*-vragen.

We hebben informatie over de syntactische structuur van de vragen gebruikt om het semantische antwoordtype te voorspellen. Onze module voor vraaganalyse kent eerst aan iedere vraag een syntactische categorie toe, zoals 'actievraag', 'procesvraag' en '*existential there*-vraag'. Dit gebeurt op basis van een aantal linguïstische bronnen: syntactische analyses gegenereerd door een parser [71], een verzameling handgeschreven regels, werkwoordklassen uit *VerbNet* [48] en de *Levin verb index* [52], en naamwoordklassen uit *WordNet* [27]. Uit de automatisch bepaalde syntactische categorie voorspelt onze module het semantische antwoordtype.

We hebben ontdekt het onderscheid tussen *oorzaak*- en *motivatie*-vragen grotendeels voorspeld kan worden uit (1) het onderscheid tussen actiewerkwoorden (bijv. *schrijven*) en proceswerkwoorden (bijv. *groeien*), (2) de modaliteit van het hulpwerkwoord (geen modaliteit vs. bijv. *kunnen*) en (3) het wel of niet menselijk zijn van het onderwerp (bijv. *Mr. Bocuse* vs. *klassengrootten*). Voor een vraaganalysemodule die het semantische antwoordtype van *waarom*-vragen voorspelt zijn externe lexico-semantische bronnen dus onmisbaar.

Om de lexico-semantische informatie voor het onderwerp en de werkwoordelijke groep te kunnen opzoeken, moeten deze taalkundige elementen uit de vraag worden geëxtraheerd. Dit kan gedaan worden met een set van regels die worden toegepast op de uitvoer van een syntactische parser. In Hoofdstuk 2 hebben we handmatig

de beste analyse geselecteerd uit de uitvoer van de TOSCA-parser. In Hoofdstuk 4 zullen we zien dat deze syntactische analyse ook geheel automatisch kan worden gedaan zonder een aanzienlijk verlies van kwaliteit.

Voor de evaluatie van de analysemodule hebben we 130 *waarom*-vragen uit onze collectie handmatig geclassificeerd naar semantische antwoordtypen. Onze analysemodule kende het correcte antwoordtype toe aan 62,2% van de vragen, het verkeerde antwoordtype aan 2,4% van de vragen en geen antwoordtype aan de overige 35,4%. Dat betekent dat de precisie van onze analysemodule hoog is en de *recall* redelijk. In onze data was de verdeling van de vragen over de antwoordtypen echter scheef: een grote meerderheid van de vragen had als antwoordtype *oorzaak*. Als gevolg daarvan kan slechts een beperkte relatieve verbetering gehaald worden uit de voorspelling van het antwoordtype.

Hoofdstuk 3: Discourse-analyse voor antwoordextractie

Zoals hierboven opgemerkt zijn de eerste twee stappen van het vraag-antwoord-proces vraaganalyse en *document/passage retrieval*. In Hoofdstuk 3 richtten we ons op de derde stap: antwoordextractie. Antwoorden op *waarom*-vragen zijn tekstfragmenten die een (mogelijk impliciete) verklaring, motivatie, uitweiding, etc. geven. Dergelijke *discourse*-relaties in teksten kunnen expliciet worden gemaakt door annotaties uit de *Rhetorical Structure Theory* (RST) toe te passen. In Hoofdstuk 3 stond de volgende vraag centraal:

Vraag II. In welke mate kunnen annotaties op het niveau van de retorische structuur (discourse-analyse) helpen bij het extraheren van antwoorden op *waarom*-vragen?

Om deze vraag te beantwoorden hebben we *waarom*-vragen geëliciteerd bij *Wall Street Journal*-teksten die zijn voorzien van handmatige annotaties in de RST Treebank [18]. We hebben een module geïmplementeerd voor de extractie van de antwoorden uit die teksten. In deze opzet zijn we ervan uitgegaan dat het relevante antwoorddocument al is opgeleverd door de tweede stap in het vraag-antwoordsysteem, de *retrieval engine*.

Om het antwoord op een *waarom*-vraag uit de RST-analyse van het antwoorddocument te kunnen extraheren, zouden zowel vraag als antwoord moeten corresponderen met tekstfragmenten in het antwoorddocument. Voor het matchen van de vraag op tekstfragmenten uit de RST-annotatie hebben we een taalmodel geïmplementeerd dat tekstfragmenten selecteert die een relatief groot aantal woorden gemeen hebben

met de vraag. Bovendien houdt het model rekening met het type RST-relatie waarin elk tekstfragment participeert en de a priori waarschijnlijkheid van dit relatietype voor de antwoorden op *waarom*-antwoorden.

Deze methode toetst de hypothese dat er een relevante RST-relatie bestaat tussen het tekstfragment dat bij de vraag hoort en het tekstfragment dat bij het antwoord hoort. Daartoe selecteerden we een aantal RST-relaties waarvan we aannamen dat ze relevant zijn voor *waarom*-vragen, zoals *explanation* (verklaring), *cause* (oorzaak) en *elaboration* (uitweiding). We namen aan dat deze relatietypen ook verband houden met het onderscheid tussen antwoordtypen in Hoofdstuk 2.

Onze module extraheerde vervolgens als kandidaatantwoorden de tekstfragmenten die via een relevante RST-relatie verbonden waren met de vraagteksten. Daarbij werden de kandidaatantwoorden gerangschikt naar de score die het taalmodel had toegekend aan de vraagteksten. Gemiddeld kregen 16,7 kandidaatantwoorden per vraag een score die boven een vooraf bepaalde drempelwaarde lag.

We evalueerden onze module voor antwoordextractie op 336 *waarom*-vragen die door proefpersonen waren geformuleerd voor documenten uit de RST-Treebank. Een handmatige analyse van de data toonde aan dat de maximale recall met onze methode 58,0% was: 42,0% van de *waarom*-vragen kon niet beantwoord worden met de voorgestelde RST-benadering. Onze extractiemodule behaalde 91,8% van deze maximale recall: voor 53,3% van de 336 vragen vond de extractiemodule het referentieantwoord in de RST-analyse van het brondocument

Deze resultaten lijken op het eerste gezicht veelbelovend, maar onze methode is gebaseerd op vooraf geannoteerde data. We hebben onderzocht in hoeverre we automatische discourse-annotaties konden verkrijgen die waren toegesneden op *waarom*-vragen. Automatische annotatie bleek echter onhaalbaar [91]. Bovendien vonden we dat veel bestaande technieken voor automatische RST-annotatie voornamelijk gebaseerd zijn op de herkenning van *cue phrases* [87].

Daarom is onze aandacht in de tweede helft van het onderzoek verschoven naar een aanpak met eenvoudigere tekstanalysetechnieken.

Hoofdstuk 4: Taalkundige features voor passage ranking

Nadat we de mogelijkheden hadden onderzocht voor het gebruik van taalkundige informatie voor vraaganalyse en antwoordextractie, hebben we in Hoofdstuk 4 een benadering voor het beantwoorden van *waarom*-vragen beschreven waarin de tweede stap van het vraag-antwoord-proces centraal staat: *passage retrieval*.

Het systeem dat we hebben gepresenteerd in Hoofdstuk 4 en 5 combineert bestaande *text retrieval*-technologie met taalkundige en andere structurele kennis van *waar-*

om-vragen en hun antwoorden. Voor deze experimenten hebben we een nieuwe dataset gebruikt: 186 *waarom*-vragen uit de Webclopedia-data [39], geformuleerd door gebruikers van een operationeel vraag-antwoordsysteem. Hiervoor hebben we handmatig referentie-antwoorden gezocht in het Wikipedia XML-corpus [24].

Vraag III. In hoeverre kunnen we de ranking van antwoorden verbeteren door het toevoegen van structurele taalkundige informatie aan een *passage retrieval*-module voor *waarom*-vragen en welke informatie uit de vraag en de kandidaatantwoorden is het belangrijkst?

Waar Hoofdstuk 2 en 3 grotendeels taalkundig georiënteerd waren, zijn Hoofdstuk 4 en 5 gebaseerd op *Natural Language Processing*-technieken. We hebben een systeem geïmplementeerd waarin we gebruik maakten van de *Lemur Retrieval Engine* voor het verkrijgen van antwoordpassages. In de baseline-instelling werd de vraag omgezet naar een Lemur-query door het verwijderen van stopwoorden en interpunctie en werd Lemur ingesteld om voor elke vraag 150 antwoordpassages uit Wikipedia te zoeken. De ranglijst van antwoorden zoals gegenereerd door het woordgebaseerde *TF-IDF ranking model* gaf ons de baseline-resultaten: succes@150 was 78,5%, succes@10 was 45,2% en *Mean Reciprocal Rank* (MRR) was 0,25.

Gezien het feit dat de meeste antwoorden op *waarom*-vragen passages zijn die de lengte hebben van een alinea [104], zagen we af van de stap voor de extractie van antwoorden uit de passages die Lemur gevonden had. In plaats daarvan hebben we de passages uit Wikipedia (die 500 tot 800 tekens lang waren) beschouwd als mogelijke antwoorden, en besloten om kennis over de structuur van *waarom*-vragen en hun antwoorden te gebruiken om de TF-IDF-ranking van antwoordpassages te verbeteren.

In Hoofdstuk 4 hebben we de output van ons baseline-systeem geanalyseerd om de sterke en zwakke punten van een woordgebaseerde aanpak voor het beantwoorden van *waarom*-vragen te achterhalen. Daartoe bestudeerden we de vragen waarvoor onze passage-retrieval-module een correct antwoord gevonden had in de top 150, maar niet in de top 10 van de resultatenlijst. Op basis van deze analyse en onze bevindingen uit hoofdstuk 2 en 3 hebben we een set van 37 features geïdentificeerd die de vraag- en antwoordinformatie bevatten om de correcte antwoorden te kunnen onderscheiden van de incorrecte antwoorden. We hebben deze features geïmplementeerd als overlap-scores voor specifieke onderdelen van de vraag en de kandidaat-antwoorden. De waarden van de features hebben we uit onze data geëxtraheerd met behulp van een Perl-script en een aantal taalkundige bronnen.

Door af te zien van de antwoordextractie-stap hebben we onze methode voor het voorspellen van het antwoordtype uit Hoofdstuk 2 en de RST-gebaseerde benadering

uit Hoofdstuk 3 grotendeels buiten beschouwing gelaten. We hebben echter wel de kennis die we hebben opgedaan in deze hoofdstukken geïmplementeerd in de vorm van features die de syntactische structuur van de vraag beschrijven en een feature voor de aanwezigheid van cue phrases zoals *because* ('omdat') in het antwoord.

Na het vaststellen van de featureset hebben we een logistisch regressiemodel (LRM) getraind om het onderscheid tussen de antwoorden en de niet-antwoorden op de *waarom*-vragen in onze collectie te leren. In de testfase gebruikten we de *log odds* die door het model waren toegekend aan elk vraag-antwoordpaar om de antwoorden per vraag te sorteren. Door toepassing van deze *re-ranking module* op onze set van kandidaatantwoorden verkregen we significant hogere scores in termen van MRR (van 0,25 naar 0,34) en Succes@10 (van 45% naar 57%).

Uit de uitvoer van het LRM bleek dat slechts een klein deel (8) van onze 37 features significant bijdroeg aan de *re-ranking score*. De onderdelen van de vraag die het belangrijkste bleken voor de ranking van de antwoorden waren de focus van de vraag, het hoofdwerkwoord en het lijdend voorwerp. Aan de antwoordkant waren de belangrijkste onderdelen de titel van het document waarin het kandidaatantwoord was ingebed en de aanwezigheid van *cue phrases* zoals *because* ('omdat'). Omdat onze features gebaseerd waren op termoverlap waren ze relatief eenvoudig te implementeren. Bovendien waren voor het verkrijgen van de featurewaarden geen handmatige analyses nodig. Voor de implementatie van sommige belangrijke features was een vorm van syntactische analyse nodig die het onderwerp, hoofdwerkwoord en lijdend voorwerp kon identificeren in de vraag en in de kandidaat-antwoorden. Deze elementen konden geëxtraheerd worden uit de automatisch gegenereerde uitvoer van een syntactische parser. Een aantal aanvullende regels waren nodig voor het vinden van de vraagfocus. Tot slot hadden we een lijst van cue phrases voor *waarom*-antwoorden nodig.

Hoofdstuk 5: De ranking van de antwoorden optimaliseren

In Hoofdstuk 5 hebben we onze re-ranking-module geoptimaliseerd door een aantal *machine learning*-technieken voor het leren ranken van de antwoorden met elkaar te vergelijken. Hierbij hebben we steeds de set van 37 features gebruikt die we in Hoofdstuk 4 hadden samengesteld.

Vraag IV. Welke *machine learning*-technieken zijn het meest geschikt om de ranking van de *waarom*-antwoorden te leren, met gebruikmaking van een set taalkundig gemotiveerde overlap-features?

Bij het evalueren van *machine learning*-technieken voor het leren van de rankingfunctie kwamen we twee uitdagingen tegen. Ten eerste hadden we in de handmatige annotatie van de data besloten om de correctheid van de antwoorden als een binaire variabele te beschouwen, opdat de annotatietaak beter haalbaar was. Het gevolg daarvan was dat onze rankingfunctie een geordende lijst moest afleiden uit binaire relevantie-oordelen. De tweede uitdaging was dat de verhouding tussen positieve en negatieve voorbeelden in onze trainingset sterk uit balans was: 98,6% van de antwoorden in onze data was geannoteerd als irrelevant.

We hebben de volgende technieken geëvalueerd voor het leren van de antwoordranking: *Naive Bayes*, *Support Vector Classification*, *Support Vector Regression*, LRM, *Ranking SVM*, SVM^{map} en een Genetisch Algoritme. Aan de hand van de literatuur over 'learning to rank' beschouwden we drie verschillende benaderingen van het rankingprobleem: (1) de zogenoemde *pointwise*-benadering, waarin kandidaatantwoorden individueel geclassificeerd worden (ongeacht de clustering per vraag), (2) de *pairwise*-benadering, waarin paren van kandidaatantwoorden voor dezelfde vraag worden geclassificeerd, en (3) de *listwise*-benadering, waarin de complete ranking van alle kandidaatantwoorden op dezelfde vraag wordt geoptimaliseerd.

We ontdekten dat we met alle *machine learning*-technieken een MRR-score konden bereiken die significant hoger was dan de TF-IDF baseline van 0,25 en niet significant lager dan de beste score van 0,35. We behaalden goede resultaten met zowel de pointwise-, de pairwise- als de listwise-benadering. De optimale score werd bereikt door Support Vector Regression in de pairwise-benadering, maar sommige van de pointwise-settings behaalden scores die niet significant lager waren dan het optimum. We ontdekten ook dat sommige technieken erg afhankelijk zijn van de tuning van hyperparameters voor onze sterk ongebalanceerde data. Als we echter de onbalans oplosten door de correcte antwoorden vaker op te nemen in de trainingset of door het probleem als een pairwise-classificatie-taak te presenteren, bleken de standaardwaarden van de hyperparameters wel geschikt voor onze data en was tuning minder belangrijk.

Gezien het experimentele resultaat van $MRR = 0,35$ en het theoretische optimum van $MRR = 0,79$ (als voor alle vragen met minimaal één correct antwoord een correct antwoord op positie 1 gezet zou worden) concludeerden we dat onze features niet in staat zijn om altijd onderscheid tussen correcte en incorrecte antwoorden te maken. Aangezien we al veel tijd hadden geïnvesteerd in het vinden van de meest geschikte features om onze data te beschrijven (zie Hoofdstuk 4), concludeerden we dat het onderscheid tussen antwoorden en niet-antwoorden op *waarom*-vragen zo complex is dat een benadering op basis van tekstuele overlap-features niet de totale oplossing kan leveren.

Hoofdstuk 6: Beperkingen van een IR-benadering voor het beantwoorden van waarom-vragen

In Hoofdstuk 6 hebben we het beantwoorden van *waarom*-vragen in detail geanalyseerd. We hebben de geschiedenis van het gebruik van kennis in vraag-antwoordsystemen besproken, beginnend bij de databasesystemen uit de jaren 60, via de expertsystemen uit de jaren 70 en 80, naar de IR-gebaseerde benaderingen die opkwamen in de jaren 90. In Hoofdstuk 5 hadden we geconstateerd dat het beantwoorden van *waarom*-vragen te complex is voor een benadering die is gebaseerd op tekstuele overlap-features.

Vraag V. Wat zijn de beperkingen van een IR-benadering van *waarom*-vragen en in hoeverre kunnen modellen van menselijk tekstbegrip de tekortkomingen van moderne vraag-antwoordsystemen verklaren?

In Hoofdstuk 6 hebben we een uitgebreide foutenanalyse uitgevoerd van onze benadering voor het beantwoorden van *waarom*-vragen, waarin we een aantal problematische vraagcategorieën hebben geïdentificeerd. We ontdekten dat de lexicale kloof tussen vraag en antwoord met afstand de belangrijkste beperking van de IR-benadering van vraag-antwoordsystemen is. We hebben deze lexicale kloof gerelateerd aan het menselijke begrip van geschreven taal. We beschouwden de situatie waarin iemand wordt gevraagd om te beoordelen of een tekstfragment een bevredigend antwoord op een gegeven vraag is. Uit deze analyse concludeerden we dat ons systeem niet zozeer last had van een *lexicale kloof*, maar meer van een *kennisklоof*. Een methode die stoelt op lexicale overlap kan associatieve relaties tussen concepten in de vraag en concepten in het antwoord niet herkennen, terwijl menselijke lezers dit wel kunnen en die associatieve relaties ook (onbewust) gebruiken tijdens het lezen en interpreteren van tekst.

In onze pogingen om de kennisklоof te overbruggen ontdekten we dat de meeste semantische relaties tussen woorden uit de vraag en woorden uit het antwoord geen synoniem- of hyperoniem-relaties waren, maar veel vagere associatieve relaties. Als gevolg daarvan konden de conventionele lexico-semantische bronnen ons niet de benodigde informatie verschaffen om de lexicale kloof te overbruggen. Als deze analyse correct is, betekent het ook dat er geen grote bijdrage aan de prestaties van het automatisch beantwoorden van *waarom*-vragen te verwachten is van betere of meer gedetailleerde syntactische analyses van de vragen en de kandidaatantwoorden.

Hieruit volgen een aantal aanbevelingen voor systeemontwikkelaars die de kennisklоof in toekomstige vraag-antwoordsystemen willen overbruggen: Ten eerste,

vertrouw niet uitsluitend op semantische relaties zoals synonymie en hyperonymie maar implementeer een bron van algemenere associatieve relaties. Ten tweede, onderzoek de mogelijkheden van methodes die concepten in plaats van woorden opslaan in het lexicon (zoals *Latent Semantic Analysis*). Als het in de toekomst mogelijk is om een vraag en een antwoordcorpus te representeren als een 'bag of concepts' in plaats van een 'bag of words' dan worden impliciete relaties tussen woorden explicieter gemaakt. En ten slotte, benader het concept-matching-probleem als een conceptactivatietaak: woorden in de vraag en het antwoord hoeven niet noodzakelijkerwijs te overlappen, zolang ze maar (deels) dezelfde set van concepten activeren.

Het antwoord op de hoofdvraag

Zoals in het begin van dit hoofdstuk vermeld werken in de algemene benadering van het automatisch beantwoorden van vragen minimaal drie modules na elkaar: (1) vraaganalyse, (2) document/passage retrieval en (3) antwoordextractie. We hebben voor *waarom*-vragen de taalkundige aspecten van zowel de vragen als de antwoorden geanalyseerd, en daarna een opzet gekozen waarin we bestaande technologie voor retrieval en ranking van tekstpassages gebruiken. Aangezien de antwoorden op *waarom*-vragen tekstpassages met de lengte van een alinea zijn, hebben we antwoordextractie vervangen door een re-ranking-module die gebruik maakt van taalkundige informatie voor verbetering van de ranking van de antwoorden. We hebben onze methode geëvalueerd met behulp van een set *waarom*-vragen die door gebruikers van een operationeel vraag-antwoordsysteem geformuleerd waren, met Wikipedia als antwoordcorpus.

Hoofdvraag Wat zijn de mogelijkheden en beperkingen van een benadering van het beantwoorden van *waarom*-vragen die taalkundige informatie gebruikt ter verbetering van de resultaten van conventionele tekst-retrieval-technieken?

Tekst-retrieval-technieken met moderne rankingmodellen zijn erg krachtig in het vinden en ranken van tekstfragmenten die lexicaal overlappen met de query. Deze technieken representeren de query en de antwoordpassage als een *bag of words* die wordt gewogen met woordfrequenties. We ontdekten dat een moderne retrieval-techniek voor 45% van de *waarom*-vragen een correct antwoord in de top-10 van de resultatenlijst weet te plaatsen, zonder de structuur van de vragen en de antwoorden in acht te nemen.

Woordgebaseerde retrieval en ranking heeft een aantal bekende beperkingen. Voor het beantwoorden van *waarom*-vragen bleek de belangrijkste beperking dat

er geen rekening wordt gehouden met de structuur van de vragen en de kandidaat-antwoorden. We hebben een aantal niveaus van taalkundige informatie onderzocht aan zowel de kant van de vraag als de kant van het antwoord, met als doel uit te vinden welk soort informatie het belangrijkste is bij het beantwoorden van *waarom*-vragen.

Voor de vraag vonden we dat het voorspellen van het semantische antwoordtype (oorzaak of motivatie) tot op zekere hoogte mogelijk is met behulp van syntactische annotatie. Deze informatie is echter alleen nuttig als we ook een robuuste methode zouden hebben om onderscheid te maken tussen de verschillende antwoordtypen in de kandidaatantwoorden. Daartoe hebben we het gebruik van annotaties op het discourse-niveau onderzocht: retorische relaties zoals verklaring, doel en oorzaak bleken nuttige informatie te leveren voor het extraheren van antwoorden op *waarom*-vragen. Hoewel we concludeerden dat de toegevoegde waarde van discourse-structuur voor antwoordextractie in theorie redelijk hoog is, bleek dat het automatisch annoteren van teksten met retorische relaties een nog onopgelost probleem is.

We ontdekten dat het een krachtigere benadering voor het beantwoorden van *waarom*-vragen is om gebruik te maken van een bestaande passage-retrieval-module en het verbeteren van de kwaliteit van de ranking van opgeleverde kandidaatantwoorden door middel van kennisgebaseerde maar relatief eenvoudig te extraheren overlap-features. We hebben een re-ranking-module geïmplementeerd die gebruik maakt van kennis over de syntactische structuur van *waarom*-vragen en de algemene structuur van de antwoorddocumenten. Met deze module behaalden we een significante verbetering van de (al redelijk goede) 'bag-of-words'-baseline. Nadat we de combinatie van de features hadden geoptimaliseerd in een 'learning-to-rank'-experiment, behaalde ons systeem een MRR van 0,35 met een success@10-score van 57%. Deze scores werden bereikt met gebruik van slechts acht overlap-features, waaronder de baseline-ranking TF-IDF en zeven features die de structuur van de vragen en de antwoorden beschrijven.

Ondanks dat we een significante verbetering behaalden, was de algehele kwaliteit van ons retrieval- en rankingsysteem voor *waarom*-vragen nog steeds beperkt: voor 43% van de vragen die een antwoord in het Wikipedia-corpus hadden kon ons systeem dit antwoord niet in de top-10 van de resultaten plaatsen. We beargumenteerden dat de oorzaak hiervan de beruchte lexicale kloof tussen vragen en antwoorden was. Aan de andere kant zijn menselijke lezers goed in staat om de correcte antwoorden op *waarom*-vragen te herkennen, ook als het antwoord andere woorden bevat dan de vraag en er geen expliciete *cue phrases* zoals *because* ('omdat') in het antwoord staan. Dit komt doordat NLP-systemen alleen woorden kunnen relateren aan hun synoniemen en hyperoniemen uit een lexicale bron, terwijl menselijke lezers alle

soorten associatieve relaties tussen concepten gebruiken, ook als de concepten alleen impliciet en vaag aan elkaar gerelateerd zijn.

Samenvattend kunnen we zeggen dat het nauwkeurig beantwoorden van *waarom*-vragen nog steeds een uitdaging is. De belangrijkste reden hiervoor is dat bestaande kennisbronnen voor NLP-onderzoek te beperkt zijn om het tekstbegrip te realiseren dat nodig is om een antwoord op een groot deel van de *waarom*-vragen te kunnen herkennen. Aangezien dit vermogen problematisch is voor machines maar relatief gemakkelijk voor menselijke lezers, verdient het proces van het beantwoorden van *waarom*-vragen hernieuwde aandacht uit het veld van kunstmatige intelligentie. Aan het einde van Hoofdstuk 6 hebben we een aantal suggesties voor toekomstig onderzoek gedaan.

Bibliography

[1] R. Akbani, S. Kwek, and N. Japkowicz. Applying support vector machines to imbalanced datasets. In *Lecture Notes in Computer Science: Machine Learning: ECML 2004*, volume 3201, pages 39–50. Springer, 2004.

[2] J. A. Aslam, E. Kanoulas, V. Pavlu, S. Savev, and E. Yilmaz. Document selection methodologies for efficient and effective learning-to-rank. In *Proceedings of the 32nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 468–475, New York, NY, USA, 2009. ACM.

[3] R.H. Baayen, R. Piepenbrock, and H. van Rijn. *The CELEX Lexical Database (CD-ROM)*. Linguistic Data Consortium, University of Pennsylvania, Philadelphia, USA, 1993.

[4] S. Banerjee and T. Pedersen. An adapted Lesk algorithm for word sense disambiguation using WordNet. In *Lecture Notes In Computer Science: Computational Linguistics and Intelligent Text Processing*, pages 136–145. Springer, 2002.

[5] S. Banerjee and T. Pedersen. Extended gloss overlaps as a measure of semantic relatedness. In *International Joint Conference on Artificial Intelligence*, volume 18, pages 805–810. Lawrence Erlbaum Associates Hillsdale, NJ, 2003.

[6] A. Berger, R. Caruana, D. Cohn, D. Freitag, and V. Mittal. Bridging the lexical chasm: statistical approaches to answer-finding. In *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 192–199, New York, NY, USA, 2000. ACM.

[7] M.W. Bilotti, B. Katz, and J. Lin. What works better for question answering: Stemming or morphological query expansion. In *Proceedings of the Workshop on Information Retrieval for Question Answering (IR4QA) at SIGIR 2004*, 2004.

[8] M.W. Bilotti, P. Ogilvie, J. Callan, and E. Nyberg. Structured retrieval for question answering. In *Proceedings of the 30th Annual International ACM*

SIGIR Conference on Research and Development in Information Retrieval, pages 351–358, New York, NY, USA, 2007. ACM.

[9] W. Bosma. Query-Based Summarization Using Rhetorical Structure Theory. In T. Van der Wouden, M. Poß, H. Reckman, and C. Cremers, editors, *Proceedings of the 15th Meeting of Computational Linguistics in the Netherlands (CLIN 2004)*, pages 29–44. LOT, 2005.

[10] E. Breck, J. Burger, L. Ferro, D. House, M. Light, and I. Mani. A sys called Qanda. *NIST Special Publication*, pages 499–506, 2000.

[11] S. Bromberger. Why-Questions. *Mind and Cosmos*, pages 86–111, 1966.

[12] S. Bromberger. *On what we Know we don't Know*. University of Chicago Press, 1992.

[13] S. Buchholz. Using grammatical relations, answer frequencies and the World Wide Web for TREC question answering. *NIST special publication*, pages 502–509, 2002.

[14] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. Learning to rank using gradient descent. In *Proceedings of the 22nd International Conference on Machine Learning*, pages 89–96, 2005.

[15] L. Burnard. Users reference guide for the British National Corpus. Technical report, Oxford University Computing Services, 2000.

[16] Z. Cao, T. Qin, T.Y. Liu, M.F. Tsai, and H. Li. Learning to rank: from pairwise approach to listwise approach. In *Proceedings of the 24th international conference on Machine learning*, pages 129–136, New York, NY, USA, 2007. ACM.

[17] L. Carlson and D. Marcu. *Discourse Tagging Reference Manual*. Univ. of Southern California/Information Sciences Institute, 2001.

[18] L. Carlson, D. Marcu, and M. Okurowski. Building a discourse-tagged corpus in the framework of rhetorical structure theory. In *Proceedings of the Second SIGdial Workshop on Discourse and Dialogue*, pages 1–10. Association for Computational Linguistics, 2001.

[19] B. Carterette and D. Petkova. Learning a ranking from pairwise preferences. In *Proceedings of the 29th Annual International ACM SIGIR Conference on*

Research and Development in Information Retrieval, pages 629–630, New York, NY, USA, 2006. ACM.

[20] S. Chakrabarti, R. Khanna, U. Sawant, and C. Bhattacharyya. Structured Learning for Non-Smooth Ranking Losses. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge Discovery and Data Mining*, pages 88–96, New York, NY, USA, 2008. ACM.

[21] E. Charniak. A maximum-entropy-inspired parser. *ACM International Conference Proceedings Series*, 4:132–139, 2000.

[22] D. Cossock and T. Zhang. Subset ranking using regression. In *Lecture Notes in Computer Science: Learning Theory*, volume 4005, page 605. Springer, 2006.

[23] S. Deerwester, S.T. Dumais, G.W. Furnas, T.K. Landauer, and R. Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science and Technology*, 41(6):391–407, 1990.

[24] L. Denoyer and P. Gallinari. The Wikipedia XML corpus. *ACM SIGIR Forum*, 40(1):64–69, 2006.

[25] R. Engelmore. *Knowledge-based systems in Japan*. International Technology Research Institute, 1993.

[26] W. Fan, E.A. Fox, P. Pathak, and H. Wu. The Effects of Fitness Functions on Genetic Programming-Based Ranking Discovery for Web Search. *Journal of the American Society for Information Science and Technology*, 55(7):628–636, 2004.

[27] C. Fellbaum, editor. *WordNet: An Electronic Lexical Database*. MIT Press, Cambridge, MA, USA, 1998.

[28] O. Ferret, B. Grau, M. Hurault-Plantet, G. Illouz, L. Monceaux, I. Robba, and A. Vilnat. Finding an answer based on the recognition of the question focus. *NIST Special Publication*, pages 362–370, 2002.

[29] P. Forner, A. Peñas, E. Agirre, I. Alegria, C. Forăscu, N. Moreau, P. Osenova, P. Prokopidis, P. Rocha, and B. Sacaleanu. Overview of the CLEF 2008 Multilingual Question Answering Track. In *Evaluating Systems for Multilingual and Multimodal Information Access 9th Workshop of the Cross-Language Evaluation Forum, CLEF*, pages 17–19, 2008.

[30] Christopher Fox. A stop list for general text. *SIGIR Forum*, 24(1-2):19–21, 1989.

[31] Y. Freund, R. Iyer, R.E. Schapire, and Y. Singer. An efficient boosting algorithm for combining preferences. *The Journal of Machine Learning Research*, 4:933–969, 2003.

[32] J. Furnkranz and E. Hullermeier. Pairwise Preference Learning and Ranking. In *Lecture Notes in Computer Science: Machine Learning: ECML 2003*, pages 145–156. Springer, 2003.

[33] D.E. Goldberg and J.H. Holland. Genetic Algorithms and Machine Learning. *Machine Learning*, 3(2):95–99, 1988.

[34] B.F. Green Jr, A.K. Wolf, C. Chomsky, and K. Laughery. Baseball: an automatic question-answerer. *AIEE-IRE*, pages 219–224, 1961.

[35] S.M. Harabagiu, M.A. Paşca, and S.J. Maiorano. Experiments with open-domain textual question answering. In *Proceedings of the 18th conference on Computational linguistics-Volume 1*, pages 292–298. Association for Computational Linguistics, 2000.

[36] R. Herbrich, T. Graepel, and K. Obermayer. Large margin rank boundaries for ordinal regression. In *Proceedings of the 8th ACM SIGKDD international conference on Knowledge Discovery and Data Mining*, pages 133–142, New York, NY, USA, 2002. ACM.

[37] R. Higashinaka and H. Isozaki. Corpus-based Question Answering for Why-Questions. In *Proceedings of the International Joint Conference on Natural Language Processing (IJCNLP)*, pages 418–425, 2008.

[38] E.H. Hovy, L. Gerber, U. Hermjakob, C-J Lin, and D. Ravichandran. Toward Semantics-Based Answer Pinpointing. In *Proceedings of the DARPA Human Language Technology Conference (HLT)*, San Diego, CA, USA, 2001.

[39] E.H. Hovy, U. Hermjakob, and D. Ravichandran. A Question/Answer Typology with Surface Text Patterns. In *Proceedings of the Human Language Technology conference (HLT)*, pages 247–251, San Diego, CA, USA, 2002.

[40] C.W. Hsu, C.C. Chang, C.J. Lin, et al. A Practical Guide to Support Vector Classification. Technical report, Department of Computer Science, National Taiwan University, Taipei 106, Taiwan, 2003.

[41] T. Huong and G. Abeysinghe. A Study to Improve the Efficiency of a Discourse Parsing System. In *Proceedings of CICLing-03*, pages 104–117. Springer, 2003.

[42] N. Japkowicz and S. Stephen. The class imbalance problem: A systematic study. *Intelligent Data Analysis*, 6(5):429–449, 2002.

[43] V. Jijkoun and M. De Rijke. Retrieving Answers from Frequently Asked Questions Pages on the Web. In *Proceedings of CIKM-2005*, 2005.

[44] T. Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the 8th ACM SIGKDD international conference on Knowledge Discovery and Data Mining*, pages 133–142, New York, NY, USA, 2002. ACM.

[45] T. Joachims, L. Granka, B. Pan, H. Hembrooke, and G. Gay. Accurately interpreting clickthrough data as implicit feedback. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 154–161, New York, NY, USA, 2005. ACM.

[46] J. Kamps and M. Koolen. Is Wikipedia link structure different? In *Proceedings of the Second ACM International Conference on Web Search and Data Mining*, pages 232–241, New York, NY, USA, 2009. ACM.

[47] M. Khalid and S. Verberne. Passage Retrieval for Question Answering using Sliding Windows. In *Proceedings of the workshop Information Retrieval for Question Answering (IR4QA) at COLING*, 2008.

[48] K. Kipper, H.T. Dang, and M. Palmer. Class-based construction of a verb lexicon. In *Proceedings of the National Conference on Artificial Intelligence*, pages 691–696. AAAI Press, 2000.

[49] C.H.A. Koster and J.G. Beney. Phrase-Based Document Categorization Revisited. In *Proceedings of the 2nd International Workshop on Patent Information Retrieval (PaIR'09)*, 2009.

[50] J.M. Kupiec. MURAX: Finding and Organizing Answers from Text Search. In T. Strzalkowski, editor, *Natural Language Information Retrieval*, pages 311–332. Kluwer Academic Publishers, Dordrecht, the Netherlands, 1999.

[51] W. Lehnert. Human and computational question answering. *Cognitive Science*, 1(1):47–73, 1977.

[52] B. Levin. English verb classes and alternations: A preliminary investigation. Technical report, Department of Linguistics, Northwestern University, Evanston, IL, 1993.

[53] D. Lin. Automatic retrieval and clustering of similar words. In *Proceedings of the 17th international conference on Computational Linguistics*, pages 768–774. Association for Computational Linguistics, 1998.

[54] J. Lin, D. Quan, V. Sinha, K. Bakshi, D. Huynh, B. Katz, and D.R. Karger. What makes a good answer? The role of context in question answering. In *Proceedings of the International Conference on Human-Computer Interaction (INTERACT'03)*, pages 25–32, 2003.

[55] K.C. Litkowski. CL Research Experiments in TREC-10 Question Answering. In *The Tenth Text Retrieval Conference (TREC 2001). NIST Special Publication*, pages 500–250, 2002.

[56] T.Y. Liu. Learning to Rank for Information Retrieval. *Foundations and Trends in Information Retrieval*, 3(3):225–331, 2009.

[57] T.Y. Liu, J. Xu, T. Qin, W. Xiong, and H. Li. Letor: Benchmark dataset for research on learning to rank for information retrieval. In *Proceedings of the Workshop on Learning to Rank for Information Retrieval (LR4IR) at SIGIR 2007*, pages 3–10, 2007.

[58] C. Macleod, R. Grishman, A. Meyers, L. Barrett, and R. Reeves. Nomlex: A lexicon of nominalizations. In *Proceedings of the 8th International Congress of the European Association for Lexicography*, pages 187–193, 1998.

[59] G.S. Mann. Fine-grained proper noun ontologies for question answering. In *International Conference On Computational Linguistics*, pages 1–7. Association for Computational Linguistics Morristown, NJ, USA, 2002.

[60] W.C. Mann and S.A. Thompson. Rhetorical Structure Theory: Toward a Functional Theory of Text Organization. *Text*, 8(3):243–281, 1988.

[61] D. Marcu and A. Echihabi. An Unsupervised Approach to Recognizing Discourse Relations. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 368–375. Association for Computational Linguistics, 2001.

[62] M.T. Maybury. Toward a Question Answering Roadmap. In *New Directions in Question Answering*, pages 8–11, 2003.

[63] A. Meyers, C. Macleod, R. Yangarber, R. Grishman, L. Barrett, and R. Reeves. Using NOMLEX to produce nominalization patterns for information extraction. In *Proceedings: the Computational Treatment of Nominals, Montreal, Canada*, volume 2, pages 25–32, 1998.

[64] D. Milne. Computing Semantic Relatedness using Wikipedia Link Structure. In *Proceedings of the New Zealand Computer Science Research Student conference (NZCSRSC07), Hamilton, New Zealand*, 2007.

[65] D. Moldovan, S. Harabagiu, M. Pasca, R. Mihalcea, R. Girju, R. Goodrum, and V. Rus. The structure and performance of an open-domain question answering system. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, pages 563–570. Association for Computational Linguistics, 2000.

[66] D. Moldovan, S. Harabagiu, M. Pasca, R. Mihalcea, R. Goodrum, R. Girju, and V. Rus. Lasso: A tool for surfing the answer net. *NIST Special Publication*, pages 175–184, 1999.

[67] R.K. Morris. Lexical and message-level sentence context effects on fixation times in reading. *Learning, Memory*, 20(1):92–103, 1994.

[68] K. Nakayama, T. Hara, and S. Nishio. Wikipedia link structure and text mining for semantic relation extraction towards a huge scale global web ontology. In *Proceedings of Sem-Search 2008 CEUR Workshop*, pages 59–73, 2008.

[69] S. Narayanan and S. Harabagiu. Question answering based on semantic structures. In *Proceedings of the 20th international conference on Computational Linguistics (COLING)*, pages 693–701. Association for Computational Linguistics, 2004.

[70] J.H. Neely. Semantic priming and retrieval from lexical memory: Roles of inhibitionless spreading activation and limited-capacity attention. *Journal of Experimental Psychology*, 106(3):226–254, 1977.

[71] N. Oostdijk. Using the TOSCA analysis system to analyse a software manual corpus. In R. Sutcliffe, H. Koch, and A. McElligott, editors, *Industrial Parsing of Software Manuals*, pages 179–206. Amsterdam: Rodopi, 1996.

[72] J. Orkin and D. Roy. The restaurant game: Learning social behavior and language from thousands of players online. *Journal of Game Development*, 3(1):39–60, 2007.

[73] A.B. Owen. Infinitely imbalanced logistic regression. *The Journal of Machine Learning Research*, 8:761–773, 2007.

[74] P. Pantel and D. Lin. Discovering word senses from text. In *Proceedings of the 8th ACM SIGKDD international conference on Knowledge Discovery and Data Mining*, pages 613–619, New York, NY, USA, 2002. ACM.

[75] C. Pechsiri and A. Kawtrakul. Mining Causality from Texts for Question Answering System. *IEICE Transactions on Information and Systems*, 90(10):1523–1533, 2007.

[76] T. Pedersen, S. Patwardhan, and J. Michelizzi. WordNet::Similarity — Measuring the Relatedness of Concepts. In *Proceedings of the National Conference on Artificial Intelligence*, pages 1024–1025. AAAI Press, 2004.

[77] S.P. Ponzetto and M. Strube. Deriving a large scale taxonomy from Wikipedia. In *Proceedings of the National Conference on Artificial Intelligence*, volume 22, pages 1440–1445, Vancouver, BC, 2007.

[78] J. Prager, E. Brown, A. Coden, and D. Radev. Question-answering by predictive annotation. In *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 184–191. ACM New York, NY, USA, 2000.

[79] T. Qin, T.Y Liu, J. Xu, and H. Li. LETOR: A Benchmark Collection for Learning to Rank for Information Retrieval. Technical report, Microsoft Research Asia, 2008.

[80] S. Quarteroni, A. Moschitti, S. Manandhar, and R. Basili. Advanced Structural Representations for Question Classification and Answer Re-ranking. In *Lecture Notes in Computer Science: Advances in Information Retrieval*, volume 4425, pages 234–245, 2007.

[81] R. Quirk, S. Greenbaum, G. Leech, and J. Svartvik. *A comprehensive grammar of the English language*. London, Longman, 1985.

[82] G. Salton and C. Buckley. Term-weighting Approaches in Automatic Text Retrieval. *Information Processing and Management*, 24(5):513–523, 1988.

[83] R.C. Schank and R.P. Abelson. *Scripts, plans, goals and understanding: An inquiry into human knowledge structures*. Lawrence Erlbaum Associates Hillsdale, NJ, 1977.

[84] M.S. Seidenberg. Visual word recognition: An overview. *Speech, Language and Communication*, pages 137–179, 1995.

[85] L. Shen and A.K. Joshi. Ranking and reranking with perceptron. *Machine Learning*, 60(1):73–96, 2005.

[86] R.F. Simmons. Answering English questions by computer: a survey. *Communications of the ACM*, 8:53–70, 1965.

[87] R. Soricut and D. Marcu. Sentence level discourse parsing using syntactic and lexical information. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, pages 149–156. Association for Computational Linguistics, 2003.

[88] M. Surdeanu, M. Ciaramita, and H. Zaragoza. Learning to Rank Answers on Large Online QA Collections. In *Proceedings of the 46th Annual Meeting on Association for Computational Linguistics*, pages 719–727, 2008.

[89] Y. Tang, Y.Q. Zhang, NV Chawla, and S. Krasser. SVMs Modeling for Highly Imbalanced Classification. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 39(1):281–288, 2009.

[90] S. Tellex, B. Katz, J. Lin, A. Fernandes, and G. Marton. Quantitative evaluation of passage retrieval algorithms for question answering. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 41–47, New York, NY, USA, 2003. ACM.

[91] D. Theijssen, H. Van Halteren, S. Verberne, and L. Boves. Features for automatic discourse analysis of paragraphs. In S. Verberne, H. Van Halteren, and P.A. Coppen, editors, *Proceedings of the 18th meeting of Computational Linguistics in the Netherlands (CLIN 2007)*, pages 53–68, 2008.

[92] J. Tiedemann. Improving passage retrieval in question answering using NLP. In *Lecture Notes in Computer Science: Progress in Artificial Intelligence*, volume 3808, pages 634–646. Springer, 2005.

[93] J. Tiedemann. A Comparison of Genetic Algorithms for Optimizing Linguistically Informed IR in Question Answering. In *Lecture Notes in Computer Science: AI*IA 2007: Artificial Intelligence and Human-Oriented Computing*, volume 4733, page 398. Springer, 2007.

[94] A. Trotman. An Artificial Intelligence Approach To Information Retrieval. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, page 603, New York, NY, USA, 2004. ACM.

[95] N. Usunier, M.R. Amini, and P. Gallinari. Boosting weak ranking functions to enhance passage retrieval for Question Answering. In *Proceedings of the Workshop on Information Retrieval for Question Answering (IR4QA) at SIGIR 2004*, pages 1–6, 2004.

[96] B. Van Fraassen. The Pragmatics of Explanation. *The Philosophy of Science*, pages 317–327, 1991.

[97] B.C. Van Fraassen. The Pragmatic Theory of Explanation. In Joseph Pitt, editor, *Theories of Explanation*, pages 135–155. Oxford University Press, 1988.

[98] S. Verberne. Developing an Approach for Why-Question Answering. In *Conference Companion of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2006)*, pages 39–46, Trento, Italy, 2006. Association for Computational Linguistics.

[99] S. Verberne. Evaluating Answer Extraction for Why-QA using RST-annotated Wikipedia Texts. In *Proceedings of the Twelfth ESSLLI Student Session*, pages 255–266, 2007.

[100] S. Verberne. Paragraph Retrieval for Why-Question Answering. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 922–922, New York, NY, USA, 2007. ACM.

[101] S. Verberne, L. Boves, N. Oostdijk, and P.A. Coppen. Data for Question Answering: the Case of Why. In *Proceedings of the 5th edition of the International Conference on Language Resources and Evaluation (LREC 2006)*, Genoa, Italy, 2006.

[102] S. Verberne, L. Boves, N. Oostdijk, and P.A. Coppen. Exploring the Use of Linguistic Analysis for Why-Question Answering. In *Proceedings of the 16th Meeting of Computational Linguistics in the Netherlands (CLIN 2005)*, pages 33–48, Amsterdam, the Netherlands, 2006.

[103] S. Verberne, L. Boves, N. Oostdijk, and P.A. Coppen. Discourse-based Answering of Why-Questions. *Traitement Automatique des Langues (TAL), special issue on "Discours et document: traitements automatiques"*, 47(2):21–41, 2007.

[104] S. Verberne, L. Boves, N. Oostdijk, and P.A. Coppen. Evaluating Discourse-based Answer Extraction for Why-Question Answering. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 735–736, New York, NY, USA, 2007. ACM.

[105] S. Verberne, L. Boves, N. Oostdijk, and P.A. Coppen. Evaluating Paragraph Retrieval for Why-QA. In *Lecture Notes in Computer Science: Advances in Information Retrieval*, volume 4956, pages 669–773. Springer, 2008.

[106] S. Verberne, L. Boves, N. Oostdijk, and P.A. Coppen. Using Syntactic Information for Improving Why-Question Answering. In *Proceedings of the 22nd International Conference on Computational Linguistics (COLING)*, pages 953–960, 2008.

[107] S. Verberne, L. Boves, N. Oostdijk, and P.A. Coppen. What is not in the Bag of Words for Why-QA? *Computational Linguistics*, 32(2), 2010.

[108] S. Verberne, S. Raaijmakers, D. Theijssen, and L. Boves. Learning to Rank Answers to Why-Questions. In *Proceedings of the 9th Dutch-Belgian Information Retrieval Workshop (DIR 2009), Enschede*, pages 34–41, 2009.

[109] S. Verberne, H. Van Halteren, D. Theijssen, S. Raaijmakers, and L. Boves. Learning to Rank QA Data. In *Proceedings of the Workshop on Learning to Rank for Information Retrieval (LR4IR) at SIGIR 2009*, pages 41–48, 2009.

[110] E.M. Voorhees. Overview of the TREC-9 question answering track. In *Proceedings of TREC-9*, volume 7, pages 1–80, 2000.

[111] E.M. Voorhees. The TREC-8 question answering track report. *NIST special publication*, pages 77–82, 2000.

[112] E.M. Voorhees. Overview of the TREC 2001 question answering track. In *In Proceedings of the Tenth Text REtrieval Conference (TREC)*, pages 42–51, 2001.

[113] E.M. Voorhees. Overview of the TREC 2003 question answering track. In *Proceedings of the Twelfth Text REtrieval Conference (TREC 2003)*, volume 142, pages 54–68, 2003.

[114] E.M Voorhees. Overview of TREC 2002. *NIST special publication*, pages 1–16, 2003.

[115] E.M. Voorhees. Overview of the TREC 2004 question answering track. In *Proceedings of TREC 2004*, volume 13, pages 52–62, 2004.

[116] E.M. Voorhees. Overview of TREC 2005. In *Proceedings of the Text REtrieval Conference (TREC). Gaithersburg, Maryland*, pages 1–15, 2005.

[117] E.M. Voorhees and D.M. Tice. Building a question answering test collection. In *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 200–207. ACM New York, NY, USA, 2000.

[118] X.J. Wang, X. Tu, D. Feng, and L. Zhang. Ranking community answers by modeling question-answer relationships via analogical reasoning. In *Proceedings of the 32nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 179–186, New York, NY, USA, 2009. ACM.

[119] W.A. Woods. Lunar rocks in natural English: Explorations in natural language question answering. *Fundamental Studies in Computer Science*, 5:521–569, 1977.

[120] F. Xia, T.Y. Liu, J. Wang, W. Zhang, and H. Li. Listwise approach to learning to rank: theory and algorithm. In *Proceedings of the 25th international conference on Machine learning*, pages 1192–1199, New York, NY, USA, 2008. ACM.

[121] J. Xu and H. Li. Adarank: a boosting algorithm for information retrieval. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 391–398, New York, NY, USA, 2007. ACM.

[122] J.Y. Yeh, J.Y. Lin, H.R. Ke, and W.P. Yang. Learning to rank for information retrieval using genetic programming. In *Proceedings of the Workshop on Learning to Rank for Information Retrieval (LR4IR) at SIGIR 2007*, 2007.

[123] T. Yokoi. The EDR electronic dictionary. *Communications of the ACM*, 38(11):42–44, 1995.

[124] Y. Yue, T. Finley, F. Radlinski, and T. Joachims. A support vector method for optimizing average precision. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 271–278, New York, NY, USA, 2007. ACM.

[125] T. Zesch, C. Muller, and I. Gurevych. Extracting lexical semantic knowledge from wikipedia and wiktionary. In *Proceedings of the Conference on Language Resources and Evaluation (LREC)*, pages 1646–1652, 2008.

[126] C. Zhai. Notes on the Lemur TFIDF model. Technical report, School of Computer Science. Carnegie Mellon University, 2001.

Curriculum Vitae

Suzan Verberne was born in Nijmegen in 1980. She grew up in Twente, where she followed secondary education at Scholengemeenschap De Grundel, Hengelo. She graduated with a curriculum of science subjects and languages ('Gymnasium Bèta'). From 1998 to 2002, Suzan studied Language and Speech Technology ('Taal, Spraak en Informatica') at the University of Nijmegen. Her master thesis addressed the problem of detecting real-word spelling errors using sequences of words.

After her graduation, Suzan worked as a linguistic engineer and project manager for Polderland Language and Speech Technology. A vacancy at the department of Language of Speech for a junior researcher in Natural Language Processing made her decide to go back to the University of Nijmegen in 2005. The following years she worked on her research project "In Search of the Why", resulting in this PhD thesis. In 2008, she worked as a guest researcher at the Intelligent Systems Lab of the University of Amsterdam for several months.

Since January 2009, Suzan has been working as a postdoctoral researcher for the project "Text Mining for Intellectual Property" at the Radboud University Nijmegen. In the first quarter of 2010, she was a guest researcher at the University of Wolverhampton, working on anaphora resolution for patent texts.

List of publications

[1] S. Verberne. Developing an Approach for Why-Question Answering. In *Conference Companion of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2006)*, pages 39–46, Trento, Italy, 2006. Association for Computational Linguistics.

[2] S. Verberne, L. Boves, N. Oostdijk, and P.A. Coppen. Data for Question Answering: the Case of Why. In *Proceedings of the 5th edition of the International Conference on Language Resources and Evaluation (LREC 2006)*, Genoa, Italy, 2006.

[3] S. Verberne, L. Boves, N. Oostdijk, and P.A. Coppen. Exploring the Use of Linguistic Analysis for Why-Question Answering. In *Proceedings of the 16th Meeting of Computational Linguistics in the Netherlands (CLIN 2005)*, pages 33–48, Amsterdam, the Netherlands, 2006.

[4] S. Verberne, L. Boves, N. Oostdijk, and PA Coppen. Discourse-based Answering of Why-Questions. *Traitement Automatique des Langues (TAL), special issue on "Discours et document: traitements automatiques"*, 47(2):21–41, 2007.

[5] S. Verberne. Evaluating Answer Extraction for Why-QA using RST-annotated Wikipedia Texts. In *Proceedings of the Twelfth ESSLLI Student Session*, pages 255–266, 2007.

[6] S. Verberne. Paragraph Retrieval for Why-Question Answering. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 922–922, New York, NY, USA, 2007. ACM.

[7] S. Verberne, L. Boves, N. Oostdijk, and P.A. Coppen. Evaluating Discourse-based Answer Extraction for Why-Question Answering. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 735–736, New York, NY, USA, 2007. ACM.

[8] D. Theijssen, H. Van Halteren, S. Verberne, and L. Boves. Features for automatic discourse analysis of paragraphs. In S. Verberne, H. Van Halteren, and P.A. Coppen, editors, *Proceedings of the 18th meeting of Computational Linguistics in the Netherlands (CLIN 2007)*, pages 53–68, 2008.

[9] S. Verberne, L. Boves, N. Oostdijk, and P.A. Coppen. Evaluating Paragraph Retrieval for Why-QA. In *Lecture Notes in Computer Science: Advances in Information Retrieval*, volume 4956, pages 669–773. Springer, 2008.

[10] M. Khalid and S. Verberne. Passage Retrieval for Question Answering using Sliding Windows. In *Proceedings of the workshop Information Retrieval for Question Answering (IR4QA) at COLING*, 2008.

[11] S. Verberne, L. Boves, N. Oostdijk, and P.A. Coppen. Using Syntactic Information for Improving Why-Question Answering. In *Proceedings of the 22nd International Conference on Computational Linguistics (COLING)*, pages 953–960, 2008.

[12] M. van der Heijden, M. Hinne, W. Kraaij, S. Verberne, and T. van der Weide. Using query logs and click data to create improved document descriptions. In *Proceedings of Workshop on Web Search Click Data (WSCD) at WSDM 2009*, pages 64–67. ACM New York, NY, USA, 2009.

[13] S. Verberne, S. Raaijmakers, D. Theijssen, and L. Boves. Learning to Rank Answers to Why-Questions. In *Proceedings of the 9th Dutch-Belgian Information Retrieval Workshop (DIR 2009), Enschede*, pages 34–41, 2009.

[14] C. Koster, N. Oostdijk, S. Verberne, and E. D'hondt. Challenges in Professional Search with PHASAR. In *Proceedings of the Dutch-Belgium Information Retrieval workshop (DIR 2009), Enschede*, 2009.

[15] M. Hinne, W. Kraaij, S. Verberne, S. Raaijmakers, T. van der Weide, and M. van der Heijden. Annotation of URLs: more than the sum of parts. In *Proceedings of the 32nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 632–633, 2009.

[16] S. Verberne, H. Van Halteren, D. Theijssen, S. Raaijmakers, and L. Boves. Learning to Rank QA Data. In *Proceedings of the Workshop on Learning to Rank for Information Retrieval (LR4IR) at SIGIR 2009*, pages 41–48, 2009.

[17] S. Verberne, M. Hinne, M. van der Heijden, W. Kraaij, E. D'hondt, and T. van der Weide. Annotating URLs with query terms: What factors predict reliable annotations? In *Proceedings of the Understanding the User (UIIR) workshop at SIGIR 2009*, pages 40–43, 2009.

[18] E. D'hondt, S. Verberne, N. Oostdijk, and L. Boves. Re-ranking based on Syntactic Dependencies in Prior-Art Retrieval. In *Proceedings of the Dutch-Belgium Information Retrieval Workshop (DIR)*, pages 23–30, 2010.

[19] S. Verberne and E. D'hondt. Prior art retrieval using the claims section as bag of words. In *Proceedings of the Cross Language Evaluation Forum 2009 (CLEF)*, 2010.

[20] S. Verberne, E. D'hondt, N. Oostdijk, and C. Koster. Quantifying the Challenges in Parsing Patent Claims. In *Proceedings of the 1st International Workshop on Advances in Patent Information Retrieval (AsPIRe) at ECIR*, 2010.

[21] S. Verberne, L. Boves, N. Oostdijk, and P. Coppen. What is not in the Bag of Words for Why-QA? *Computational Linguistics*, 32(2), 2010.