Acquiring Syntactic and Semantic Transformations in Question Answering

Michael Kaisser

Doctor of Philosophy

Institute for Communicating and Collaborative Systems

School of Informatics

University of Edinburgh

2010

Abstract

One and the same fact in natural language can be expressed in many different ways by using different words and/or a different syntax. This phenomenon, commonly called paraphrasing, is the main reason why Natural Language Processing (NLP) is such a challenging task. This becomes especially obvious in Question Answering (QA) where the task is to automatically answer a question posed in natural language, usually in a text collection also consisting of natural language texts. It cannot be assumed that an answer sentence to a question uses the same words as the question and that these words are combined in the same way by using the same syntactic rules.

In this thesis we describe methods that can help to address this problem. Firstly we explore how lexical resources, i.e. FrameNet, PropBank and VerbNet can be used to recognize a wide range of syntactic realizations that an answer sentence to a given question can have. We find that our methods based on these resources work well for web-based Question Answering. However we identify two problems: 1) All three resources as of yet have significant coverage issues. 2) These resources are not suitable to identify answer sentences that show some form of *indirect evidence*. While the first problem hinders performance currently, it is not a theoretical problem that renders the approach unsuitable–it rather shows that more efforts have to be made to produce more complete resources. The second problem is more persistent. Many valid answer sentences–especially in small, journalistic corpora–do not provide *direct evidence* for a question, rather they strongly suggest an answer without logically implying it. Semantically motivated resources like FrameNet, PropBank and VerbNet can not easily be employed to recognize such forms of indirect evidence.

In order to investigate ways of dealing with indirect evidence, we used Amazon's Mechanical Turk to collect over 8,000 manually identified answer sentences from the AQUAINT corpus to the over 1,900 TREC questions from the 2002 to 2006 QA tracks. The pairs of answer sentences and their corresponding questions form the QASP corpus, which we released to the public in April 2008. In this dissertation, we use the QASP corpus to develop an approach to QA based on matching dependency relations between answer candidates and question constituents in the answer sentences. By acquiring knowledge about syntactic and semantic transformations from dependency relations in the QASP corpus, additional answer candidates can be identified that could not be linked to the question with our first approach.

Acknowledgements

First of all, I would like to thank my supervisor, Bonnie Webber, for her patient guidance and continuous support throughout my time in Edinburgh. I know to have been extremely lucky to have a supervisor who is always available to discuss any aspects of my work and who responds to questions and queries so helpfully and promptly. Furthermore, her ability to see the big picture helped a great deal in finishing this thesis.

I also thank the School of Informatics for providing a very productive and friendly research environment, since recently in a great, shiny new building.

I am also grateful to Powerset, Inc., and my supervisor John B. Lowe there, which gave me the opportunity to experience the Silicon Valley culture first hand during a three month internship in their San Francisco offices. (Moreover, Powerset kindly financed the experiment described in Chapter 4.)

Last but not least I would like to thank Microsoft Research for selecting me for one of their Microsoft Research PhD Scholarships and for providing me with the financial support that comes with it.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Michael Kaisser*)

To my late mother, who would have been immensely proud.

Table of Contents

Chapter 1

Introduction

1.1 Factoid Question Answering: A Solved Problem?

This thesis is concerned with Question Answering (QA). Commonly, QA is defined as either

a) a type of Information Retrieval (IR), or

b) a subfield of Natural Language Processing (NLP).

These definitions mark the two poles between which the research area of QA is located. Information Retrieval on one side, which usually deals with large quantities of information, and Natural Language Processing on the other, which mostly is concerned with the interpretation of much smaller pieces of text, e.g. sentences.

The task of a QA system is, given a collection of documents (for example a local collection or alternatively the World Wide Web) to retrieve answers to questions posed in natural language. The traditional approach to QA consists of three steps [Prager, 2006]: 1) Question Analysis, 2) Document Retrieval and 3) Answer Extraction. During question analysis significant keywords from the question are extracted. Additionally, the answer type of the question is determined. A question starting with "When" for example usually asks for a date, while a question starting with "Who" in most cases asks for a person or an organisation. The key words are then sent to an IR module to retrieve a set of documents that contain these keywords, and are therefore considered to be likely to contain the answer. (Many QA systems also work with smaller pieces of text, for example paragraphs or sentences.) Then the answer extraction module uses the input from the IR module and information about the expected answer type to determine the answer. This final answer extraction step is where QA systems differ the most. Usually this is where the bulk of NL Processing comes in, but it is difficult to generalize here. (Note also that some QA systems have a very different architecture.)

QA is often described as one of the, or even the, most challenging tasks in NLP. Unlike other NLP subfields which work within closely defined boundaries and on very specialized tasks, a QA system needs to be able to deal with an arbitrary text and determine which part of it (if any) answers a given question. In order to do this, a "perfect" QA system (most likely in its answer extraction step) would need to incorporate systems from most other subfields of NLP, for example part-of-speech taggers, parsers, named entity recognition systems and also modules for anaphora resolution, word sense disambiguation and even textual entailment. It can be argued that the task a QA system has to achieve is very similar to the human concept of "text understanding".

While NLP has made significant process especially in the most recent decade, researchers in the field are aware that there still is a lot of work to do. Today, part-of-speech taggers, parsers and named entity recognition systems perform impressively well, but still are far from being perfect. This holds even more for anaphora resolution and word sense disambiguation systems where still a lot is to be done, and where off-the-shelf tools with good performance are hard to come by. Most importantly, despite all the progress, NLP is still far away from developing systems that artificially model human text understanding. In order to achieve this, all of NLP's sub areas would have be brought together under one convincingly designed architecture, that also would have to be able to deal with all the pragmatic aspects of human conversations. While this is still fiction, one could argue that, because of the complexity of the QA task, a perfect QA system would need to do just that.

This is the situation the area of QA currently is in. Nevertheless, in recent years some researchers declared the (factoid) QA problem as "solved". The origins of this claim can mainly be found at the QA track in Text REtrieval Conference (TREC), organized by the National Institute of Technology (NIST) since 1999 (see, for example, [Voorhees, 2004], [Voorhees and Dang, 2005], [Dang et al., 2006]).[1] Each year TREC releases a test set of a few hundred questions. Participants run these sets through their fully automatics systems and send their answers to NIST which evaluates them with the

[1] The view of factoid QA as being a solved field can often be heard on conferences. In writing, there are only a few papers addressing this claim, either by repeating it [Zheng et al., 2007] or arguing against it [Prager, 2006].

help of human assessors. TREC then publishes each system's results, measured either in MRR (Mean Reciprocal Rank; the reciprocal rank is the multiplicative inverse of the rank of the first correct answer; Mean Reciprocal Rank is the average of the reciprocal ranks for a set of answers) or, since 2002, accuracy (the fraction of correctly answered questions out of all questions in a question set). Table 1.1 shows the results of the best-performing systems in the TREC QA track since its beginning in 1999. It also lists the median score for the years where this data is available.

year	measure	best	2nd	3rd	median
1999	MRR	.660	.555	.356	.261
2000	MRR	.58	.32	.32	.23
2001	MRR	.68	.57	.48	?
2002	accuracy	.830	.580	.542	?
2003	accuracy	.700	.622	.562	.177
2004	accuracy	.770	.643	.626	.170
2005	accuracy	.713	.666	.326	.152
2006	accuracy	.578	.538	.390	.186
2007	accuracy	.706	.494	.289	.131

Table 1.1: Evaluation results of the best performing QA systems at TREC from 1999 to 2007 (for factoid questions). TREC changed the evaluation metric in 2002. Before systems were evaluated using Mean Reciprocal Rank (the table gives the results for runs with a 50-byte limit on the response length). From 2002 on, top-1 accuracy is used.

There are several observations one can make in Table 1.1:

- The scores for the best system are usually quite high.
- The median scores of all participating systems are very low.
- In most years only a few systems receive good scores, but a large gap exists between these and the rest of the field.

In other fields of NLP it has been observed that whenever a group pulls ahead in performance based on a new idea, this idea quickly becomes adapted by the other groups in the field, so that the performance gap is closed in the following years. This does not seem to be the case in QA. The gap in performance than can be observed in

Table 1.1, especially since 2005, seems not to be based on ideas that the rest of the research community was able to pick up.

Target 141: "Warren Moon" (year: 2006)

141.1: What position did Moon play in professional football?

141.2: Where did Moon play in college?

141.3: In what year was Moon born?

141.4: How many times was Moon a Pro Bowler?

141.5: Who is Warren Moon's agent?

Target 216: "Paul Krugman" (year: 2007)

216.1: For which newspaper does Krugman write?

216.2: At which university does Krugman teach?

216.3: From which university did he receive his doctorate?

216.4: What is Krugman's academic specialty?

216.5: What prize originating in Spain has Krugman won?

Figure 1.1: The first two series in the question sets from 2006 and 2007 (only factoid questions are listed).

TREC 2006	No.	Percent
Time	95	23.6%
Location	69	17.1%
Person	68	16.9%
Number	61	15.1%
Organization	29	7.2%
Work of Art	17	4.2%
Measure	14	3.5%
Money	11	2.7%
Product	6	1.5%
Other	33	8.1%

TREC 2007	No.	Percent
Time	73	20.3%
Person	65	18.1%
Organization	48	13.3%
Number	48	13.3%
Location	41	11.4%
Measure	14	3.9%
Product	13	3.6%
Work of Art	11	3.1%
Money	7	1.9%
Other	38	10.5%

Table 1.2: Distribution of answer types for factoid questions in TREC 2006 and TREC 2007 data. In both tables, the first column shows the answer type, the second column how often this type occurred and the third the percentage this answer type makes up in each year's question set.

Do TREC results show factoid QA to be a solved problem? Certainly some researchers report impressive results. But the majority of the community still produces systems that perform significantly worse. As of yet, there are no standard techniques that one can employ that will produce an accuracy of better than, say, 0.5 on recent TREC test sets. It seems difficult to declare QA as solved as long as there are no reproducible methods that can achieve high accuracy values on TREC or other test sets. This still leaves the question open as how “high” accuracy should be defined. Figure 1.1 lists two example question series from 2006 and 2007. As can be seen factoid TREC questions are still fairly short, precise and mostly ask for well known Named Entities (see Table 1.2, which lists the distribution of answer types in TREC’s test sets from 2006 and 2007). Thus, in order to declare the research area as solved, it would seem reasonable to expect systems to get at least 80% of these questions correct. Yet, such a result was only achieved once in TREC–and that was six years ago in 2002. Still, if we would have reproducible methods to build systems which receive an accuracy of 80% on TREC test sets, this would not mean that no more improvment could be achieved. While it seems utopian to expect an automatic QA system to get every question correct, certainly algorithms are conceivable that achieve an accuracy of .95 on one of the test sets. Remember also that the factoid questions in TREC test sets are still fairly simple. What about test sets consisting of more complicated or longer factoid questions or questions that do not ask for one of the common Named Entities?

1.2 Why is QA Difficult?

Yet, even standard factoid Question Answering provides many yet unsolved problems. A mayor one is that one and the same fact in natural language can be expressed in many different ways by using different words and/or a different syntax. This phenomenon, commonly called paraphrasing, is the main reason why Natural Language Processing is such a challenging task: All NLP applications have to deal with it in one way or another and a lot of research in NLP’s subfields revolve about this issue. This is especially true in Question Answering. It cannot be assumed that an answer sentence to a question uses the same words as the question and that these words are combined in the same way by using the same syntactic rules. If this were the case, standard IR methods based on word overlap would be sufficient, and the perfect QA system would have been built decades ago. There are cases where it is that straightforward:

(1) Where was Franz Kafka born?

(2) Franz Kafka was born in Prague.

Here, a purely keyword-based method is sufficient to realize the overlap and thus relatedness between the question and the candidate sentence. The words “Franz”, “Kafka” and “born” appear in both sentences. Furthermore, each of them contains (beside the stop words “was” and “in”) just one additional word: The interrogative “where” and the answer “Prague”. As already mentioned, this information is additionally used in virtually every QA system: Most questions contain a special word that indicates the semantic class of the entity the question asks for. Here “where” suggests that the answer is some kind of location. Because the answer sentence contains just one word (“Prague”) which denotes a location, this is extracted as the answer.

Nevertheless it is easy to give examples where this traditional IR approach to QA is not sufficient. Two cases have to be distinguished: A QA system might take a sentence that does not contain the answer as one that does or it might not recognize that a given sentence actually contains the answer. For question 1, example sentences illustrating the first point are:

(3) The father of Franz Kafka was born in Munich.

(4) On that day, Max Born met Franz Kafka in Prague.

(5) The Franz Kafka museum in Prague was born some 20 years ago.

(6) Franz Kafka was not born in Munich.

(7) Franz Kafka might have been born in Prague.

(8) He argued that Franz Kafka was born in Munich, but nobody believed him.

(9) I am very unsure about the fact that Franz Kafka was born in Munich.

Examples for the second case would be:

(10) Kafka’s birthplace is Prague.

(11) Franz Kafka was a native of Prague.

(12) Julie Kafka gave birth to her son Franz on July 3, 1883 in Prague.

(13) Prague, Franz Kafka’s birthplace, is a beautiful city.

(14) Historians claiming Franz Kafka not born in Prague proven wrong.

Assuming that a set of suitable candidate sentences has already been selected, a QA system’s task can be reduced to finding out which of these sentences answer the question and which do not.

But what is it that precisely distinguishes the bad examples 3 to 9 from the good examples 10 to 14? Crucially, all the good examples express the same underlaying fact, the fact that is partially expressed in the question (see example 1) and completely in the questions most simple reformulation (example 2). In other words: Sentences 10 to 14 show the same core meaning. [2]

Expressing the same message in a different form or with different words is usually called paraphrasing. Paraphrasing and Question Answering have long been recognized as related problems. In both fields recognizing different surface structures that express the same underlying meaning is a central concern. Consequently, systems to detect paraphrases have been used in QA to improve performance, see for example [Lin and Pantel, 2001] or [Ravichandran and Hovy, 2002]. Work like this also helps to illustrate that the challenge faced in QA is similar, but not the same as in paraphrasing.

Consider, for example, the question:

“Who is Tom Cruise married to?”

For which the answer is:

“Nicole Kidman”

Although question and answer are rather short, a sentence containing this answer, depending on the underlying text collection used, can potentially be very long, as this example from the AQUAINT corpus shows:

> “The drama is said to be about a pair of married psychiatrists (played by the ***married*** **Tom Cruise** and **Nicole Kidman**) and their sexual lives, but only a few Warner executives, Cruise and Kidman, and Pat Kingsley, a top public relations executive, have seen the film.”

Crucially, from a QA perspective there is no need to paraphrase the complete sentence, as only a small part of the sentence answers the question (the relevant parts are highlighted).

Note also that there is some confusion about how exactly the term “paraphrase” is defined. In the *Oxford English Dictionary* [Simpson and Weiner, 1989] we find the following definition:

paraphrase (noun): “an expression in other words, usually fuller and clearer, of the sense of any passage or text; a free rendering or amplification of a passage

[2] Note, however, that some of these sentences contain additional information: Sentence 14, for example, says that there is a dispute between historians about Franz Kafka’s birthplace.

...”

paraphrase (verb): to express the meaning of (a word, phrase, passage, or work) in other words, usually with the object of fuller and clearer exposition...”

In *Webster’s Third New International Dictionary* [Gove, 1961], we find this definition:

paraphrase (noun): “A restatement of a text, passage, or work giving the meaning in another form usually for clearer and fuller exposition ... ”

paraphrase (verb): “To express, interpret, or translate with latitude; to give the meaning of a passage in other language ...”

These definitions are close to how the term is commonly understood in Computational Linguistics/Natural Language Processing, yet they are not spot-on. The above definitions, unsurprisingly, seem to focus on *humans creating* paraphrases, in this thesis however we are much more concerned with *machines detecting* paraphrases. Another potential problem, form our perspective, with some of these definitions (especially the ones found in the Oxford English Dictionary) is that they center on the term “word”. Yet, paraphrasing in a broader sense is not only about changes in the used words but also about syntactical changes, as the following examples illustrate:

(15) Mary sold Paul the guitar.

(16) Mary sold the guitar to Paul.

(17) Paul bought the guitar from Mary.

Sentences 15 and 16, except for “to” use exactly the same words to express the same meaning, yet they are different on a syntactic level. Sentence 17 also expresses the same meaning using a different word for “sold” (“bought”), but crucially the syntax changes as well (“Paul”, for example, moved to the subject position). All three sentences can be considered paraphrases of the same fact. (At least from our QA-centered perspective. That is because all three sentences express the same core fact and all three sentences are are suitable to answer, for example, the question “Who bought Mary’s guitar?”)
Interestingly, when we take a look at WordNet [Miller et al., 1993], we find the following definitions:

paraphrase, paraphrasis (noun): rewording for the purpose of clarification

paraphrase, rephrase, reword (verb): express the same message in different words

That WordNet defines paraphrase in this way–using the term "word" as well–probably is no coincidence. WordNet has been used in the past to help to deal with some problems that arise due to paraphrases in QA, NLP and IR. Query expansion techniques in the IR phase of a QA system are one example here. As mentioned, we cannot expect that a sentence, paragraph or document in the document collection uses the same words as the question. For this reason, some QA systems add words semantically related to those in the question to the IR query. Often these words are found in WordNet. [Voorhees, 1994] conducted a study examining query expansion based on WordNet more than a decade ago, but could not show any significant improvement in performance. More recently researches were able show that WordNet, if used in the right way (for example with appropriate term weighting strategies), can improve performance, see [Fang, 2008]. Nevertheless, this line of research is only suited to address lexical variation. WordNet contains no information about syntactic alternations, as for example the ones given above in sentences 15 to 17. Thus WordNet can only help with the detection of paraphrases, if the term is defined in a narrow senses, as in the above definitions found in WordNet itself.

While this thesis centers on paraphrasing, arguably one of the toughest and most persistent problems in QA (and NLP in general), we should not forget that there are many other additional factors that make QA (and NLP in general) difficult.

Anaphoric Coreference is one example here. If a topic is referred to twice (or more) in a sentence, an anaphor is often used to replace subsequent occurrences. Consider the question "What is the name of the volcano that destroyed Pompeii?" and the answer sentence "Mount Vesuvius fascinated people ever since it destroyed Pompeii in 79 AD." Here, "it" fills the subject position of "destroy", which, in order to capture the correct meaning has to be resolved by "Mount Vesuvius." In such cases the system needs to perform some form of anaphora resolution (see, amongst others, [Vicedo and Ferrandez, 2000b]).

Mood and Negation also provide difficulties for many QA systems, especially for strategies based on key words. Consider the question "Who purchased YouTube?" and the candidate sentences "Google purchased YouTube", "Google did not purchase YouTube" and "Google may purchase YouTube" The first of these sentences constitutes a proper answer sentence to the question, while the second sentence negates the core fact and the third contains a meaning-altering modal verb. Crucially however all three sentences contain the question key words “purchase” and “YouTube” and therefore might easily be interpreted as being valid answer sentences.

In some cases it is even necessary to use inference, reasoning and/or world knowledge to link a question to a valid answer sentence. Consider again the question “‘What is the name of the volcano that destroyed Pompeii?” and the answer sentence “Pompeii was buried by the ashes of Mount Vesuvius in A.D. 79.” Most people would probably, without much thinking, agree that the given sentence answers the question, but a considerable amount of inference and world knowledge is necessary to arrive at this conclusion. To illustrate why this is the case it might help to compare the given answer sentence to a sentence like “I was late at work because in the morning I found my car buried in snow and it took me 10 minutes to remove all of it.” Of course, a town buried by volcano ashes and a car buried in snow describe two completely different scenarios, but this cannot be derived from the syntax of the sentences, instead knowledge about volcano ashes and snow amongst other things is necessary.

As already seen in the last example, some of the problems in QA go well beyond traditional NLP or IR. In recent years the QA community has started to deal with some of these issues. Temporally restricted questions like “Who was president of the United States in 1999?” are one example here. (Such questions have been included in TREC’s question sets since 2006 [Dang et al., 2006].) To answer them it is often necessary to take hints into account that are not provided in the supportive document’s text itself, but are meta information about the document, e.g. its date of publication. Similar problems arise for geographical constraints (“How many people live in Scandinavia?” might require a system to know that Scandinavia is made up of Norway, Sweden and Denmark and add up the number of inhabitants of these countries.) and numerical constraints (e.g. “How many cities worldwide have more than more than one million inhabitants?”).

The just mentioned problem areas all add to the complexity of the QA problem. And of course, there are additional areas beside the ones that have been mentioned here. None of these however are the scope of the work in this thesis. As already noted, we are concerned with the problem that paraphrasing provides for QA. In the remaining sections of this chapter we will look at the contributions this thesis makes in more detail.

1.3 Lexico-Syntactic Paraphrases

In this thesis we are concerned with developing new paraphrasing methods for Question Answering. While much previous work has focused on the lexical aspects of paraphrasing, for example by utilizing WordNet, we focus on the syntactic side of paraphrasing. We argue that many ways in which answer sentences to a question can be formulated can be acquired from (annotated or unannotated) resources that contain a large number of semantically related sentences.

In Chapter 3 of this thesis, lexical resources like FrameNet, PropBank and VerbNet are used to enable a QA system to recognize a wider range of syntactic and semantic variants in answer sentences. Both FrameNet and PropBank contain more than 100,000 annotated sentences that can be employed by a QA system to recognize different ways in which one and the same core fact expressed in different answer sentences can be formulated. FrameNet additionally contains information about the semantic relationships between certain words, e.g. “buy” and “sell” and how the meaning of sentences using the one can be mapped to the meaning of sentences using the other. Two methods based on these resources for web-based Question Answering are described and evaluated by using question sets from TREC’s QA track from 2002 to 2006. Two separate evaluation runs are carried out, the first of which searches for answers on the web using Google while the other searches for answers in the document collection used by TREC, the AQUAINT corpus [Graff, 2002]. We find that our methods work well in a web-based setting (for which they were developed), but that there many candidate sentences that makes our semantically-inspired approach difficult to work on small text collections. This is because a significant subset of candidate sentences, judged to be supportive for an answer by human assessors, do in a strict logical sense not imply the answer. These sentences provide *indirect evidence* for the answer. FrameNet, PropBank and VerbNet, however, are suitable to detect answer sentences which contain *direct evidence* for the answer.

The observation that many of the valid answer sentences in the AQUAINT corpus (from documents judged as relevant by TREC assessors), do not actually answer the question in a strictly logical sense leads to the work described in Chapter 4. In order to be able to better characterize the relations between TREC questions and their answer sentences, a relatively new web service, Amazon’s Mechanical Turk, is used to help locate valid answer sentences in all documents identified as supportive for a question by TREC. The collected data, consisting of more than 8,000 answer sentences to more than 1,900 questions, forms a corpus of **Q**uestion **A**nswer **S**entence **P**airs (QASPs) which we released to the public. [3] This corpus can be beneficial for research into QA on many different levels: 1) Just by studying it researchers are able to better understand what the actual challenges for a factoid QA system are 2) It can be used to automatically characterize different kind of links between question and answer sentences, e.g. word overlap 3) It can be used as training data for various kinds of QA algorithms. For this thesis, the QASP corpus is important because it contains many answer sentences expressing *indirect evidence* for the question.

Chapter 5 describes work addressing point (3). An approach to QA that acquires knowledge from the QASP corpus about how answer sentences to a question can be formulated is presented. It is based on matching dependency relations between answer candidates and question constituents in candidate sentences. Because this approach acquires its knowledge from answer sentences judged as supportive by human assessors, the nature of the training data fits the goal (to return answers in answer sentences that are (or will be) judged as supportive by human assessors) very well. This was not necessarily the case with our approach based on lexical resources, which acquired knowledge from sentences exemplyfing strict semantic equivalence. We evaluate performance on the same TREC test sets that were used for the our first approach. We expect this approach to perform considerably better on a local corpus that the first approach, which it does although the size of the training data is much smaller. We are furthermore able to show that the algorithm's performance steadily increases when comparing runs with small amounts of training data to runs using larger amounts of training data. This strongly suggests that performance should further rise, if more training data would be added.

To sum up what has already been mentioned: Both kinds of corpora used in this thesis, on the one hand lexical resources like FrameNet, PropBank (and to a certain extend VerbNet) and on the other the QASP corpus provide a new perspective on paraphrasing for QA, especially its syntactic side:

1. FrameNet and the like provide a large number of sentences which are annotated with semantic roles. These resources usually list more than one sentence per predicate and thus can be used to automatically exploit the various ways of how one core fact can be expressed.

[3] http://homepages.inf.ed.ac.uk/s0570760/data/

2. The QASP corpus explicitly lists questions and their answer sentences (and the answers in these sentences). It usually lists more than one answer sentence per question, and more than one question per syntactic question class, thus knowledge about how answer sentences for a certain question class can be formulated can be extracted. (As of yet, no other corpus with similar properties is publicly available.)

1.4 Contributions of this Thesis

As mentioned, in this thesis new ways of dealing with paraphrases in QA are examined. Other than previous work which addresses lexical variation between questions and answer sentences (or passages), we shift the focus to paraphrases involving grammatical variations. We argue that such variations can be acquired from (annotated or unannotated) corpora.

More precisely, this thesis contributes to the field of QA in the following ways:

- It demonstrates that lexical resources like FrameNet, PropBank and VerbNet can beneficial to a Question Answering system by enabling a QA system to detect many forms of paraphrases. (Chapter 3)
- It identifies limitations of these resources, most notably coverage (this might be addressed in the future by creating more complete resources) and a tendency of the resources to only accept answer sentences exemplyfing *direct evidence* for the question. (Chapter 3)
- It argues that methods in QA based on *direct evidence* are suitable for very large corpora (especially the web), but that for small corpora, methods that accept *indirect evidence* are necessary. (Chapter 3)
- It places a novel corpus of **Q**uestion **A**nswer **S**entence **P**airs (QASPs) in the public domain. This corpus contains more than 8,000 answer sentences for more than 1,900 questions and can be used as training data for various QA algorithms. (Chapter 4)
- It provides a numerical analysis of the QASP corpus, more specifically of some selected properties of the questions and answer sentences it contains and the

relations between them. The results provide further evidence for the need of strong paraphrasing capabilities in Question Answering. (Chapter 4)

- It describes an algorithm for factoid Question Answering that is based on extracting dependency relations from the data in the QASP corpus. This algorithm is suitable to identify a much wider range of potential syntactic and semantic answer sentence structures than previous algorithms, some of which are, for example, based on matching syntactic structures of questions to those of candidate sentences. It furthermore is capable of detecting forms of indirect evidence. (Chapter 5)

1.5 Outline of this Thesis

The chapters of this thesis are organized as follows:

Chapter 1 "Introduction" (this chapter) provides a brief introduction to the field of Question Answering and motivates the research carried out in this thesis.

Chapter 2 "The QuALiM Question Answering System" gives some necessary background about the QA system with provides the context for many of the experiments carried out in this thesis.

Chapter 3 "Question Answering based on Semantic Roles" describes an approach to QA based on the lexical resources FrameNet, PropBank and VerbNet. FrameNet alone lists more than 135,000 annotated example sentences that can be used to recognize different potential surface structures of answer sentences.

Chapter 4 "A Corpus of Question Answer Sentence Pairs (QASPs)" describes the creation of a corpus of Question Answer Sentence Pairs. It also gives an analysis of a few selected features of this corpus.

Chapter 5 "Learning Syntactic and Semantic Reformulations from the QASP Corpus" details how the QASP corpus has been used as training data for a QA algorithm in order to acquire syntactic and semantic transformation rules.

Chapter 6 "Conclusions" sums up the thesis and recapitulate what has been achieved. Open questions and directions for possible future work are discussed.

Discussion of related work can be found in the appropriate chapters, most notably the two chapters in this thesis that introduce new algorithms, Chapter 3 “Question Answering based on Semantic Roles” and Chapter 5 “Learning Syntactic and Semantic Reformulations from the QASP Corpus.”

Chapter 2

The QuALiM Question Answering System

2.1 Introduction

This chapter briefly describes the QuALiM [1] Question Answering system which initially was developed to participate in TREC 2004 and has since then been continuously advanced. This system provides the context in which many of the experiments in this thesis are carried out. Sometimes QuALiM is used as a baseline against which other algorithms are compared, sometimes new algorithms borrow certain modules from QuALiM. Thus, in order to fully understand the experiments in the following chapters some explanations about QuALiM are necessary. The research and engineering efforts that went into QuALiM however are not part of this thesis.

QuALiM is a pattern-based QA system that searches the web for answers. Each of its patterns contains a syntactic description that matches a subclass of questions, a set of syntactic descriptions of potential answer sentences, and semantic information concerning the appropriate answer type for the question class. When asked a question QuALiM will search all of the patterns' question descriptions and retain those that matches the question. The matching pattern's information about potential answer sentence formulations is used to create rather specific, quoted search queries that are send to a web search engine (either Google or Yahoo). From the web sentences are retrieved that match the search queries on a string level. The retrieved sentences are then parsed and tagged and it is checked whether they also match the syntactic structure

[1] QuALiM stands for **Qu**estion **A**nswering based on **Li**nguistic **M**ethods

proposed in the first place. From those candidate sentence that show a syntactic match the exact answer is extracted, which is then checked on its semantic type. Additionally, QuALiM implements a fallback mechanism, which does not propose reformulations, but instead sends queries created from key words and key phrases in the question to the web search engine. From the returned snippets n-grams are mined, which are also checked on their semantic type.

The basic approach QuALiM uses is similar to [Dumais et al., 2002] in that a web search engine is fed with partial answer sentence gained from reformulating the question. However, while the reformulation procedure in [Dumais et al., 2002] is string based, QuALiM's reformulations are based on syntax. As a result a wider range of more exact reformulations can be created. Furthermore, QuALiM's knowledge about the phrasal type of the answer and about its position in the answer sentence enables the extraction of exact answers. (The approach in [Dumais et al., 2002], just because it lacks syntactic knowledge, cannot determine answer boundaries and therefore returns only passages.)

In the following the mentioned processing steps and a few other concepts relevant for later work will be described in more detail.

2.2 Algorithms an System Modules

2.2.1 Strict Pattern Matching

QuALiM's strict pattern matching algorithm relies on the already mentioned patterns which are used to define linguistic constraints on questions, potential answer sentences to these questions and the answers themselves. A pattern consists of three parts:

- *Sequences* are used to classify questions according to their syntactic structure.
- *Targets* describe the syntactic structure of potential answer candidates.
- *AnswerTypes* express semantic constraints on the answers.

Figure 2.1 gives an example of such a pattern.

Each question that the system is asked is checked on whether it matches one of the sequences in the pattern files. In 2004, 157 such patterns existed, currently there are 244. The sequence which can be seen in figure 2.1 matches any question that starts with the word "When", followed by the word "did", followed by an NP, followed by

```
<pattern name="When+did+NP+Verb+NPorPP" level="5">
  <sequence>
    <word id="1">When</word>
    <word id="2">did</word>
    <parse id="3">NP</parse>
    <morph id="4">V INF</morph>
    <parse id="5">NP|PP</parse>
    <final>?</final>
  </sequence>

  <target name="target1">
    <ref>3</ref>
    <ref morph="V PAST">4</ref>
    <ref>5</ref>
    <word>in</word>
    <answer>NP</answer>
  </target>
  <target name="target2">
    <word>in</word>
    <answer>NP</answer>
    <punctuation optional="true">,<punctuation>
    <ref>3</ref>
    <ref morph="V PAST">4</ref>
    <ref>5</ref>
  </target>

  ... more targets ...

   <answerType phrases="NP|PP">
     <built-in weight="6">dateComplete</built-in>
     <namedEntity weight="12">date</namedEntity>
     <built-in weight="9">year|in_year</built-in>
     <other ignore="true"/>
   </answerType>
 </pattern>
```

Figure 2.1: Example pattern as used in the current version of the QuALiM system.

a verb in its infinitive form, followed by an NP or a PP, followed by a question mark which has in addition to be the last element in the question.

For the TREC 2004 question set, for example, this sequence matches five questions:

- When did Floyd Patterson win the title?
- When did Amtrak begin operations?
- When did Jack Welch become chairman of General Electric?
- When did Jack Welch retire from GE?
- When did the Khmer Rouge come into power?

If a question matches a sequence, the targets are used to predict (rather flat) linguistic structures of potential answer sentences. Two targets are shown in figure 2.1. For the question “When did Amtrak begin operations?”, they suggest the following answer sentences (or answer sentence parts):

1. Amtrak began operations in ANSWER[NP]
2. In ANSWER[NP] (,) Amtrak began operations

The numbers in the *ref* elements are variables that point back to the sequence element with the corresponding *id* attribute. Beside the *target* elements which can be seen in the example (*ref*, *word*, *punctuation* and *answer*), three others exist: *pos* to match single words with a particular part of speech tag, *parse* to match phrasal constituents in a parse tree (e.g. “NP” or “PP”) and *unknown* to a specified number of words without placing any constraints on them. These targets are used to propose surface structures of the potential answer sentences, from which search queries are created which are sent to the web search engine. For our example these queries are:

```
"Amtrak began operations in"
"In" "Amtrak began operations"
```

From the first 40 snippets returned for each search query those sentences that contain all words from the query are extracted. At the time of writing, for the first query listed above, the first five sentences QuALiM finds are:

- “Since Amtrak began operations in 1971, federal outlays for intercity rail passenger service have been about $18 billion.”
- “Amtrak began operations in 1971.”
- “Amtrak of the obligation to operate the basic system of routes that was largely inherited from the private railroads when Amtrak began operations in 1971.”
- “Amtrak began operations in 1971, as authorized by the Rail Passenger Service Act of 1970.”
- “A comprehensive history of intercity passenger service in Indiana, from the mid-19th century through May 1, 1971, when Amtrak began operations in the state.”

These candidate sentences are parsed with the the LINK parser [Grinberg et al., 1995], [Sleator and Temperley, 1993], tagged with QTag [Tufis and Mason, 1998] and checked on weather the linguistic structure described in the target really matches the sentences. If the system finds this structure, it also knows which constituent of the sentence must be the answer. In the first four examples given above it is “1971”, in the last “the state” (which is sorted out in a later processing step, when the system recognizes that "the state" is not an appropriate answer for this type of question).

The system will place all answers it has found in a *Weighted Bag of Strings*, a custom-built data structure that holds a set of strings, each of which has a value attached. This value is set to one each time a new string is added. Each time a string is attempted to be added which already is present in the bag, it in fact will not be added, but instead the value of the similar string already in the bag will be increased by one.

For the above example, "1971" is added four times, and "the state" once, thus the resulting Weighted Bag of Strings looks like this:

```
4: "1971"
1: "the state"
```

2.2.2 Fuzzy Pattern Matching

During development, it became obvious that the constraints placed on answer sentences by the strict pattern matching algorithm are sometimes too strict. Sometimes a retrieved sentence contains the correct answer, but it cannot be extracted because it is located at a different position than described by the target. A Fuzzy Pattern Matching algorithm was designed to retrieve such results. It re-uses the candidate sentences mined from the web by the strict pattern matching algorithm. For the second target shown in figure 2.1 a possible answer sentence received from Google might for example be:

"In 1971, the railroad company Amtrak began operations."

This sentence does not match the target because no single NP is placed between the word "In" and the NP "Amtrak". Because, in this example, the target used to create the search query and retrieve the candidate sentence specifies the answer as an NP, the fuzzy pattern matching algorithm will now extract all NPs in the candidate sentences retrieved by the strict pattern matching algorithm, regardless of their position. For the answer sentence given the parser returns five NPs: "1971", "the railroad company", "the railroad company Amtrak", "Amtrak" and "operations". The last three NPs are disregarded because they contain words that are part of the query. The remaining two are used to create another Weighted Bag of Strings, in our example:

```
1: "1971", "the railroad company"
```

This bag constitutes this algorithm's results.

2.2.3 The Web Fallback Algorithm

QuALiM implements a further algorithm to find answers. It constructs three search queries that combine NPs in the question and non-stop words. The first query consists of all non-stop words, the second query of all NPs and the third of all NPs plus all non-stop words not mentioned in the NPs. The question “When was Jim Inhofe first elected to the senate?”, for example, becomes:

1. `Jim Inhofe senate first elected`
2. `"Jim Inhofe" "the senate"`
3. `"Jim Inhofe" "the senate" first elected`

These queries are sent to the web search engine and from the snippets returned, n-grams are mined. These are placed in a Weighted Bag of Strings, where the values show how often an n-gram has been found in the snippets. Each value is then multiplied with a modifier based on the n-gram’s length in words. Currently we use the following very simple formula, to determine the multiplicator m which depends on the number of words n in the n-gram:

$$m = \begin{cases} 1 & \text{if } n = 1 \\ (n/2) + 1 & \text{if } n > 1 \end{cases} \tag{2.1}$$

2.2.4 Combining Results from Different Algorithms

Each of QuALiM’s algorithms returns its results in an already mentioned data structure named a *Weighted Bag of Strings*. Essentially it contains a set of string, where each string is only contained once, but with a weight attached. So far the Weighted Bag of Strings of two of the three algorithms were given:

Strict Pattern Matching:

```
4: "1971"
1: "the state"
```

Fuzzy Pattern Matching:

```
1: "1971", "the railroad company"
```

Let us assume that, for the same question, the third algorithm, the web fallback mechanism, would return the following bag:

```
6: "United States"
4: "National Railroad Passenger Corporation"
2: "1971", "Richard Nixon"
1: "1970"
```

Crucially, each algorithm has a weight assigned (usually set manually) meant to expresses the reliability of the algorithm: The higher the weight, the more reliable the algorithm. (Reliability is here defined as the number of correct answers divided by the number of questions for which the system identified at least one answer candidate.) For the three algorithms in questions the weights used are:

Strict Pattern Matching	20
Fuzzy Pattern Matching	5
Web Fallback	1

When the three earlier mentioned algorithms' results are combined into one overall results, each entries' value in each bag is first multiplied by its algorithm's weight. Then all three bags are added up, so that the final bag contains all entires from all three bags and the values for each of the entires is the sum from all those entires individual sums. For our example the result is:

```
87: "1971"
20: "the state"
6:  "United States"
5:  "the railroad company"
4:  "National Railroad Passenger Corporation"
2:  "Richard Nixon"
1: "1970"
```

(The equation that leads to the value 87 for the answer candidate "1971" is $20*4+5*1+1*2=87$.) This method of combining the different results from individual algorithms showed to be quite effective. If QuALiM finds many sentences that match the targets exactly, the results from the fuzzy pattern matching algorithm are of almost no importance. If there are no or only a few exact matching sentences found, the fuzzy results will become more important. The fallback algorithm is just that, a fallback strategy that becomes important only if the other two algorithms return no results or as a tie-breaker if their results are not consistent.

2.2.5 Semantic Answer Type Checking

The last step in QuALiM's processing pipeline is to check the answer candidates on their correct semantic type. To do this the *answerType* element in the pattern's XML structure is used. Each of the possible child elements in the XML structure is associated with a different information source. Beside a few experimental features three main tools are used:

1. Named Entity Recognition–we use ANNIE, the Named Entity Recognizer that comes with GATE [Cunningham et al., 2002]

2. WordNet [Miller et al., 1993]

3. built-in named entity recognition features, which are used to recognize standard date specifications, year specifications, numbers, number/unit compounds etc.

As can be seen in Figure 2.1, each answer type element has a weight attached. It is used to multiply those items in the Weighted Bag of Strings which match the condition expressed by the element.

For the "When did Amtrak begin operations?" example four answer type elements are listed in Figure 2.1 which are repeated here:

```
<built-in weight="6">dateComplete</built-in>
<namedEntity weight="12">date</namedEntity>
<built-in weight="9">year|in_year</built-in>
<other ignore="true"/>
```

The first element matches if an element in a bag is a complete date in a standard format, e.g. "May 1, 1971". The second element matches if ANNIE recognizes the string as a date. (Note that whenever this is the case the first element matches as well.) The third element matches any year specifications, e.g. "1971" or "100 BC", it also matches these if the year is preceded by the preposition "in", e.g. "in 1971". The last element says that any strings that do not match any of the previous conditions are to be discarded. This last element is usually included for answer types where it is easy to tell whether a candidate belongs to this class or not, e.g. dates (all possible dates in standard form can be created by a fairly simply algorithm). This element is usually missing (or included as `<other ignore="false"/>`) for answer types that come from a more open class, e.g. person names or location names, where it often is hard to determine weather a string belongs to one of these classes or not. [2]

For our example, because none of the answer candidates is a complete date, only the third answer type element matches the candidates “1970” and “1971”, thus their values are multiplied with 9.[3] All other candidates are excluded. The final result, of all three algorithms combined and after type checking, is:

```
783: "1971"
9:   "1970"
```

From the final Weighted Bag of Strings, the entry with the highest value is selected as the final answer, for our example this is “1971”, which is the correct answer.

2.3 TREC Evaluations

The QuALiM system participated in TREC’s QA track in the years 2004, 2005 and 2006. Table 2.1, 2.2 and 2.3 show the official results as determined by TREC, which QuALiM received in the corresponding year. The runs usually differed only in parameter settings.

TREC 04	run 1	run 2	run 3	median	best	rank
Factoid	**0.343**	0.339	**0.343**	0.170	0.770	4
List	0.096	0.111	**0.125**	0.094	0.622	9
Other	0.145	0.181	**0.211**	0.184	0.460	10
Combined	0.232	0.242	**0.256**	?	0.601	6

Table 2.1: Official TREC 2004 results for the QuALiM system (overall 28 participants).

Rows two to four in these tables show the results achieved for factoid, list and other questions, repetitively. Row five shows the combined results when the individual

[2] Consider the answer candidate “Vatsyayana” picked up by the fallback algorithm for all three questions “Who wrote the Kama Sutra?”, “Where was the Kama Sutra written?” and “When was the Kama Sutra written?” Obviously the candidate is not a date, so it definitely should be excluded as an answer for the third question. But is it a name for a person or a location? ANNIE does not tag it as either of these two possibilities, although in reality “Vatsyayana” is a person name and the correct answer to the first question. Knowing that Named Entity Recognition Systems often fail in such cases, we do not want to exclude a candidate just because it is not recognized as belonging to this class by ANNIE.

[3] If we would have a string that matches the first two answer type elements, with the weights 6 and 12, these would be added up and the sum would then be multiplied with that strings value.

TREC 05	run 1	run 2	run3	median	best	rank
Factoid	0.207	**0.235**	-	0.152	0.713	9
List	0.029	**0.032**	-	0.053	0.468	?
Other	**0.147**	0.123	-	0.156	0.248	?
Combined	0.150	**0.158**	-	0.123	0.534	10

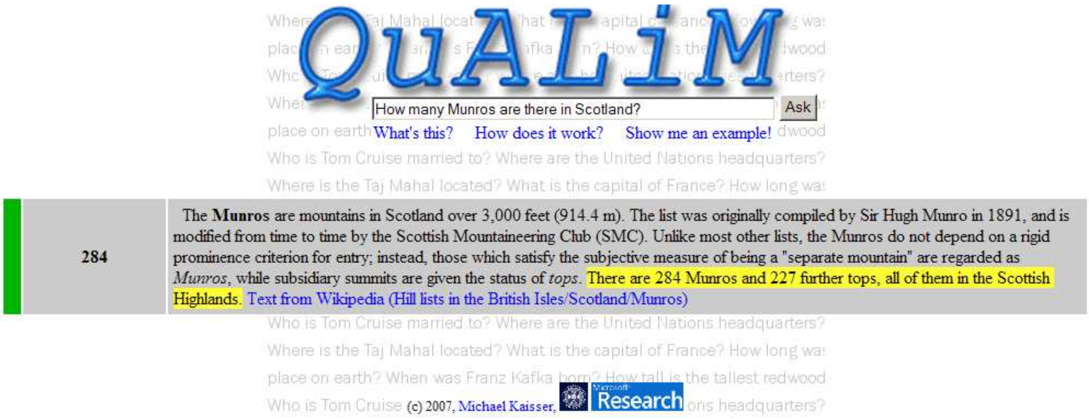
Table 2.2: Official TREC 2005 results for the QuALiM system (overall 30 participants). In 2005, we only submitted two out of three possible runs.

TREC 06	run 1	run 2	run 3	median	best	rank
Factoid	**0.323**	0.303	0.293	0.186	0.578	4
List	0.051	0.053	**0.054**	0.087	0.433	?
Other	**0.250**	0.229	0.203	0.125	0.250	1
Combined	**0.207**	0.192	0.181	0.134	0.394	4

Table 2.3: Official TREC 2006 results for the QuALiM system (overall 27 participants).

results for all three question types are combined. Columns two to four show the results the individual runs received. Columns five and six show median and best results of **all** participants in TREC's QA track in the corresponding year. Column seven lists how our best run ranked compared to the other participant's best run. (TREC usually only publishes rankings for the top 10 participants, so if we did not make it into the top 10 this is indicated with a question mark.)

More details about these figures and how they are computed can be obtained in the relevant year's TREC QA track paper: [Voorhees, 2004, Voorhees and Dang, 2005, Dang et al., 2006]. More details about the methods used in our submissions can be obtained in our papers published in TREC's proceedings: [Kaisser and Becker, 2004, Kaisser, 2005, Kaisser et al., 2006]. In 2004, QuALiM basically consisted of the three algorithms described in this chapter. For TREC 2005, an early version of the algorithm based on FrameNet, described in Chapter 3 of this thesis was added. In 2006, all algorithms described in Chapter 3 in all variations were used in parallel to the three algorithms described in this chapter. We did not participate in TREC after 2007. The reason for this is the high workload that comes with it. Since late 2007/early 2008 however, an online demo of QuALiM (which supplementing answers with paragraphs drawn from Wikipedia) can be found here: http://demos.inf.ed.ac.uk:8080/qualim/. A screenshot can be seen in Figure 2.2, the demo is described in more detail in [Kaisser, 2008].



284

The **Munros** are mountains in Scotland over 3,000 feet (914.4 m). The list was originally compiled by Sir Hugh Munro in 1891, and is modified from time to time by the Scottish Mountaineering Club (SMC). Unlike most other lists, the Munros do not depend on a rigid prominence criterion for entry; instead, those which satisfy the subjective measure of being a "separate mountain" are regarded as *Munros*, while subsidiary summits are given the status of *tops*. There are 284 Munros and 227 further tops, all of them in the Scottish Highlands. Text from Wikipedia (Hill lists in the British Isles/Scotland/Munros)

(c) 2007, Michael Kaisser,

Figure 2.2: Screenshot of QuALiM's response to the question "How many Munros are there in Scotland?" The green bar to the left indicates that the system is confident to have found the right answer, which is shown in bold: "284". Furthermore, one Wikipedia paragraph which contains additional information of potential interest to the user is displayed. In this paragraph the sentence containing the answer is highlighted. This display of context also allows the user to validate the answer.

Chapter 3

Question Answering Based on Semantic Roles

3.1 Introduction

The work described in this chapter is concerned with an approach to web Question Answering based on lexical resources, i.e. FrameNet [Baker et al., 1998], PropBank [Palmer et al., 2005] and VerbNet [Schuler, 2005]. All three resources convey information about lexical predicates, their arguments and the relationship that exists between them–the latter commonly called semantic roles. The process of assigning semantic roles to text, shallow semantic parsing, is often described as "the process of assigning a WHO did WHAT to WHOM, WHEN, WHERE, WHY, HOW etc. structure to plain text" [Pradhan et al., 2005a]. It thus provides a link between the surface appearance of a string and its underlying semantic representation. Looking at it from a slightly different angle, a shallow semantic parser can be seen as a tool to recognize paraphrases in natural language texts. It seems obvious that this potentially can be largely beneficial for a factoid Question Answering system.

In this thesis, we explore the use of FrameNet, PropBank and VerbNet for web-based, factoid Question Answering, in the first instance because of their ability to assist with the detection of paraphrases. For the main part, these resources focus on the syntactic aspects of paraphrasing.

Before describing these resources in more detail in the next section, let us look at a few of their underlying features and how they relate to and can be employed in Question Answering. In the following we will argue that

1. the notion of semantic roles provides a good framework to capture the semantics

of questions (and their answer sentences).

2. the notion of valence, which is central to these resources is a necessity in every syntax semantics interface and that knowledge about valences enables a QA system to recognize a wider range of answer sentences.

3. the frames in FrameNet enable the recognition of even wider paraphrases.

Question Answering and Semantic Roles

Wh-questions differ from the declarative sentences found in ordinary text, in being open propositions – i.e. there is at least one (and usually only one) part of the underlying proposition that is unknown to the questioner, who wants to know true ways of filling it. Adopting the notion of Semantic Roles provides an intuitive way of capturing the relations between questions and their answer sentences on a semantic level. Consider the following examples, each of which is given with a semantic representation inspired by FrameNet frames:

"Google purchased YouTube for $1.65 billion in 2006."
purchase(Buyer=*Google*, Goods=*YouTube*, Time=*2006*, Price=*$1.65_billion*)

"Who purchased YouTube?"
purchase(Buyer=*X*, Goods=*YouTube*)

"When did Google purchase YouTube?"
purchase(Buyer=*Google*, Goods=*YouTube*, Time=*X*)

"For how much did Google purchase YouTube?"
purchase(Buyer=*Google*, Goods=*YouTube*, Price=*X*)

All three resources contain information that makes it possible to transform questions into semantic representations similar to the ones above, and they contain data about how potential answer sentences to such questions might be formulated.

Valence as a necessary part of a Syntax-Semantics Interface

While it might seem like syntax alone could be sufficient to identify one constituent in an answer sentences as the answer to a given question[1], upon closer inspection one

[1] Many QA systems work on this assumption, e.g. [Attardi et al., 2001] or [Katz and Lin, 2003]. Section 5.2.1 of this thesis discusses these papers in more detail.

finds that this often is not the case. Consider the following example:

"Who stars in the Poseidon Adventure?"
star(Performer=*X*, Performance=*Poseidon_Adventure*)

"Gene Hackman stars in the Poseidon Adventure."
star(Performer=*Gene_Hackman*, Performance=*Poseidon_Adventure*)

"The Poseidon Adventure stars Gene Hackman."
star(Performer=*Gene_Hackman*, Performance=*Poseidon_Adventure*)

The above question shows the wh-word "Who" at subject position which could be seen to indicate that in an answer sentence having the string "the Poseidon Adventure" as the object of the verb *to star*, the answer should be found at subject position. Indeed, this is the case in the first answer sentence given above. However, in the second answer sentence, the answer is found at object position. This example illustrates that a particular semantic role is not always realized in the same syntactic function. The phenomenon observed here is commonly called *valence* and all of the three mentioned resources provide data about verb valence that can help with the assignment of semantic roles to syntactic functions, which is necessary to interpret many answer sentences, for example the second of the above. Other examples for alternations that the resources recognize as expressing the same fact are:

"Peter gives a book to Mary."
"Peter gives Mary a book."

"The door opens with this key."
"This key opens the door."

"The firm merges with the company."
"The firm and the company merges."
"The government merges the firm and the company."

A Question Answering system that draws on such data can be expected to identify more answer sentences than a system working on syntax alone, and thus it can be expected to perform better.

Extended Paraphrasing Possibilities with FrameNet

FrameNet, unlike PropBank and VerbNet, contains not only verbs but also entries of other parts of speech and it organizes its lexical entries in frames between which different kinds of relations exist (described in more detail in the next section). This enables a much wider form of paraphrasing than it is the case with the other two resources. FrameNet makes it for example possible to locate the answer in following sentences to the question “When was Alaska purchased?”:

“The United States purchased Alaska in 1867.”

“Alaska was bought from Russia in 1867.”

“In 1867, Russia sold Alaska to the United States.”

“The acquisition of Alaska by the United States in 1867 is known as Seward’s Folly.”

The fact that the capability to recognize such paraphrases can benefit a QA system (or another NLP application) is also expressed in Section 6.3.3 of the FrameNet book [Ruppenhofer et al., 2006] (In the excerpt, FE stands for *frame element*, essentially FrameNet terminology for *semantic role*):

> *In many ways, paraphrasing is at the core of what we intend FrameNet to facilitate. A properly powerful ability to paraphrase enables many of the other goals of semantic NLP, including Question Answering, Summarization, and Translation. Question Answering can be thought of as looking in a corpus to find a paraphrase, but with real information filled in for the questioned FE. Summarization is equivalent to paraphrase of a text, but with the strategic omission of information from FEs and targets. Translation is paraphrasing with the limitation that all the resulting paraphrase must be in the target language.*

Of the mentioned research fields, Question Answering is arguably the most natural candidate to show that FrameNet and the like can be beneficial for Natural Language Processing. Unlike Summarization and Machine Translation, QA, for the most part, works on sentence level and the resources annotate on sentence level. (Although PropBank annotates continuous text, it does not annotate inter-sentence relations, e.g. discourse markers.) Furthermore, Question Answering, at least as far as we are in this thesis concerned, is mono-lingual.[2] Translation, naturally, is not, thus parallel resources for different languages are needed. As of 2009 such data is only sparsely available.[3]

[2] Of course, there is a lot of research concerning multi-lingual QA. For example, the Cross-Language Evaluation Forum (CLEF), broadly speaking the European equivalent of TREC, runs a Multiple Language Question Answering track since 2003. [Magnini et al., 2004, Magnini et al., 2006]

[3] 2008 saw two non-English versions of FrameNet release their first data to the public, German

An additional point worth mentioning is that other lexical resources, most notably WordNet [Miller et al., 1993], have been and are being employed by many researchers in NLP and that today there is consensus that they are highly useful. In QA, for example, WordNet has been used for virtually all components of a QA system, most notably question analysis [Vicedo and Ferrandez, 2000a], document/passage retrieval [Hovy et al., 2000], answer extraction [Cardie et al., 2000] or all of these [Pasca and Harabagiu, 2001]. FrameNet, PropBank and VerbNet on the other hand are rather new. Yet, *in theory* they offer possibilities that go far beyond those of WordNet. As of now however, it remains to be seen how useful they are *in practice*.

In the remainder of this chapter we will first describe the three resources–FrameNet, PropBank and VerbNet–on which all of this chapter's experiments are based in more detail. This is done in Section 3.2. Section 3.3 is concerned with related work. The following two sections describe two methods that use these resources to annotate both questions and sentences containing answer candidates with semantic roles. The first algorithm presented in Section 3.4 uses the three lexical resources to generate potential answer-containing templates. While the templates contain holes–in particular, for the answer–the parts that are known can be used to create exact quoted search queries. Sentences can then be extracted from the output of the search engine and annotated with respect to the resource being used. From this, an answer candidate (if present) can be extracted. The second algorithm, described in Section 3.5 analyzes the dependency structure of the annotated example sentences in FrameNet and PropBank. It poses rather abstract queries to the web, but can in its candidate sentence analysis stage deal with a wider range of syntactic possibilities. Section 3.6 presents an evaluation of both algorithms' performance separately and when they are combined. Finally Section 3.7 contains a discussion of our findings.

3.2 Lexical Resources–An Overview

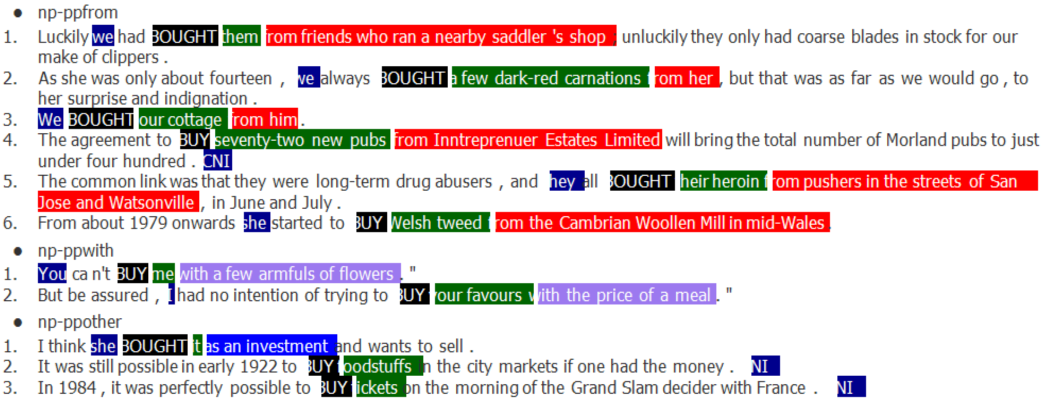
This section gives an overview of the three lexical resources–FrameNet, PropBank and VerbNet–which in the following sections of this chapter are used in a Question Answering system.

SALSA [Burchardt et al., 2006] and Spanish SFN [Subirats and Sato, 2004].

3.2.1 FrameNet

FrameNet [Baker et al., 1998, Ruppenhofer et al., 2006] is a lexical resource based on Frame semantics [Fillmore, 1976], a theory relating linguistic semantics to encyclopedic knowledge, which was developed by Charles J. Fillmore, and is a further development of his case grammar [Fillmore, 1968]. In frame semantics, a word evokes a frame of semantic knowledge relating to the specific concept it highlights. A semantic frame is defined as a set of concepts, related in a way that without knowledge of the complete set, one cannot understand a single concept in that set. A common example is the situation of commercial transfer. Frame Semantics states that it is not possible to understand the word “buy” without knowing anything about the situation of commercial transfer, which involves, among other things, a buyer, a seller, goods, money, and the relation between them. This example also illustrate that in Frame Semantics a word specifies a perspective in which a frame can be viewed: “buy”, for example views the situation from the perspective of the buyer, whereas “sell” views the same situation from the seller’s perspective.

FrameNet’s aim is to document these observations through computer-assisted annotation of example sentences. At the time of writing–FrameNet is still in development–the latest release (1.3) comes with a lexical database containing more than 8,900 lexical entries of which more than 6,100 are fully annotated. These entries are organized in more than 825 semantic frames and exemplified in more than 135,000 annotated sentences. Figure 3.1 shows a subset of the annotated example sentences for “buy.v”. Figure 3.2 lists the semantic roles–frame elements–used for “buy.v”, and thus can serve as a legend for Figure 3.1.



- np-ppfrom

1. Luckily we had BOUGHT them from friends who ran a nearby saddler 's shop ; unluckily they only had coarse blades in stock for our make of clippers .
2. As she was only about fourteen , we always BOUGHT a few dark-red carnations from her , but that was as far as we would go , to her surprise and indignation .
3. We BOUGHT our cottage from him .
4. The agreement to BUY seventy-two new pubs from Inntreprenuer Estates Limited will bring the total number of Morland pubs to just under four hundred . CNI
5. The common link was that they were long-term drug abusers , and they all BOUGHT their heroin from pushers in the streets of San Jose and Watsonville , in June and July .
6. From about 1979 onwards she started to BUY Welsh tweed from the Cambrian Woollen Mill in mid-Wales .

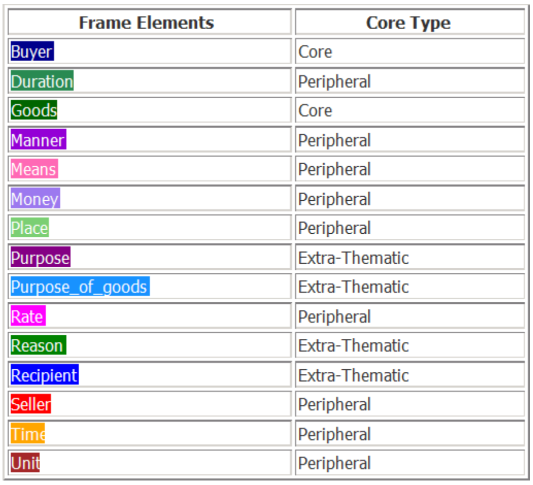
- np-ppwith

1. You ca n't BUY me with a few armfuls of flowers . "
2. But be assured , I had no intention of trying to BUY your favours with the price of a meal . "

- np-ppother

1. I think she BOUGHT it as an investment and wants to sell .
2. It was still possible in early 1922 to BUY foodstuffs in the city markets if one had the money . NI
3. In 1984 , it was perfectly possible to BUY tickets on the morning of the Grand Slam decider with France . NI

Figure 3.1: Some annotated example sentences as found in the FrameNet database for “buy.v”. (Screenshot from http://framenet.icsi.berkeley.edu)



Frame Elements	Core Type
Buyer	Core
Duration	Peripheral
Goods	Core
Manner	Peripheral
Means	Peripheral
Money	Peripheral
Place	Peripheral
Purpose	Extra-Thematic
Purpose_of_goods	Extra-Thematic
Rate	Peripheral
Reason	Extra-Thematic
Recipient	Extra-Thematic
Seller	Peripheral
Time	Peripheral
Unit	Peripheral

Figure 3.2: List of frame elements in the *Commerce_buy* frame. (Screenshot from http://framenet.icsi.berkeley.edu)

As already indicated, FrameNet organizes its lexical entries (lexical units) in frames. Each frame captures a set of words closely related in meaning. “buy.v”, for example shares a frame called *Commerce_buy* with the lexical units *purchase.v* and *purchase_((act)).n*. Between frames eight types of frame-to-frame relations exist, which are listed in Table 3.1.

Relation	**Sub**	**Super**
Inheritance	Child	Parent
Perspective_on	Perspectivized	Neutral
Subframe	Component	Complex
Precedes	Later	Earlier
Inchoative_of	Inchoative	State
Causative_of	Causative	Inchoative/State
Using	Child	Parent
See-also	Referring Entry	Main Entry

Table 3.1: Types of frame-frame relations in FrameNet

Figure 3.3 illustrates how Commerce buy is related to Commerce sell, which contains six lexical units retail.v, retailer.n, sale.n, sell.v, vend.v and vendor.n. A subset of the example sentences FrameNet lists for one of these entries (sell.v) is listed in Figure 3.4. When comparing this figure with the earlier Figure 3.1, it becomes cleat that both lexical units (actually both frames these lexical units are in) use the same frame elements.

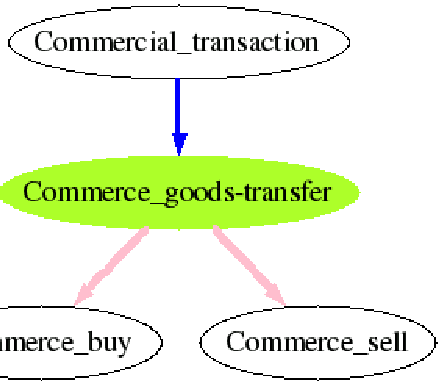
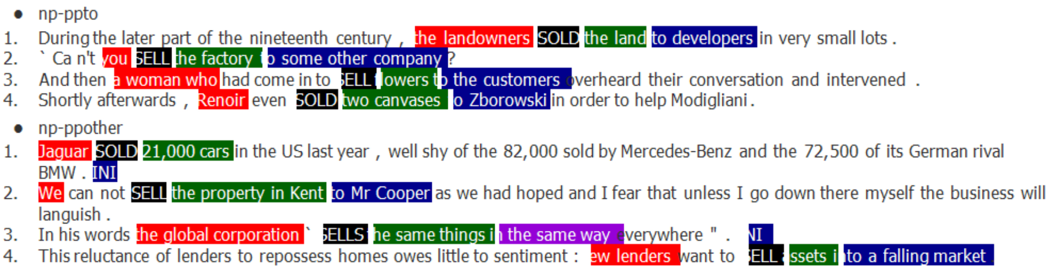


Figure 3.3: Frame-frame relations between *Commerce_buy* and *Commerce_sell*. Pink symbolizes the *Perspective_on* relation, while blue represents the *Subframe* relation.



- np-ppto

1. During the later part of the nineteenth century , the landowners SOLD the land to developers in very small lots .
2. ` Ca n't you SELL the factory o some other company ?
3. And then a woman who had come in to ELL lowers o the customers overheard their conversation and intervened .
4. Shortly afterwards , Renoir even SOLD two canvases o Zborowski in order to help Modigliani .

- np-ppother

1. Jaguar SOLD 21,000 cars in the US last year , well shy of the 82,000 sold by Mercedes-Benz and the 72,500 of its German rival BMW . INI
2. We can not SELL the property in Kent to Mr Cooper as we had hoped and I fear that unless I go down there myself the business will languish .
3. In his words the global corporation ` ELLS he same things i the same way everywhere " . NI
4. This reluctance of lenders to repossess homes owes little to sentiment : ew lenders want to ELL ssets i to a falling market

Figure 3.4: Subset of annotated example sentences in FrameNet for “sell.v”. (Screenshot from http://framenet.icsi.berkeley.edu)

3.2.2 PropBank

PropBank [Palmer et al., 2005], [Kingsbury et al., 2002], [Kingsbury and Palmer, 2002] builds on the syntactic structures of the Penn Treebank [Marcus et al., 1994b], [Marcus et al., 1994a], to which it adds a layer of predicate-argument information. Unlike FrameNet, where linguistically interesting example sentences are manually selected by humans, it covers every instance of every verb in the Wall Street Journal part of the Penn Treebank. Because of its linkage to a syntactic annotated corpora, PropBank (unlike FrameNet) delivers fully POS tagged and parsed example sentences where the position of the semantic fillers for a given head verb are specified as nodes in the parse tree. Arguments are labeled from ARG0 to ARG5 and defined separately for each verb. Generalizations can be draw for lower-numbered arguments: ARG0 is usually used for the subject of transitive verbs, mostly corresponding to the *Agent* role, while ARG1 is usually used for *Patient* or *Theme* roles and assigned to objects of transitive verbs and the subjects of some intransitive verbs. No such generalizations can be made for higher-numbered labels. A different picture emerges for adjuncts which are consistently annotated across all verbs: ARGM-LOC, for example, stands for locatives and ARGM-TMP for temporals. In total, there are 12 secondary tags for ARGM labels, which are listed in Table 3.2.

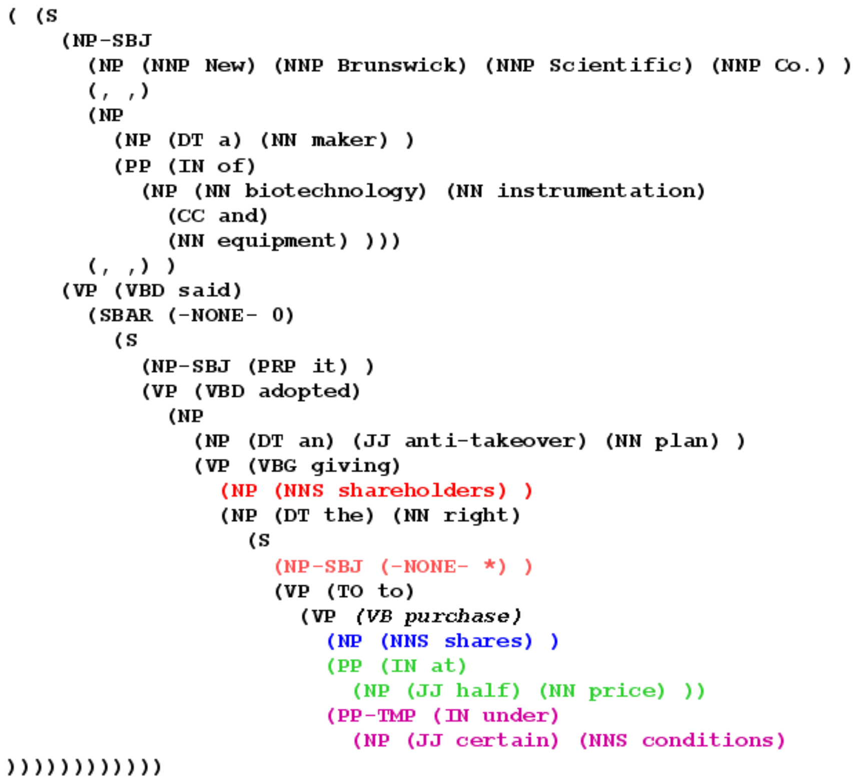
Tag	**Description**
EXT	extent
DIR	direction
LOC	location
TMP	temporal
REC	reciprocal
PRD	predication
NEG	negation
MOD	modal
ADV	adverbial
MNR	manner
CAU	cause
PNC	purpose not cause.
DIS	discourse

Table 3.2: List of all 13 secondary tags used for ARGM labels in PropBank.

Figure 3.5 shows an example of the data as it appears in the files of Penn Treebank distribution. Colours were added to highlight the roles: red is for ARG0, blue for ARG1, green for ARG3, violet for ARG-TMP. Orange also stands for ARG0, but it marks a trace associated with the constituents actual position.

PropBank's data files contain lists of all arguments for each of its entries, which also link arguments, whenever possible, to thematic (or theta) roles (essentially a small set of universal semantic roles used across all verbs, with about ten to 30 members, depending on the underlying theory) and additionally give a short description of the arguments. Two examples, once for *purchase.01* (sense 1 of 1) and once for *sell.01* (sense 1 of 3), can be seen in Table 3.3 and 3.4, respectively. When comparing these three points stand out:

1. The thematic roles do not allow a semantic linking of arguments between both



```
( (S
    (NP-SBJ
      (NP (NNP New) (NNP Brunswick) (NNP Scientific) (NNP Co.) )
      (, ,)
      (NP
        (NP (DT a) (NN maker) )
        (PP (IN of)
          (NP (NN biotechnology) (NN instrumentation)
            (CC and)
            (NN equipment) )))
      (, ,) )
    (VP (VBD said)
      (SBAR (-NONE- 0)
        (S
          (NP-SBJ (PRP it) )
          (VP (VBD adopted)
            (NP
              (NP (DT an) (JJ anti-takeover) (NN plan) )
              (VP (VBG giving)
                (NP (NNS shareholders) )
                (NP (DT the) (NN right)
                  (S
                    (NP-SBJ (-NONE- *) )
                    (VP (TO to)
                      (VP (VB purchase)
                        (NP (NNS shares) )
                        (PP (IN at)
                          (NP (JJ half) (NN price) ))
                        (PP-TMP (IN under)
                          (NP (JJ certain) (NNS conditions)
))))))))))))
```

Figure 3.5: Example of an annotated sentence in PropBank. See text for details.

verbs: What used to be the *Seller* role in FrameNet, in PropBank becomes *Source* and *Agent*, respectively.

2. The descriptions are just that, descriptions: While “seller” and “price paid” are consistent, what would be FrameNet’s *Buyer* role is once called “purchaser” and once “buyer”, FrameNet’s *Goods* becomes “thing purchased” and “thing sold”, respectively.

3. The description “price paid” occurs twice and while it once gets assigned to a thematic role called *Asset*, on the other occasion no thematic role is specified.

3.2.3 VerbNet

VerbNet [Schuler, 2005], unlike both other resources, does not annotate example sentences. Rather it lists abstract frame structure for its entries which are, as the name suggests, all verbs. It uses Levin verb classes [Levin, 1993] to construct its lexical entries, which are hierarchically organized to ensure that all their members have common semantic and syntactic properties. Each class in the hierarchy contains a set of

role	theta	description
ARG0	Agent	purchaser
ARG1	Theme	thing purchased
ARG2	Source	seller
ARG3	Asset	price paid
ARG4	-	benefactive

Table 3.3: Frame for “purchase” (sense 1 of 1) in PropBank.

role	theta	description
ARG0	Agent	seller
ARG1	Theme	thing sold
ARG2	Source	buyer
ARG3	-	price paid
ARG4	-	benefactive

Table 3.4: Frame for “sell” (sense 1 of 2) in PropBank.

verbs and a set of syntactic frames and semantic predicates applicable to them. The hierarchical structure of the resource reduces the effort to construct the lexicon and also allows to identify common syntactic and semantic behavior of verbs, something that is not possible with the empirical approach undertaken by PropBank and only to a lesser extent with FrameNet. (FrameNet organizes its entries semantically, but there is no way to immediately recognize that two verbs show the same argument structures.)

The verb “purchase”, for example is listed in a class with the ID *obtain-13.5.2-1*, which also contains “acquire” and “obtain”. Table 3.5 shows the verb argument structures listed in this class.

Frame	Example

`Agent[NP] VERB Theme[NP]`
`"for" Asset[NP]`
	“Carmen purchased a dress for $50.”

`Asset[NP] VERB Theme[NP]`	“$50 won’t even purchase a dress.”

Table 3.5: Entries in VerbNet’s *obtain-13.5.2-1* class.

This class is a subclass for the much larger class *obtain-13.5.2* which contains entries for the verbs “accept”, “accrue”, “accumulate”, “appropriate”, “borrow”, “cadge”, “collect”, “exact”, “grab”, “inherit”, “receive”, “recover”, “regain”, “retrieve”, “seize”, “select” and “snatch”. Table 3.6 shows the verb argument structures listed here.

Frame	**Example**
`Agent[NP] VERB Theme[NP]`	"Carmen obtained the spare part."

`Agent[NP] VERB Theme[NP]`
`"from" Source[NP]`
	"Carmen obtained the spare part from Diana."

Table 3.6: Entries in VerbNet's *obtain-13.5.2* class.

While the verb argument structures in the subclass can only be used for the verbs in that class, the structures in the super class can be used for the super- and the subclasses verbs.

3.3 Related Work

This section gives an overview about work carried out by other researchers which is of significance to the approach to Question Answering described in this chapter. It starts with automatic role labeling, which is a task that any NLP application based on semantic roles has to deal with in one way or another. Therefore it provides some ground work on which other applications build. We then take a look at how FrameNet, PropBank and VerbNet have been used to date in Question Answering and to what effect. Finally, we describe a few selected interesting applications of these resources outside Question Answering.

3.3.1 Automatic Role Labeling

Automatic Role Labeling is the task to assign labels expressing semantic relationships–or semantic roles–to certain constituents in a sentence. The annotated roles can be of an abstract nature such as *Agent* or *Patient* (as in PropBank, see Section 3.2.2), or much more specific such as *Buyer*, *Seller* and *Goods* (as in FrameNet, see Section 3.2.1).

The first paper describing an algorithm for this task is [Gildea and Jurafsky, 2002], where the authors train a classifier on roughly 40,000 of the annotated FrameNet sentences. Each of these sentences is parsed and the following lexical and syntactic features are extracted:

Phrase Type The syntactic category of the phrase expressing the semantic role, e.g. NP, PP, ADVP etc.

Parse Tree Path The syntactic relation between the predicate invoking the semantic frame and the constituent in question. Phrase structure representations are used.

Governing Category (restricted to NPs) The first node reached of type *S* or *VP*, moving up the parse tree from the constituent corresponding to the frame element.

Position The relative position of the constituent to be labeled, either *before* or *after* the predicate defining the semantic frame.

Voice Either *active* or *passive*.

Head Word The head word of the constituent.

These features are combined with other information such as knowledge of the predicate and prior probabilities of various combinations of semantic roles. Their system achieves 82% accuracy in identifying the correct semantic role of a constituent if their boundaries are manually pre-assigned. At the more challenging task of simultaneously segmenting constituents and identifying their semantic role, the system achieves 65% precision and 61% recall.

Most of the work following [Gildea and Jurafsky, 2002] sticks to their general approach of using a statistical classifier, but modifies the feature set and/or the classifier used. [Xue and Palmer, 2004], for example, show that a more careful feature selection–especially by taking more information from the target sentence's parse trees into account–leads to an overall better performance. Furthermore they argue that the argument identification and the argument classification subtasks require the use of different features. In [Pradhan et al., 2005b] Support Vector Machine classifiers are used for the task of semantic role labeling. The authors add new features including some extracted from CCG parses, perform feature selection and calibration and combine parses obtained from several semantic parsers. The latter is motivated by an analysis stating that parse errors account for about half of the total mistakes of the author's role labeling system.

Automatic Role Labeling is a challenging task when performed on free text. However, the problem formulation in this thesis, because of its QA setting, is slightly different from the one used in the above papers. For our work, the most crucial part is assigning correct roles to questions (see Section 3.4). Usually, questions are much shorter than their answer sentences (Section 4.4.2 reports an average length of 8.14

words for questions in the QASP corpus, versus an average length of length of 28.99 words for answer sentences.) and therefore also syntactically less complex. Thus annotating questions can be expected to be easier that annotating declarative sentences. Yet, the fact that shallow semantic parsers are usually trained on declarative sentences might create a mismatch between training and test data when using them to parse questions. In the next section, when looking at previous work employing semantic roles in QA, we will see that some researchers reported problems in this respect.

3.3.2 Semantic Roles in Question Answering

This section reports on the of the research carried out at the intersections of Question Answering and semantic roles. As mentioned the first work in Semantic Role Labeling started with [Gildea and Jurafsky, 2002] in 2000. [4] The first papers using semantic roles in NL applications is [Surdeanu et al., 2003], which appeared in 2003. For Question Answering this was [Narayanan and Harabagiu, 2004] is 2004. Up until the time of writing (early 2009), still not much research has been dedicated at assessing whether semantic role information can be beneficial for Question Answering. Many of the papers we will take a look at in this section describe Question Answering systems that use one or more of the three resources in conjunction with or as extensions to other algorithms. Very few of these papers report evaluation results on what the contributions of the resources alone to system performance are. Early versions of our own work on Semantic Roles in Question Answering have been published in [Kaisser, 2005] and [Kaisser, 2006]. Final versions, describing all algorithms as they are detailed in this thesis, have been published in [Kaisser et al., 2006] and [Kaisser and Webber, 2007]. We therefore claim that our work has been one of the first, if not the first, that performed a study especially dedicated to the use of semantic roles in Question Answering. It is furthermore the only study so far that uses all three resources, FrameNet, ProbBank and VerbNet, compares their performance and uses them in conjunction in a Question Answering system.

[Narayanan and Harabagiu, 2004] is commonly considered the first paper employing semantic roles for Question Answering. Here, the authors present a QA system for complex questions that identifies predicate argument structures and semantic frames in the document collection and performs probabilistic inference using the extracted

[4] An shorter version of this paper was published as [Gildea and Jurafsky, 2000]

relations in the context of a domain and scenario model. The argument structures the system identifies are based on PropBank, while the semantic frames come from FrameNet. The authors automatically annotate questions and candidate sentences in PropBank and FrameNet terms and retrieve the answers if the questions and a candidate sentence overlap. However, the paper mainly focuses on the use of Coordinated Probabilistic Relational Models to perform probabilistic and temporal inferences, thus their methods with regards to the use of FrameNet and PropBank are not explained in great detail. Also, the evaluation section of the paper only gives a few hints as to what contributions FrameNet and PropBank make to system performance. Rather than evaluating the system's ability to return correct answers, the authors evaluate number of correct answer types identified. They report that they can identify the correct argument role of the answer (for PropBank) in 32% and the correct frame element (for FrameNet) in 19% of all cases. These figures seem low, but the authors mention that they " have used a set of 400 questions pertaining to four different topics:(T1) UN inspections; (T2) Thefts in Russia's nuclear navy, (T3) Status of India's Prithvi ballistic missile project and (T4) China's participation in non-proliferation regimes." The authors claim that using this highly specialised question set makes the correct assignment of answer types harder. While this seems likely, it unfortunately also makes their results uncomparable to other work conducted in the field. Overal, the authors report that "52% of the extracted answers were correct."

[Fliedner, 2004] described the functionality of a planned system based on the German version of FrameNet, SALSA [Burchardt et al., 2006]. As the system is reported to still be in its design phase, no evaluation results are given. Since then, no paper (that we are aware of) describing the completed system has been published.

[Novischi and Moldovan, 2006] use a technique that builds on a combination of lexical chains and verb argument structures extracted from VerbNet to re-rank answer candidates. The authors' aim is to recognize changing syntactic roles in cases where an answer sentence shows a head verb different from the question. In the paper they give an example based on the question "When was it established?" with the target "Abercrombie & Fitch" (question 28.2 from the TREC 2004 test set) and the answer sentence "... Abercrombie & Fitch began life in 1982 ...". Between the two verbs "begin" from the answer sentence and "establish" from the question the following lexical chain can be found in WordNet:

SynSet: (v-begin#2, start#4)
Relation: R-CAUSATION
SynSet: (v-begin#3, lead off#2, start#2, commence#2)
Relation: SIM-DERIV
SynSet: (v-establish#2, found#1)

The thematic structures found in VerbNet for each of the verbs are then propagated along this chain in the following way:

```
[Patient="Abercrombie & Fitch"] begin\#2 [Theme=n-life\#2]
[Agent=X] begin\#3 [Patient="Abercrombie & Fitch"]
[Agent=X] establish\#2 [Patient="Abercrombie & Fitch"]
```

Because the final structure matches the argument structure from the question their system concludes that the candidate sentence is likely to be valid and it receives the highest rank during the candidate ranking phase.

However, since VerbNet is based on *thematic* roles there are problems when using it like this. This can be illustrated by the following VerbNet patterns for *buy* and *sell*:

```
[Agent] buy [Theme] from [Source]
[Agent] sell [Recipient] [Theme]
```

Starting with the sentence “Peter bought a guitar from Johnny”, and mapping the above roles for *buy* to those for *sell*, the resulting paraphrase in terms of *sell* would be “Peter sold UNKNOWN a guitar”. That is, there is nothing blocking the Agent role of *buy* being mapped to the Agent role of *sell*, nor anything linking the Source role of *buy* to any role in *sell*. Furthermore, the authors face a massive coverage problem: The authors report that their approach can be applied to only 15 of 230 TREC 2004 questions. They report a performance gain of 2.4% (Mean Reciprocal Rank based on the top 50 answers).

[Sun et al., 2005] and [Schlaefer et al., 2007] both describe QA systems participating in TREC which make use of ASSERT [Pradhan et al., 2004], a publicly available shallow semantic parser trained on PropBank, to annotate questions and candidate sentences with predicate-argument structures. The annotations of questions and candidate sentences are subsequently compared. Both papers are typical for TREC QA track papers in that they describe a range of new methods used in the respective systems (both systems had already participated in earlier years), but report overall performance of the complete systems in TREC's evaluation. The papers do not give independent evaluation figures about their approaches based on semantic role labeling. Yet both papers comment on ASSERT's recall and mention that it is (as of yet) not high enough: For many questions and answer sentences the parser returns no or incomplete role assignments, leaving consequent processing steps without data to work with.

In [Shen and Lapata, 2007] the authors enhance the answer extraction module of a pre-existing QA system by incorporating a custom-build semantic role assignment module based on FrameNet which assigns roles to questions and answer sentences. This module is based on dependency paths and assigns roles by comparing the paths between the predicate and its roles found in FrameNet's annotated sentences with all paths found in a candidate sentence. The authors find that their FrameNet enhanced answer extraction module significantly outperforms a similar module that does not use FrameNet. In comparison they found that their baseline method, which uses the publicly available shallow semantic parser Shalmaneser [Erk and Pado, 2006] to annotate questions and answer sentences does not improve performance. Similar to [Sun et al., 2005] and [Schlaefer et al., 2007] they note that the shallow semantic parser tends to favor precision over recall, thus reducing the number of questions for which answers can be found.

In [Moschitti et al., 2007] the authors study the impact of syntactic and shallow semantic information in automatic question classification and answer re-ranking for a web-based QA system. To this end they employ a tree kernel–a mathematical formalism to measure similarity between two trees by measuring how many of their substructures are identical. The paper introduces a new type of tree kernel, the Shallow Semantic Tree Kernel, which is able to evaluate predicate argument structure trees and allows partial matches. They evaluate this kernel against baselines which are based on bag-of-words, bag-of-POS-tags and parse trees and combinations thereof. For the question classification task they report no improvement when including information from predicate argument structures. The authors note that since questions tend to be short and have few verbal predicates the potential of predicate argument structures can not be fully exploited in question classification. However they find that predicate argument structures are useful for answer classification, where the system has to deal with longer sentences.

Finally, [Ofoghi et al., 2009] address the issue of a QA system that uses a state-of-the-art shallow semantic parser for question answering and provide numbers on its performance. They implement the most obvious approach, also described in [Sun et al., 2005] and [Schlaefer et al., 2007]: Questions and answer sentences are annotated with semantic roles (coming from FrameNet) with the help of a shallow semantic parser, here Shalmaneser [Erk and Pado, 2006], and then both annotations are compared. If an overlap is detected and the answer role is filled, it can be extracted as the answer. They compare this approach against a baseline which locates the answer with help of a named entity system. The baseline system achieves a MRR of 0.400 on a partial TREC 2004 test set. This test sets consists of 143 factoid questions (out of a total 230), for which their IR system returned minimum one correct answer in the top 10 passages. When combining their baseline system with the methods based on frame semantics their performance drops to 0.347. However, if they manually correct Shalmaneser's output their performance increases considerably to 0.520. They also present numbers on labeling accuracy, here defined as the ratio of the number of correctly assigned frames or frame elements by the role labeller to the whole number of frames or frame elements assigned by humans, both for questions and answer sentences from the AQUAINT corpus. Based on the top ten passages returned by the IR module this is 41.8% for frames and 17.0% for frame elements. For questions this is 59.2% for frames and 60.3% for frame elements. This illustrates what was so far only hinted at in the literature: The main reason why such an approach does not work is the suboptimal performance of state-of-the-art shallow semantic parsers.

3.3.3 Other Uses of Lexical Resources in NLP

This section describes work in NLP, but outside QA, that makes use of FrameNet, PropBank and/or VerbNet. While not directly related to the work described in this thesis, it is relevant in a broader sense because it shows that these resources can (and have) also be used in other fields of NLP.

We start with Textual Entailment, another application of NLP beside QA where the notion of paraphrase is highly relevant. Starting with two text fragments called *Text* and *Hypothesis*, Textual Entailment Recognition is the task of determining whether the meaning of the Hypothesis can be inferred from the Text. In 2006 two papers were presented at the PASCAL Recognizing Textual Entailment Challenge [Bar-Haim et al., 2006] that are based on lexical resources:

[Burchardt and Frank, 2006] present a baseline approach to the textual entailment task in the PASCAL RTE Challenge. The paper assesses whether entailment can be approximated in terms of structural and semantic overlap of text and hypothesis by combining LFG parsing with FrameNet frames and frame structures. LFG f-structures are used with frame semantics projections for text and hypothesis pairs. Structural and semantic similarities are recognized and represented in a match graph. Features are extracted from the text, the hypothesis and the match graph in order to characterize their syntactic and semantic properties, as well as various proportional measures potentially relevant for entailment. These features are then used to train a machine learning model. Their best run classifies 59% of the pairs in the 2006 PASCAL RTE Test Set correctly.

In [Hickl et al., 2006] the authors describe an approach to textual entailment that combines lexico-semantic information with a large collection of paraphrases acquired automatically from the web. In their system, called GROUNDHOG, text-hypothesis pairs are sent to a text preprocessing module, where they are syntactically parsed and processed by a Named Entity Recognition system. Semantic dependencies are identified using a semantic parser trained on PropBank's predicate-argument annotations. To determine whether an entailment relationship exists for a text-hypothesis pair, the system uses an Entailment Classifier, based on decision trees. Four of the features used in the classifier are based on the PropBank-style annotations:

entity arg-match is a Boolean feature which fires when aligned entities in the text and hypothesis show the same argument role label.

entity near arg-match collapses ARG1 and ARG2 into one category, and all the ARGM features in another category.

predicate arg-match feature which compared the role labels associated with aligned predicates.

predicate near arg-match feature which is a less exact version of the former feature.

The complete system classifies 75.38% of the pairs in the 2006 PASCAL RTE Test Set correctly. When only the four features computed from the PropBank style annotations are used this figure is 62.50%.

In the field of Information Extraction (IE) [Surdeanu et al., 2003] describes a domain-independent IE paradigm based on PropBank's predicate-argument structures, which are automatically identified either by the method reported in [Gildea and Palmer, 2002] or by a new method based on inductive learning, with an extended feature set that includes named entity information. Predicate-argument structures are used to identify and extract relevant information, dictated by *templettes*, essentially frame-like structures with slots representing the event's basic information (for example main event participants, event outcome, time and location). The fact that these predicate-argument structures are very similar in nature to the templettes, makes it easy to transform the first to the latter by mapping predicate arguments into templettes slots (a set of manually created rules is used for this). When evaluated on two domains ("market change" and "death"), their system performs 10% worse when compared to an IE system based on hand-crafted patterns, but the authors note that much less human effort is necessary to adapt the system to a new domain.

Another line of research is concerned with combining the individual resources. [Shi and Mihalcea, 2005] describe how they integrate FrameNet, VerbNet, and WordNet into a unified, richer knowledge-base, to enable more robust semantic parsing. By doing this, they extend FrameNet's coverage, augment VerbNet with frame semantics, and implement selectional restrictions using WordNet semantic classes. Essentially they do two things: Firstly, they connect VerbNet to WordNet. Here they link all 36 semantic constraints which are imposed on the arguments of syntactic frames defined in VerbNet to one or more nodes in the WordNet semantic hierarchy. Secondly, they match VerbNet with FrameNet data. Here, they map VerbNet entries to corresponding semantic frames in FrameNet. Furthermore, they map VerbNet's syntactic frame arguments with FrameNet's semantic roles. The achieved unified resource is then used by a rule-based semantic parser which the authors claim has significantly larger coverage than statistical parsers based on a single resource.

In a very similar vein, [Giuglea and Moschitti, 2006] describe a robust semantic parser, build on a broad knowledge base created by interconnecting FrameNet, VerbNet and PropBank. VerbNet and FrameNet are connected by mapping FrameNet frames to

VerbNet classes. The PropBank corpus is used to increase the verb coverage, which, once the mapping from FrameNet to VerbNet is achieved, is rather unproblematic because VerbNet entries already contain links to the corresponding PropBank entries. This work differs from [Shi and Mihalcea, 2005] in that VerbNet roles are assigned to FrameNet frames and not via versa and in that the semantic parser is statistical not rule based (it is described in [Moschitti et al., 2005]).

3.4 Question Answering by Natural Language Generation

In this section we describe the first of two methods using FrameNet, PropBank and VerbNet in a web-based Question Answering system. It uses the data available in the resources to generate potential answer templates to the question. While at least one component of such a template (the answer) is yet unknown, the remainder of the sentence can be used to query a web search engine. The results can then be analyzed, and if they match the originally-proposed answer template structure, an answer candidate can be extracted. The basic approach we use is similar to [Dumais et al., 2002] in that a web search engine is fed with partial answer sentences gained from reformulating the question. While the reformulation procedure in [Dumais et al., 2002] is string based, the QuALiM system (see Chapter 2) uses reformulations which are based on syntax, thus enabling a wider range of more exact reformulations and the extraction of exact answers. (The approach in [Dumais et al., 2002], just because it lacks syntactic knowledge, returns only passages.) The method proposed in this section is based on QuALiM's approach but further enhances it by enabling the system to create reformulations based on the data in FrameNet, PropBank and VerbNet. This makes a much wider range of paraphrases available to the system. In the following, we will give a detailed account of our method and its implementation. As an example we will use the question "Who purchased YouTube?"

The first processing step is to parse the incoming question using MiniPar, a dependency parser [Lin, 1998b], and Shalmaneser, a publicly available shallow semantic parser [Erk and Pado, 2006]. MiniPar returns the following list of dependency nodes:

```
E1   noWord     noBase     noPar  noPOS  noRel
E0   noWord     fin        E1     C      noRel
1    Who        who        E0     N      whn
```

```
2     purchased   purchase   E0     V      i
E2    noWord      who        2      N      subj
3     YouTube     YouTube    2      N      obj
```

Here, each line represents one node. The first column gives the node's identifier, the second the node's surface, if any; the third column states the node's lemma; the fourth gives the identifier of the node that is this nodes's head; the fifth column shows the node's part of speech; and the sixth lists the relationship between the node and its head. We are mainly interested in the questions's head verb and its arguments and thus simplify MiniPar's output to the following structure:

```
head: purchased(V)
subj: Who
whn:  Who
obj:  YouTube
```

Here, `head` indicates that the head of the question is the verb *purchased*, `subj` indicates that the deep subject is *who* (which `whn` marks as also being a question word) and `obj` indicates that the deep object is *YouTube*.

Beside what is shown above, the system analyzes the tense as being *Simple Past*. In order to determine the tense of a questions (and in a later processing step, answer sentences), we compare them against a XML file named tenses.xml containing basic information about the English grammar of tenses, which was created especially for this purpose. Below is an excerpt, showing the entries for Simple Past and Past Progressive:

```
<tense name="Past Tense">
  <form aspect="Simple" voice="Active">
    <subj/>
    <verb             flection="PAST"  agr="subj"/>
  </form>
  <form aspect="Progressive" voice="Active">
    <subj/>
    <aux  base="be"   flection="PAST"  agr="subj" lex="was|were"/>
    <verb             flection="PROG" />
  </form>
  ...
</tense>
```

In the case of the question “Who purchased YouTube?” the system recognizes that here the subject “Who” is followed directly by the head verb in its Simple Past form “purchased.” This matches the first of the forms given in the excerpt, which is marked as being “Past Tense” with aspect “Simple” and voice “Active”.

Once this analysis is done, we look up the head verb in one of the lexical resources (in this example FrameNet) where exactly one lexical unit for *purchase.v* can be found. There, we find 75 associated annotated sentences, one of which is:

The company	had	PURCHASED	several PDMS terminals	, but has been ...
FE:Buyer		lexical unit	FE:Goods	

As can be seen, parts of the sentences are annotated with frame elements, here *Buyer* and *Goods*. The system will parse and simplify the annotated sentences until a set of abstract frame structures, similar to those in VerbNet, is achieved. This is done by intentionally removing words associated with certain levels of information that were present in the original data, i.e. tense, voice, mood and negation. (In a later step some of it we be reintroduced.) For the above example, “had” is recognized as being a part of the tense construction indicating that this sentence is in Past Perfect. (The already mentioned tenses.xml file is used for this.) Therefore, the verb complex “had purchased” is reduced to an abstract element “VERB”. Furthermore, the surface structure of the NPs labeled with frame elements are removed. The sentence part “but has been having difficulty in using them effectively” (only partially mentioned before) is completely removed, because it contains no annotations. The resulting abstract structure is:

```
Buyer[Subj,NP] VERB Goods[Obj,NP]
```

Beside the structure given, similarly extracted from other annotated sentences in FrameNet are:

```
Buyer[Subj,NP] VERB Goods[Obj,NP] Seller[Dep,PP-from]
Buyer[Subj,NP] VERB Goods[Obj,NP] Money[Dep,PP-for]
Buyer[Subj,NP] VERB Goods[Obj,NP] Recipient[Dep,PP-for]
 ...
```

This shows that usually, for this particular example, in active sentences, the *Buyer* role is realized as an NP at subject position, while *Goods* is an NP at object position.

From information like this the system creates a table indicating how often each syntactic function is realized as which semantic role. As mentioned earlier, the analysis of the question showed that the question word (and as such the answer slot) is in subject relation to the verb “purchase”. Furthermore, “YouTube” needs to be in object relation to the verb. Taking this information together with the most commonly observed syntactic function/semantic role combinations, it can be concluded that the filler for the *Goods* frame element is “YouTube”, and that the question asks for a *Buyer*. We then compare the role assignment we just achieved with Shalmaneser’s result. For the example at hand both methods agree. If we would get two different assignments, we would from now on perform all processing steps in parallel for both possibilities. If no method produces a (complete) result, we cannot process the question further.

This last point is also the main motivation for the described procedure. We need a complete role assignment (meaning that all question constituents have a role assigned), in order to proceed. Yet, role assignment is brittle (see Sections 3.3.1 and 3.3.2). By combining our heuristic with a shallow semantic parser’s output we increase the chance of obtaining a result. (Note that the method explained in this section, as will become clear in the following, does not need to annotated answer sentences.)

For the case at hand, role assignment suceeds, so the system can give a pseudo-semantic formula for the question:

```
purchase_2971(Buyer=X, Goods="YouTube")
```

This and the fact that the question was asked in past tense, enables the approach to create the following potential answer templates by alternating all possible past tense forms and the voice:

```
ANSWER[NP] purchased YouTube
ANSWER[NP] (was|were) purchasing YouTube
ANSWER[NP] (has|have) purchased YouTube
ANSWER[NP] had purchased YouTube
YouTube (was|were) purchased by ANSWER[NP]
...
```

To do this, the tense and voice information from the already mentioned tenses.xml file is used. For the second example (`ANSWER[NP] (was|were) purchasing YouTube`) the second form in the tense.xml excerpt given above tells us that the Past Progressive form in active voice consists of the verb “be” in its past form (either “was” or “were”), followed by the progressive form of the verb (here “purchasing”). This allows two possibilities for the verb complex: “was purchasing” or “were purchasing”.

Then, the part (or parts) of the templates that are known are quoted and sent to a search engine (Google for our experiments). For the above examples, the queries are:

```
"purchased YouTube"
"was purchasing YouTube"
"were purchasing YouTube"
"has purchased YouTube"
"have purchased YouTube"
"had purchased YouTube"
"YouTube was purchased by"
"YouTube were purchased by"
```

One point that needs to be considered here is grammatical agreement. In our example this means that in order to create fully grammatical sentence parts the auxiliary verb has to be in agreement with the surface subject. When generating active sentences, the subject is not known because it is the answer to the question, thus we cannot know the grammatical number of the answer in advance. Therefore both possibilities are generated. For active sentences we do know the subject. Thus, in theory, it would be possible to devise an algorithm that only generates templates with correct agreement. We opted for the simpler version and instead generate queries for all possibilities–mainly because In English there are just two. We expect the grammatically correct possibility to return more hits that the grammatically incorrect version. This is because it can be assumed that there are more grammatically correct sentences on the web than grammatically incorrect ones. (`"YouTube was purchased by"`, at the time of writing returns 1,210 hits on Google, while `"YouTube were purchased by"` returns just one.)

After the queries are created and the search engine is queried, sentences are extracted from the top 50 returned snippets per query and from these, candidate sentences are extracted. These are matched against the abstract frame structure from which the queries were originally created. For the above query `"YouTube was purchased by"`, for example, Google reports at the time of writing 1,200 results, the first being:

“YouTube was purchased by Google in 2006 for approximately 1.6 billion dollars, even though YouTube was not yet earning a profit.”

The sentences are parsed and can easily be matched to the template `YouTube (was|were) purchased by ANSWER[NP]` from which the search query was created. In that way, “Google” is determined as the answer, because it is the NP following directly after the template parts that were used for the query. We also achieve, almost as a side effect, a role assignment for relevant part of the answer sentence. Thus, for the given example, the system was able to find the correct, exact answer and the open proposition shown earlier can now be completed:

```
purchase_2971(Buyer="Google", Goods="YouTube")
```

Note that although the described approach uses an example based on FrameNet, the data provided in PropBank and VerbNet is used in a similar fashion. In the case of PropBank the procedure is essentially the same. When using VerbNet, one processing step can be skipped. This is because VerbNet does not list example sentences, but returns the abstract frame structure directly.

While this is the general approach we use to locate answers, one additional point is crucial to note. So far, only questions whose answer role is an argument of the head verb were discussed. However, for some question classes (especially time- or location-questions) this assumption does not hold. Here, the answer to the question is usually realized as an adjunct. This is an important difference for at least three reasons:

1. FrameNet and VerbNet do not or only sparsely annotate peripheral adjuncts. (PropBank however does.)
2. In English, the position of adjuncts varies much more than those of arguments.
3. In English, different kinds of adjuncts can occupy the same position in a sentence, although naturally not at the same time.

The following examples illustrate point 2:

YouTube was purchased by Google on October 9.

On October 9, YouTube was purchased by Google.

YouTube was purchased on October 9 by Google.

All variations are possible, although they may differ in frequency. PPs conveying other adjuncts could replace all the above temporal PPs, or they could be added at other positions.

These observations have to be accounted for, both when assigning semantic roles to questions and when creating and processing potential answer sentences. When annotating the answer role in a question which asks for an peripheral adjunct, the syntax of the question is of little help. Instead, the answer type of the question has to be consulted. (See Section 2.2.5 for an explanation of how the system processes answer types.) This means that certain answer types are matched to certain roles, e.g. whenever a temporal or location answer type is detected, the answer role becomes, in FrameNet terms, *Place* or *Time*, respectively. The approach then uses an abstract frame structure like the following to create the queries:

```
Buyer[Subj,NP,unknown] VERB Goods[Obj,NP,"YouTube"]
```

While this lacks a role for the answer, it still can be used to create a query like the following:

```
"has purchased YouTube"
```

When sentences returned from the search engine are then matched against the abstract structure, all PPs directly before the *Buyer* role, between the *Buyer* role and the verb and directly after the *Goods* role are extracted. Then all these PPs (if any) are checked on their semantic types and only those are kept that match the answer type of the question (if an answer type has been has been identified).

3.4.1 Making use of FrameNet Frames and Inter-Frame Relations

The method presented so far can be used with all three resources. But FrameNet goes a step further than just listing verb-argument structures: It organizes all of its lexical entries in frames, with relations between frames that can be used for a wider paraphrasing and inference. In the following we will explain how this information is used to generate additional answer templates.

As mentioned is Section 3.2.1, the *purchase.v* lexical unit, for example, is found in a *Commerce-buy* frame which also contains the lexical units *buy.v* and *purchase.n*. Both of these entries list annotated example sentences which use the same frame elements as *purchase.v*. Therefore, by using the same techniques which were explained earlier, we can produce reformulations based on these related entries:

```
ANSWER[NP] bought YouTube
```

```
ANSWER[NP] (has|have) bought YouTube
YouTube (has|have) been bought by ANSWER[NP]
...
```

Because FrameNet is not restricted to verbs, but lists other parts of speech as well, it is also possible to generate target paraphrases with heads which are not verbs, like:

```
ANSWER[NP-Genitive] purchase of YouTube
```

In fact, handling these is usually easier than sentences based on verbs, because no tense/voice information has to be introduced.

Furthermore, frames themselves can stand in different relations. The frame *Commerce_goods-transfer*, for example, stands both to the already mentioned *Commerce_buy* frame and to *Commerce_sell* in an *is_perspectivized_in* relation. The latter contains the lexical entries *retail.v*, *retailer.n*, *sale.n*, *sell.v*, *vend.v* and *vendor.n*. Here is one annotated example sentence listed in *sell.v*:

...	the landowner FE:Seller	SOLD lexical unit	the land FE:Goods	to developers FE:Buyer	...

As can be seen, the frame elements of these lexical units use the same labels as the frame elements in the *purchase.v* and *buy.v* entries. This enables us to create answer templates like:

```
YouTube was sold to ANSWER[NP]
```

Other templates created from this frame seem odd, e.g.

```
YouTube has been retailed to ANSWER[NP]
```

This is because the verb “to retail” usually takes mass-products as its object argument and not a company. But FrameNet does not make such fine-grained distinctions. However, we did not come across a single example during development where such a phenomenon caused an overall wrong answer. Sentences like the one above will most likely not be found on the web (just because they are in a narrow semantic sense not well-formed). Yet even if we would get a hit, it probably would be legitimate to count the odd sentence “YouTube had been retailed to Google” as evidence for the fact that Google bought YouTube.

3.5 Combining Semantic Roles and Dependency Paths

The method described in the last section uses precise, quoted search queries to locate potential answer sentences on the web. The main advantage, and indeed our main reason for deciding on this approach is that the results returned have a high probability of containing the correct answer: The method searches the web for obvious formulations of the answer fact. A potential disadvantage though is that some answer sentences might be formulated in completely different ways–and can therefore not be found. This applies for example to answer sentences where the important constituents are not adjacent to each other. To compensate for this, we implemented a second method based on the same resources. It poses abstract, unquoted search queries which are simply based on question key words. As a result, much less of the syntax of the candidate sentences we find on the web is known until we have found them and we therefore need a more powerful tool to match these candidate sentences to the annotated sentences in FrameNet and PropBank. (VerbNet does not list example sentences for lexical entries, so could not be used for this second method.) We decided to use dependency relations (more precisely dependency paths) for this task, mainly because dependency relations have been used by other researchers many times before and have shown to be suitable for this task. [Lin and Pantel, 2001, Rinaldi et al., 2003, Bouma et al., 2005a]

Here is how we proceed:

In a pre-processing step, all example sentences in PropBank and FrameNet are analyzed and the dependency paths from the head to each of the frame elements are stored. For example, in the sentence "The Soviet Union has purchased roughly eight million tons of grain this month" (found in PropBank), "purchased" is recognized as the head, "The Soviet Union" as *ARG0*, "roughly eight million tons of grain" as *ARG1*, and "this month" as an adjunct of type *TMP*. The stored paths to each are as follows:

headPath = $\Downarrow i$
role = *ARG0*, paths = $\{\Downarrow s, \Downarrow subj\}$
role = *ARG1*, paths = $\{\Downarrow obj\}$
role = *TMP*, paths = $\{\Downarrow mod\}$

This says that the head is at the root, ARG0 is at both surface subject (*s*) and deep subject (*subj*) position[5], ARG1 is the deep object (*obj*), and TMP is a direct adjunct

[5] MiniPar allows more than one path between nodes due, for example, to traces. The given example is MiniPar's way of indicating that this is a sentence in active voice.

(*mod*) of the head. The dependency paths used here represent paths in the parse tree of the corresponding sentence. Each one starts at the predicate and ends at the constituent filling one semantic role. Arrows indicate the direction in which the path moves in each step, either up ($\Uparrow$) or down ($\Downarrow$).

Semantic roles are assigned to questions as described in Section 3.4. Sentences that potentially contain answer candidates are then retrieved by posing a rather abstract query consisting of key words from the question. Once we have obtained a set of candidate-containing sentences and have parsed them, we ask the following questions of their dependency structures compared with those of the example sentences from PropBank[6]:

1a Does the candidate-containing sentence share the same head verb as the example sentence?

1b Do the candidate sentence and the example sentence share the same path to the head?

2a In the candidate sentence, do we find one or more of the example's paths to the answer role?

2b In the candidate sentence, do we find all of the example's paths to the answer role?

3a Can some of the paths for the other roles be found in the candidate sentence?

3b Can all of the paths for the other roles be found in the candidate sentence?

4a Do some of the surface strings of the other roles match those of the question?

4b Do all of the surface strings of the other roles match those of the question?

Tests 1a and 2a of the above are required criteria: If the candidate sentence does not share the same head verb or if we can find no path to the answer role, we exclude it from further processing.

Each sentence that passes steps 1a and 2a is assigned a weight of 1. For each of the remaining tests that succeeds we multiply that weight by 2. Hence a candidate sentence that passes all the tests is assigned a weight 64 times higher than a candidate that only passes tests 1a and 2a. We take this as reasonable, as the evidence for having found a correct answer is indeed very weak if only tests 1a and 2a succeeded and very high if all tests succeed. Whenever condition 2a holds, we can extract an answer

[6] Note that our process is not too different from what a role labeler would do: Both approaches are primarily based on comparing paths in parse trees. However, a standard role labeler would not take tests 3a, 3b, 4a and 4b into account.

candidate from the sentence: It is the phrase that the answer role path points to. All extracted answers are stored together with their weights, if we retrieve the same answer more than once, we add the new weight to the old ones. (This is the same procedure as described in Setion 2.2.4.) After all candidate sentences have been compared with all pre-extracted structures, the answer candidates are checked on their semantic type (see 2.2.5). This is especially important for answers that are realized as adjuncts, see Section 3.4. The answer candidate with the highest score is chosen as the final answer.

We now illustrate this method with respect to our question "Who purchased YouTube?" The roles assignment process produces this result: "YouTube" is *ARG1* and the answer role is *ARG0*. The web query used is simply `purchased YouTube` (without quotes) and we retrieve inter alia the following sentence: "Their aim is to compete with YouTube, which Google recently purchased for more than $1 billion." The dependency analysis of the relevant phrases is:

head = "purchased", path = ⇓*i*⇓*i*⇓*pred*⇓*i*⇓*mod*⇓*pcom-n*⇓*rel*⇓*i*
phrase = "Google", paths = {⇓*s*, ⇓*subj*}
phrase = "which", paths = {⇓*obj*}
phrase = "YouTube", paths = {⇑*i*⇑*rel*}
phrase = "for more than $1 billion", paths = {⇓*mod*}

If we annotate this sentence by using the analysis from the above example sentence ("The Soviet Union has purchased ...") we get the following (partially correct) role assignment: "Google" is *ARG0*, "which" is *ARG1*, "for more than $1 billion" is *TMP*.

The following table shows the results of the 8 tests described above:

1a	OK	2a	OK	3a	OK	4a	–
1b	–	2b	OK	3b	OK	4b	–

Test 1a suceeds because both sentences share the same head verb. Test 1b fails because the path to the head verbs are different. Tests 2a and 2b succeed because both paths to the answer role (⇓*s* and ⇓*subj*) are present in the candidate sentence. (They point to "Google", which therefore becomes an answer candidate.) Tests 3a and 3b succeed because all other paths present in PropBank's example sentence (⇓*obj* and ⇓*mod*) are also present in the candidate sentence. Tests 4a and 4b however fail, because the fillers for the non-answer roles at the ends of the paths in the candidate sentence do not match the fillers in the question. (In this example there only is one such a filler, for role ARG1, with in the question has the surface structure “YouTube”, whereas in the candidate sentence it is “which.”)

Because tests 1a and 2a succeeded, this sentence is assigned an initial weight of 1. However, because only three other tests succeed as well, its final weight is 8. This rather low weight for a positive candidate sentence is due to the fact that we compared it against a dependency structure which it only partially matched. However, it might very well be the case that another of the annotated sentences shows a perfect fit. In such a case this comparison would result in a weight of 64. If these were the only two sentences that produce a weight of 1 or greater, the final weight for this answer candidate would be $8+64=72$. (See also Section 2.2.4, where this method of combining results is explained in more detail.)

3.6 Evaluation

3.6.1 Coverage and Methology

Before we evaluate algorithm performance in detail, it is necessary to a look at the coverage the resources provide. The first observations to make in this regard is that FrameNet, PropBank and VerbNet all are concerned with **verb** semantics, which for our purposes has one unfortunate consequence: None of the resources contain data about the verb *to be*. Yet, more than 35% of the questions in TREC’s test sets show the head verb *to be*. For all these questions none of the resources is of any help.[7] A general breakdown of coverage issues is given separately for FrameNet, PropBank and VerbNet in Tables 3.7, 3.8 and 3.9, respectively. In all three tables, the second row shows the number of factoid questions in the test set and row three lists how many of these are questions with the head verb *to be*. The following rows indicate further coverage problems. A question, for example, might have a head verb that is not listed in the resource’s dictionary. How often this occurred is stated in row four. Row five lists how often a verb, although listed in the dictionary, contains no annotated sentences. (This happens rather often in FrameNet, where verbs that are planned to be annotated in the future are already in the dictionary–but without any data. For VerbNet this never

[7] This is not strictly true for FrameNet, which includes other parts of speech beside verbs. For a question like “Who is the president of the United States?”, in FrameNet, we would have to look up the noun *president*. Unfortunately, the coverage for nouns is very sketchy, so that we choose to not implement a mechanism to allow questions having a noun as their main frame-bearer.

is the case; it contains no annotated sentences, but abstract frame structures instead.) Row six list how often a question's head verb was looked up and found, but could not be used because none of the verb's listed senses showed the same transitivity as the question's head verb.[8] Finally, the last two rows show for how many questions the resources provide no data, once in absolute numbers and once as a percentage of the numbers of questions in the test set.

FrameNet	2002	2003	2004	2005	2006	2002-2006
question no	500	413	230	362	403	1908
head is "be"	218	144	78	119	123	682 (35.7%)
no entry	28	24	12	30	36	130 (6.8%)
no sentences	25	30	27	34	29	145 (7.6%)
transitivity	10	13	5	10	15	53 (2.8%)
no data	281 (56.2%)	211 (51.1%)	122 (53.0%)	193 (53.3%)	203 (50.4%)	1010 (52.9%)

Table 3.7: Breakdown of the availability of data in FrameNet for the TREC test sets used.

PropBank	2002	2003	2004	2005	2006	2002-2006
question no	500	413	230	362	403	1908
head is "be"	218	144	78	119	123	682 (35.7%)
no entry	5	14	6	20	20	65 (3.4%)
no sentences	20	13	11	15	25	84 (4.4%)
transitivity	15	12	7	8	14	56 (2.9%)
no data	258 (51.6%)	183 (44.3%)	102 (44.3%)	162 (44.7%)	182 (45.2%)	887 (45.5%)

Table 3.8: Breakdown of the availability of data in PropBank for the TREC test sets used.

[8] This usually indicates that the entries contained in the resource exemplifies a different sense that verb in the question. Consider the following example (the first question from the 2002 question set): "In what country did the game of croquet originate?" Here "originate" is used in an intransitive way. Yet, FrameNet contains only annotations for the transitive verb, like "... the ancient Greeks who originated the word aristocracy..." In such a case, role assignment for the question fails, and thus the question cannot be processed.

VerbNet	2002	2003	2004	2005	2006	2002-2006
question no	500	413	230	362	403	1908
head is “be”	218	144	78	119	123	682 (35.7%)
no entry	53	60	30	53	54	250 (13.1%)
no sentences	-	-	-	-	-	-
transitivity	8	9	12	6	15	50 (2.6%)
no data	279 (55.8%)	213 (51.6%)	120 (52.2%)	178 (49.2%)	192 (47.6%)	982 (51.5%)

Table 3.9: Breakdown of the availability of data in VerbNet for the TREC test sets used.

These coverage figures are rather disappointing. Obviously, an algorithm based on resources that contain no data for roughly half of all questions, can–at best–perform with an accuracy of about 0.5. We therefore decided to conduct the evaluations on two different kinds of test sets: One is based on all factoid questions in the TREC test sets from 2002 to 2006 and one is based only on those factoid questions in these sets for which data in the resources exist. Note that this second kind of test set in fact is different for each of the three resources. The size of each individual test set can be deducted from Tables 3.7, 3.8 and 3.9, where the total number of factoid questions in each test set are given and the number of factoid questions for which no data is available. (The size of the test sets used is the total number of factoid questions minus the factoid questions for which no data exists.)

From 2004 on, TREC organized their test sets in series, each headed by a target e.g. “Franz Kafka” and a few questions about this target, e.g. “When was he born?” Obviously, such questions cannot be used on its own, but first needs to be combined with the target to e.g. “When was Franz Kafka born?” For the experiments described here we used a manually resolved test set (which would contain “When was Franz Kafka born?” instead of “When was he born?”) The reason for this is that resolving the questions is not always trivial for an automatic system, but it is a problem that is unrelated to the problem at hand. Furthermore, for a linguistically-motivated approach like the one proposed here, it is important to start with grammatically correct questions in the first place.

Another slight departure from TREC’s setup is that we did not exclude questions known to have no answer in the AQUAINT document collection (so-called NIL questions). The reason for this is that we did not use the AQUAINT corpus, but the web as the underlying text source in which the answers have to be found. We simply assume that for every question the web contains the (correct!) answer.

3.6.2 Performance Method One

The evaluation results for the first method described in Section 3.4 can be seen in Table 3.10 (based on complete TREC test sets) and Table 3.11 (based on partial test sets – i.e., sets that do not include questions for which the respective method does not provide any data).

	2002	2003	2004	2005	2006	02-06	question count
FrameNet	0.114	0.075	0.161	0.105	0.062	0.099	1908
FrameNet +	0.122	0.085	0.183	0.119	0.069	0.110	1908
FrameNet ++	0.126	0.090	0.191	0.122	0.072	0.114	1908
PropBank	0.132	0.097	0.183	0.135	0.082	0.121	1908
VerbNet	0.128	0.082	0.187	0.121	0.072	0.112	1908
combined	0.162	0.111	0.200	0.157	0.091	0.140	1908
automatic	0.156	0.089	0.165	0.138	0.072	0.122	1908
comb+auto	0.174	0.119	0.200	0.163	0.096	0.147	1908

Table 3.10: Evaluation results for method one on complete TREC test sets.

	2002	2003	2004	2005	2006	02-06	question count
FrameNet	0.260	0.153	0.346	0.225	0.125	0.209	898
FrameNet +	0.279	0.173	0.389	0.254	0.140	0.234	898
FrameNet ++	0.288	0.183	0.407	0.260	0.145	0.242	898
PropBank	0.273	0.174	0.328	0.245	0.149	0.225	1021
VerbNet	0.290	0.170	0.391	0.239	0.137	0.231	926
combined	0.315	0.183	0.333	0.265	0.155	0.243	1099
automatic	0.302	0.150	0.255	0.231	0.117	0.208	1117
comb+auto	0.337	0.195	0.309	0.273	0.158	0.250	1122

Table 3.11: Evaluation results for method one on partial TREC test sets, using only questions for which data is available in the concerned resource.

In these tables column one indicates the experimental setup used, columns two to six give results in top-1 accuracy for the different test sets individually, row seven gives numbers for all these test sets combined. In row eight the overall number of questions in the used evaluation sets (02-06) is given. As far as the rows are concerned, rows two, three and four show top-1 accuracy when only FrameNet is used. In Section 3.4.1 experiments that make use of FrameNet's inter-frame relations were presented. Row two lists the results we get when using only the question head verb for the reformulations (e.g. *purchase.v* for the question "Who purchased YouTube?"), for row three the other entries in the same frame were also used (*purchase.v*, *buy.v* and *purchase_((act)).n*) whereas row four gives results using all entries in all frames to which the question's frame is related via *Inheritance, Perspective_on* and *Using* relations, when using only those frames which show the same frame elements (which adds *sell.v* and *retail.v* amongst others). Row five and six give results for PropBank and VerbNet respectively. For the combined run presented in row seven the verb was looked up in all three resources simultaneously and all entries from all three resources were used.

Additionally, Tables 3.10 and 3.11 also report, in row eight, an experiment with a cautious technique for expanding coverage. Any head verb, we assumed displays the following three patterns:

intransitive:	[ARG0] VERB
transitive:	[ARG0] VERB [ARG1]
ditransitive:	[ARG0] VERB [ARG1] [ARG2]

During processing, we then determined whether the question used the head verb (if it was not "to be") in a standard intransitive, transitive or ditransitive way. If it did, and that pattern for the head verb was not contained in the resources, we temporarily added this abstract frame to the list of abstract frames the system used. This method rarely adds erroneous data, because the question shows that such a verb argument structure exists for the verb in question. Finally, row nine show the results we achieve when the results from all three resources are combined with the coverage expansion strategy.

In Table 3.11, which shows results for partial TREC test sets, these sets for the coverage expansion strategy consist of all questions the system could analyse as exemplifying one of the basic three transitivity patterns. (258 out of 500 questions for the 2002 test set, 247 out of 413 for 2003, 149 out of 230 for 2004, 216 out of 362 for 2005, 247 out of 403 for 2006.) When the coverage expansion strategy is combined with the runs based on FrameNet, PropBank and VerbNet, the partial test sets contain of the union of all questions that the system could analyse as exemplifying one of the basic three transitivity patterns and all questions that show a head verb that can be found in any of the resources.

As mentioned earlier, the partial test sets used for the experiments in Table 3.11 differ in size (see column eight). This is because the resources differ in coverage. The evaluation measure on which the numbers in the tables are based is accuracy, which is obtained by dividing the number of correct answer by the number of questions in the test set. The fact that the divisor in the formula is different in each case explains why it is possible that a) the overall observed improvements in Table 3.11 are smaller than in Table 3.10 and that b) the "comb+auto" value for the 2004 test set in Table 3.11 is worse than the "combined" value.

The mentioned strategy was initially designed as a way to automatically extend coverage of the resources. Yet it can also serve as a baseline. After all, if we can get the necessary syntactic information directly from the question and if this is sufficient to find answers, what do we need the resources for? Indeed in Table 3.10 (based on complete test sets), we see that, when considering questions from all test sets, the automatic strategy outperforms all three resources individually. This however is no longer the case when the resources are combined. The reason for this lies, again, in coverage. The baseline outperforms the individual resources because it provides data where the resources do not. Yet, it is remarkable that the combined resources outperform the baseline, even on complete test sets (where we see a 14.7% increase). The reason for this is that additional verb valences, different from those of the question cause the resources to pick up answers that the syntactic information about the verb's usage contained in the question alone cannot deliver. As a bottom line, we can note that the three resources combined perform better on the complete test set than the baseline–despite their obvious coverage problems.

Table 3.12 shows the results of sign tests, based on complete TREC test sets. Columns one and two give the names of the compared runs (the names are the same as in Table 3.10). The last column shows the obtained p-value. The sign test looks at those questions which received different judgments for the two runs under comparison and tests the null hypothesis that there are equal numbers of differences in both direction, i.e. that run 2 produces an equal amount of correct answers for questions that have been answered wrong in run 1 and wrong answers for questions previously answered correct in run 1. Small p values indicate that there is a high chance of observing more improvements than deteriorations. Using this test, we see that all the performed comparisons are statistically significant (all p-values are smaller than 0.05).

run 1	run 2	p-value
FrameNet	FrameNet +	$<$0.01
FrameNet	FrameNet ++	$<$0.01
FrameNet	PropBank	$<$0.01
FrameNet	VerbNet	$<$0.01
VerbNet	PropBank	0.02
automatic	combined	$<$0.01
automatic	comb+auto	$<$0.01

Table 3.12: Results of sign tests, performed on complete TREC test sets (Table 3.10).

Another, unrelated point worth mentioning, is that performance varies a lot across different TREC test sets. The difference between the 2002 and 2003 test sets is simply due to the fact that the 2003 set contains more difficult questions than the 2002 test set. From 2004 on TREC used question series. TREC seems to have compensated this complication by moving to simpler individual questions in 2004. After that difficulty was again gradually increased. Additionally to question series, in 2005 and 2006 TREC included so-called event questions. These have an event as target around which the questions revolve. An example for such a target would be “1999 North American International Auto Show” (series 104 from 2005), for which one question is “What auto won the North American Car of the Year award at the show?” The manually resolved question which we used for evaluation is “What auto won the North American Car of the Year award at the 1999 North American International Auto Show?” These questions are very difficult to handle with our method, because the rephrased, quoted search queries posed to a search engine become quite monstrous, e.g.:

```
"at the 1999 North American International Auto Show the
North American Car of the Year award was won by"
```

This, because of its length, returns no results. Such questions are the main reason for the bad results on the 2006 test sets. (The 2006 test set contains more such questions than the 2005 set).

3.6.3 Performance Method Two

Our second method described in Section 3.5, can only be used with FrameNet and PropBank, because VerbNet does not contain annotated example sentences, which this method is based on. Results are presented in Table 3.13 and 3.14, in a similar manner than the the results for the first method were presented earlier: Results are shown separately for FrameNet and PropBank and additionally results are presented when both methods are combined, by looking up verbs and their annotated sentences in both resources simultaneously.

	2002	2003	2004	2005	2006	2002-2006
FrameNet	0.028	0.034	0.035	0.039	0.020	0.030
PropBank	0.094	0.070	0.139	0.110	0.074	0.093
combined	0.100	0.082	0.148	0.122	0.079	0.102

Table 3.13: Evaluation results for method two on complete TREC test sets.

	2002	2003	2004	2005	2006	2002-2006
FrameNet	0.059	0.069	0.074	0.083	0.040	0.063
PropBank	0.194	0.126	0.250	0.200	0.136	0.174
combined	0.198	0.139	0.254	0.210	0.142	0.182

Table 3.14: Evaluation results for method two on partial TREC test sets, using only questions for which data is available in the concerned resource.

What is surprising is that for the second method PropBank out-performs FrameNet very considerably. This was not the case for the first method, where PropBank also performed better but by a much smaller margin. Analysis shows that there are three reasons for this:

1. PropBank's lexicon contains more entries.
2. PropBank provides many more example sentences for each entry.
3. FrameNet only sparsely annotates peripheral adjuncts, and so does not apply to When- or Where-questions, which are common question types in TREC evaluation sets.

While all these points also apply for the first method, they are of more importance for the second method, because here the syntax of the actual annotated sentences has a greater impact: In the first method abstract patterns are used as a mediator between the annotated sentences and potential answer sentences. This mediator approach also enables the first method to add slots for peripheral adjuncts needed to answer When- or Where-questions, even if not present or annotated in the example sentences, as it is the case for FrameNet.

3.6.4 Combined Performance

Tables 3.15 and 3.16 show the results obtained when both methods are combined. The general method of how this is done is described in Chapter 2.2.4 earlier in this thesis. (The weight of method 1 was set to 5, the weight of method 2 to 1.)

	2002	2003	2004	2005	2006	2002-2006
Method 1	0.174	0.119	0.200	0.163	0.096	0.147
Method 2	0.100	0.082	0.148	0.122	0.079	0.102
Combined	0.198	0.171	0.248	0.218	0.126	0.187

Table 3.15: Evaluation results on for both methods separately and when combined based on full TREC test sets.

	2002	2003	2004	2005	2006	2002-2006
Method 1	0.337	0.195	0.309	0.273	0.158	0.250
Method 2	0.198	0.139	0.254	0.210	0.142	0.182
Combined	0.383	0.282	0.382	0.366	0.206	0.318

Table 3.16: Evaluation results on for both methods separately and when combined based on partial TREC test sets, using only questions for which data is available.

Table 3.17 presents evaluation results of the best performing factoid systems in TREC from the years 2002 to 2006. These numbers are based on the same TEST sets which we used. Note that there are three differences in our experimental setup: 1) We use manually resolved question series, 2) We do not have NIL questions, 3) We do not require the system to find a supporting document in the AQUAINT corpus. Especially the first and third point puts our system at a slight advantage. Nevertheless, as can be

	2002	2003	2004	2005	2006
median	?	0.177	0.170	0.152	0.186
best system	0.830	0.700	0.770	0.713	0.578
3rd best	0.542	0.562	0.626	0.326	0.390
5th best	0.368	0.298	0.313	0.293	0.298
10th best	0.304	0.208	0.213	0.215	0.213

Table 3.17: TREC evaluation scores for the years 2002-2006 in accuracy. Note the slight differences in the experimental setup when comparing against the methods presented here.

seen in Table 3.16 we are able to achieve results that compare well with top performing systems in the corresponding TREC evaluations–when considering only questions for which data is available. Results are less convincing when complete TREC test sets are used. Still, for three out of four years for which data about TREC system located at the median of all participants is available, our method outperforms the median.

3.6.5 Performance Impact on a pre-existing QA System

Tables 3.18 and 3.19 show how the two methods based on lexical resources improve performance of a pre-existing QA system. We used our own QuALiM system as described in Chapter 2. All three of QuALiM's algorithms were used; their combined performance is given in the first row of the tables. Note that QuALiM in itself performs well (as proven by TREC evaluations in 2004-2006, see Section 2.3). Furthermore, as mentioned QuALiM already combines three different methods. Thus, it represents a very strong baseline. Table 3.18 is based on complete TREC test sets. Table 3.19 is based on only those questions in the test set for which data was available in one of the three resources or for which the automatic coverage extension method could be applied.

Tables 3.18 and 3.19 report a 16.3% and 21.9% improvment in performance (measured in top-1 accuracy, for all TREC test sets combined, based on complete and partial TREC test sets respectively) compared to the strong baseline system.
To sum up the evaluation results so far:

- Our methods based on the resources, despite coverage problems, outperform the automatic baseline strategy.

	2002	2003	2004	2005	2006	2002-2006
QuALiM	0.464	0.341	0.409	0.354	0.325	0.380
+comb+auto	0.532	0.400	0.478	0.422	0.372	0.442

Table 3.18: Evaluation results of the QuALiM system on its own and when combined with both methods based on FrameNet, PropBank and VerbNet when also using the automatic coverage expansion method. Top-1 accuracy based on complete TREC test sets.

	2002	2003	2004	2005	2006	2002-2006
QuALiM	0.494	0.378	0.456	0.412	0.348	0.416
+comb+auto	0.616	0.446	0.557	0.518	0.413	0.507

Table 3.19: Evaluation results of the QuALiM system on its own and when combined with both methods based on FrameNet, PropBank and VerbNet when also using the automatic coverage expansion method. Accuracy based on partial TREC test sets.

- Our results compare well with the best performing systems at TREC (if using partial test sets).

- Our methods improve performance of a strong baseline system.

This leads to the following conclusion: Resources like FrameNet, PropBank and VerbNet can be highly beneficial for Question Answering (and potentially other areas of Natural Language Processing, where detecting paraphrases is of equal importance). Yet, as of today, coverage remains problematic and more efforts have to be undertaken by the community to create more complete resources.

3.6.6 Porting the Approaches to a Local Corpus

The methods described in this chapter have been developed for web based Question Answering. It has previously been observed that the web's massive size leads to much redundancy: Many facts are available not once but a thousand or more times. [Clarke et al., 2001, Kwok et al., 2001, Dumais et al., 2002] The first of the two methods described exploits this observation in that it creates very precise search queries that require the fact to be formulated in certain ways. If a known surface structure, associated with a question, is found, chances are high that it contains the answer to the question. But we only can expect to find a fact in exactly these known surface realisations if the corpus is big enough.

Generally in QA, when porting a method from the web to a local corpus, a performance loss has to be expected. Intuition suggests that for the first method described here, this loss should be considerable. In order to find out how much performance decreases we evaluated both methods on the AQUAINT corpus [Graff, 2002]. (This is a natural choice since the TREC question sets used were initially designed to be used with this corpus.) The complete corpus has been indexed with Lucene, which replaces the web search engine (Google) in our setup, which otherwise was completely identical to what is described in Sections 3.4 and 3.5. The results can be seen in Table 3.20.

Method 1	2002	2003	2004	2005	2006	2002-2006
FrameNet	0.026	0.019	0.043	0.028	0.020	0.026
FrameNet +	0.028	0.022	0.043	0.033	0.025	0.029
FrameNet ++	0.028	0.024	0.048	0.033	0.027	0.030
PropBank	0.032	0.024	0.035	0.025	0.025	0.028
VerbNet	0.024	0.022	0.039	0.030	0.025	0.027
combined	0.036	0.029	0.043	0.030	0.027	0.032
automatic	0.030	0.019	0.048	0.033	0.022	0.029
comb+auto	0.038	0.031	0.048	0.036	0.027	0.036

Method 2	2002	2003	2004	2005	2006	2002-2006
FrameNet	0.012	0.012	0.021	0.017	0.015	0.015
PropBank	0.048	0.029	0.065	0.049	0.037	0.044
combined	0.054	0.034	0.070	0.055	0.040	0.049

Method 1&2	2002	2003	2004	2005	2006	2002-2006
combined	0.064	0.046	0.087	0.069	0.050	0.061

Table 3.20: Evaluation results for both methods on complete TREC test sets and with the AQUAINT corpus as underlying document collection.

Obviously, these results are bad. One reason for this has already been mentioned: The AQUAINT corpus is much smaller in size than the web, thus redundancy is not very large. Therefore, the likelihood that there will be even one paraphrase that can be taken advantage of is much smaller. But there are other reasons as well, which we will address in the next section of this chapter.

3.7 Discussion

To sum up what has been shown so far:

- In our experiments, the use of FrameNet, PropBank and VerbNet in a web-based QA setting increases system performance considerably when comparing it against a strong baseline system.
- When evaluating with a test set containing only questions for which data is available, results are achieved that compare well against top TREC systems by using the two described methods which are based on the resources alone.
- FrameNet, PropBank and VerbNet all have major coverage issues, which affect the performance of the proposed algorithms. In fact, it has to be expected that these coverage issues affect any conceivable algorithm making use of them.
- Porting the web-based methods to a local newswire corpus leads to poor performance.

Overall, we are pleased with the methods' performance in a web-based setting (for which they were developed), especially when considering that data is available only for a subset of questions. We were able to show that the resources in theory and practice can be beneficial for Question Answering.

Yet, there are limitations to what these resources can achieve. We believe a lot can be learned from the fact that the porting of the two approaches to the AQUAINT corpus failed. Three important distinctions play a role here on which we will expand in the following. They are:

1. Direct vs. Indirect Evidence
2. Small vs. Large Corpora
3. Foreground vs. Background Information

Direct vs. Indirect Evidence

FrameNet, PropBank and VerbNet are all organized around the notion of (verb) semantics. These resources document the usage of different words (verbs for the most part) by exemplifying their range of semantic and syntactic combinatory possibilities (valences). Therefore, these resources can help to identify and recognize answer sentences that provide *direct evidence* for the question asked. This means they are beneficial in cases whenever there is a clear and obvious semantic relation between the question and its answer sentence.

Yet, often the evidence found in support of an answer is less direct. Consider the following eight sentences from the AQUAINT corpus, each of which would enable a person to answer the question "When was Alaska purchased?" [9]

1, NYT19981129.0133 "The islands were sold to the United States **in 1867 with the purchase of Alaska**."

2, APW20000812.0059 "As travelers pass through Auburn, they can stop at the homes of Harriet Tubman, who became a national heroine for her pivotal role in leading slaves to freedom through the Underground Railroad, and former Secretary of State William H. Seward, who engineered **the purchase of Alaska from Russia in 1867**."

3, NYT20000824.0333, "In Seward, the town named for Secretary of State William Seward, who **bought Alaska for $7.2 million in 1867**, a multimillion-dollar industry has developed around ships that take visitors to the bird rookeries and glaciers of Kenai Fjords National Park."

4, APW20000807.0053, "As travelers pass through Auburn, they can stop at the homes of Harriet Tubman, who became a national heroine for her pivotal role in leading slaves to freedom through the Underground Railroad, and former Secretary of State William H. Seward, who engineered **the purchase of Alaska from Russia in 1867**."

5, APW19990329.0045, "**In 1867**, U.S. Secretary of State William H. Seward reached agreement with Russia **to purchase the territory of Alaska** for $7.2 million, a deal roundly ridiculed as Seward's Folly."

6, NYT19980915.0275, "But **by 1867**, when Secretary of State William H. Seward negotiated **the purchase of Alaska** from the Russians, sweetheart deals like that weren't available anymore."

7, APW19980907.1163, "**Alaska**'s economy has been based on its vast wealth of natural resources since the United States **bought the territory from Russia in 1867**."

[9] Sentences 1 to 6 were judged as supportive by TREC. Sentences 7 & 8 were additionally identified by [Lin and Katz, 2005].

8, APW20000329.0213, "**On March 30, 1867**, U.S. Secretary of State William H. Seward reached agreement with Russia **to purchase the territory of Alaska** for $7.2 million, a deal roundly ridiculed as Seward's Folly."

Upon closer inspection it turns out that many of the above sentences cannot easily be analyzed in terms of frame semantics and thus would allow straightforward matching to the question. The formula for the first sentence for example would be:

```
sell.V(Buyer="United States",  Goods="the Islands")
```

It is not immediately obvious what should happen with the modifying PPs at the end of the sentence "in 1867" and "with the purchase of Alaska". Clearly, "in 1867" modifies the main clause (and thus the selling of the islands to the United States) and not the attached PP(s) "with the purchase of Alaska". Humans, when reading the sentence, would probably reason that, if the islands were sold with the purchase of Alaska, then probably both events happened at the same time and thus accept this sentence as a valid answer sentence. But this is not explicitly stated and situations are imaginable where the above sentence is logically true, but it is not the fact that Alaska was purchased in 1867. (The purchase of Alaska might, for example, have been a process spanning over several years with different territories being passed over at different times.)

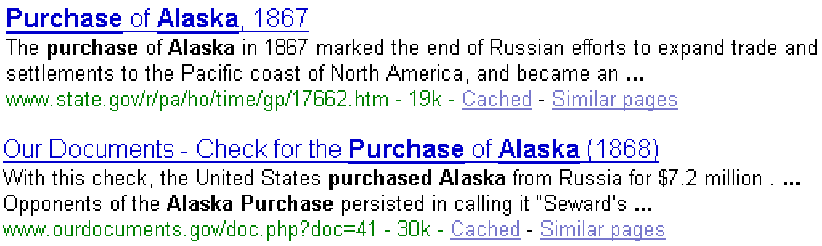
Sentences 5 and 6 further illustrate the problem at hand: Here, the year is given in which agreement was reached to purchase Alaska and in which the Alaska purchase was negotiated, respectively. Both dates need not necessarily be identical with the date when Alaska was finally sold. Yet, both sentences were judged as supportive by TREC (a decision that presumably seems reasonable to most people), and therefore a QA system which would discard these sentences would suffer a loss in performance.

What we are dealing here with are answer sentences that contain *indirect evidence* that answers the question. The bottom line of this observation is at first glance a contradictory one: A QA system, in order to successfully identify answers in a document collection, must be able to accept answers that–from a strictly logical point of view–do not answer the question. Thus an approach based on semantically motivated resources like FrameNet, PropBank and VerbNet, which were created to recognize strict semantic similarity of different surface structure, is not sufficient to identify answer sentences containing such forms of *indirect evidence*.

Small vs. Large Corpora

The huge size of the web and the–in comparison–small size of the AQUAINT corpus make a large difference for our methods. (This effect is usually called redundancy, see [Clarke et al., 2001] and [Dumais et al., 2002].) It is the amount of textual data available on the web that makes our strategy which uses precise, quoted queries not only feasible but also preferable over less precise key-word based querying. The fact that Alaska was purchased in 1867 can probably be found on (at least) several thousand different web pages. The query `"Alaska was purchased in"` on Google returns at the time of writing 463 hits with seven out of ten hits on the first result page listing “1867” directly after the search phrase in the result snippets. (`"Alaska was purchased in 1867"` returns 316 hits, indicating that overall 68% (316/463=0.68) of all results contain the correct answer at the predicted position. Almost all sentences that do not contain “1867” after “Alaska was purchased in”, do also not contain a series of digits directly after it, and can therefore be ruled out as possible answers. Thus employing this strategy delivers 316 pieces of evidence pointing towards the correct answer, while maybe bringing up only one or two that support a false answer.)

For a looser query like `Alaska purchased` (no quotes) Google approximates the number of hits to 4,790,000. On the first result page the number “1867” can be found five times. But we also find the number “1868”. Let’s look at the following two results:



Purchase of **Alaska**, 1867
The **purchase** of **Alaska** in 1867 marked the end of Russian efforts to expand trade and settlements to the Pacific coast of North America, and became an **...**
www.state.gov/r/pa/ho/time/gp/17662.htm - 19k - Cached - Similar pages

Our Documents - Check for the **Purchase** of **Alaska** (1868)
With this check, the United States **purchased Alaska** from Russia for $7.2 million . **...**
Opponents of the **Alaska Purchase** persisted in calling it "Seward's **...**
www.ourdocuments.gov/doc.php?doc=41 - 30k - Cached - Similar pages

The smallest substrings in these two snippets that contain morphological forms of the keywords and a number that could possible be a year are:

1. `Purchase of Alaska, 1867`
2. `purchase of Alaska in 1867`
3. `Purchase of Alaska (1868)`

Standard IR methods based on word overlap would have no chance of recognizing that the first two items are proper, correct answers, while the third is not. More sophisticated linguistically motivated methods might be able to figure out that the topic of the text in the second snippet is the *check that was used to pay for Alaska*, not the *purchase* *of Alaska itself*. But even that is doubtful. This example illustrates that it can be wise to search for potential answer sentence formulations directly with precise, quoted search queries, because in the case of a very large corpus (i.e. the web) one often can expect to find the desired fact in one particular surface form. Unfortunately, this strategy is far less successful when the corpus size is (much) smaller. Here, one cannot necessarily expect to find one particular surface realization of a fact. In the eight answer sentences listed earlier (and therefore probably in the complete AQUAINT corpus as well) not a single occurrence of the sentence or sentence fragment “Alaska was purchased in 1867” can be found (compare this to the mentioned 316 hits on Google).

Foreground vs. Background Information

Beside corpus size, there is another possible reason for the fact that methods based on FrameNet are less effective on newspaper corpora like AQUAINT. This has to do with the notions of foreground and background information in texts. In the foreground the most important information can be found, while the rest of the sentence is background [Huddleston and Pullum, 2002]. A standard example that illustrates this distinction in syntax is the it-cleft sentence.

Standard sentence: *I bought a red sweater.*

it-cleft sentence: *It was a red sweater that I bought.*

In the it-cleft sentence, backgrounded is the fact that the person denoted with “I” bought something, whereas foregrounded is the fact of it being a red sweater.

Comparing the content of the articles in the AQUAINT corpus with information requested by many factoid questions–at least in TREC test sets– reveals a discrepancy: Newspaper articles in AQUAINT naturally are, for the most part, about news–events that have just happened or just come to light. TREC questions however usually are not: They are of a much more general nature. As a result, most of the answers to TREC questions are not found in articles that are primarily about the topic introduced in the question; instead they are found in articles about recent events that show some (often minor) connection to the sought-after fact. In such articles this searched-for fact is often mentioned in the background.

The point just made can be illustrated with the first of the eight answer sentences listed above which comes from AQUAINT document with id NYT19981129.0133. Its headline is “ALEUT SACRED OBJECTS TO BE AUCTIONED AT SOTHEBY’S

DESPITE PROTESTS”. This sentence clearly shows some connection to Alaska (the Aleuts being the indigenous people of the Aleutian Islands of Alaska), but the article is not explicitly about the Alaska purchase. Thus indeed, as expected, the answer sentence “The islands were sold to the United States in 1867 with the purchase of Alaska.” shows the answer in the background.

All this is important because at least in FrameNet the example sentences were explicitly chosen by its creators to illustrate the usage of one particular verb and its arguments, which thus mostly occurs as the main verb of the sentence, i.e. in the foreground. (PropBank annotates continuous text, so this does not apply here.) Thus there is a disparity between the nature of the data the system learned from and the data the system deals with when finding answers: their syntactic structures are often of a different nature. Note that this is not necessarily true in the case of the web. Here, the nature of texts we are dealing with is very diverse. Of course there are many news stories on the web, but there are also many pages of other genres about a vast variety of topics. (A search on Google seems to indicate that there are probably more than one hundred pages explicitly about the subject of the Alaska purchase, many of them being headline entries, e.g. on Wikipedia.)

So far in this section, three distinctions were discussed which seem important to explain why our methods based on lexical resources work well on the web but not on a local corpus. In the following difficulties in trying to quantify the extent to which each is a factor in the observed experimental outcome are discussed.

As far as the redundancy effect, which manifests itself due to the different sizes of the web and the local corpus we use, is concerned, we would ideally like to count for all test questions (or a subset thereof) how many instances of valid answer facts are present in the corpus and on the web. This, as of today, is simply impossible for automatic methods. After all, what we would need to achieve this automatically is nothing less than a perfect QA system. Yet, there have been manual attempts, most notably [Lin and Katz, 2005], where the authors attempt to identify all documents in the AQUAINT corpus that answer 110 test questions from TREC 2002’s question set. The authors report an average of approximately 17 supportive documents per question. But even this manual method, although possible with small local corpora, reaches its limits when it comes to the web. We cannot possibly identify all answer instances to a set of questions on the web, especially when considering that the answer sentences will show very different surface structures. What we can do, and already have done earlier in this chapter, is to give some figures that can help to at least get an idea of the magnitude of the numbers we are dealing with. For the question “When was Alaska purchased?” we (based on [Lin and Katz, 2005]) identified eight answer instances in the AQUAINT corpus. At the time of writing the query `"Alaska was purchased in 1867"` returns 316 hits on Google (that is almost 40 times higher) and this of course is the sought answer fact formulated in only one of many possible surface forms. (Note also that the mentioned quoted query returns zero hits when used on the AQAUINT corpus–all answer facts present here have a different surface form.) This seems to strongly indicate that redundancy is important to explain why our methods work much better on the web than on a local corpus, but it of course helps little to quantify the problem at hand.

A different picture emerges when considering the distinctions direct/indirect evidence and foreground/background information. Possibly the best way to quantify their contributions would be to have human judges look at all (or possibly only a subset) of valid answer sentences and determine for each one whether we are dealing with some form of direct or indirect evidence or whether the answer is found in the foreground or background. However, experiments involving human judges are time consuming and expensive, especially if the judges are required to have some special knowledge and/or training, as would be the case here, where an understanding of the linguistic principles involved is necessary. On top of that, a further complication arises due to the fact that both distinctions cannot necessarily be expected to be completely independent from each other.

Bringing the notion about direct vs. indirect evidence together with observations about small vs. large corpora (and also the foreground vs. background distinction), we can conclude that a method for QA which is based on finding direct evidence is

- feasible for QA based on large corpora–especially the web–because one usually can expect to find the answers formulated in many different ways and if this is the case, it seems wise to go for the formulations that a) can be understood by the system and b) from a strictly logical point of view undoubtfully answer the question.

- less effective for QA based on smaller text collections, because only a few answer sentences might be present in the text and thus their surface structure cannot be easily predicted.

Chapter 4

A Corpus of *Question Answer Sentence Pairs* (QASPs)

4.1 Introduction

In the last chapter an approach to web-based Question Answering was described that is based on three resources–FrameNet, PropBank and VerbNet–that provide information about the relation between predicates and their arguments: semantic roles. It produced good evaluation results in a web-based setting (for which it was developed), but when porting it to a local corpus (the AQUAINT corpus) performance dropped considerably. We argued that one reason for this is that these resources provide information that helps to identify candidate sentences that provide *direct evidence*, that is sentences which answer the question in a strict logical sense. This is a suitable strategy in the case of the web, where the number of answer sentences for a factoid question can expected to be high and where it is therefore desirable to identify those answer sentences that undoubtfully answer the question. However, the situation is different when working with a small, local corpus. In that case we can expect far less valid answer sentences and therefore we cannot afford to be picky: We need to be able to utilize the answer sentences we find, even if they only provide *indirect evidence* for the answer. Indeed, manual inspection suggested that many answer sentences (judged as supportive by human assessors) in the AQUAINT corpus–from a strictly logical point of view–in fact do not answer the question at all.

In this chapter we are concerned with the creation of a research resource for Question Answering consisting of a large number Question Answer Sentence Pairs (QASPs), as found in the AQUAINT corpus. For all factoid TREC questions from 2002 to 2006 (for which TREC identified minimum one supporting document), we attempt to identify the sentences in that corpus that answer the question. For most questions the corpus contains more than one answer sentence. Furthermore, many factoid TREC questions follow simple syntactic pattens (e.g. "When was PERSON born?") and therefore can be generalized into question classes. Taking this into account, the corpus contains a considerable number of different answer sentences for many question classes. It thus can be seen as a resource containing paraphrases of answer facts: It constitutes a resource especially targeted towards paraphrasing research in Question Answering. The QASP corpus can be useful in many ways:

1. It enables researchers in the field to examine the resource and gain a better understanding about what precisely is required to build systems that automatically identify these sentences.

2. It can be used to automatically characterize the links between questions and answer sentences. This can be done on different levels, e.g. morphologically, syntactically or semantically.

3. It can be used as training data for various QA algorithms.

For this thesis, the QASP corpus is important because it contains many answer sentences that show some form of *indirect evidence* that answer the question. The only way to give an exact number or percentage of the answer sentences in the corpus showing indirect evidence would be to manually annotate each sentence. When doing this, ideally each sentence should be examined using more than one judge and the judges would need to receive a considerable amount of training in order to be able to distinguish between sentences exemplyfing direct and indirect evidence. We could not follow this approach due to time and money constraints. Instead, after we describe how the corpus was created, we provide an numerical analysis of some of its properties, obtained by automated methods in order to characterize the relations between questions and answer sentences in the corpus.

In the chapter 5 of this thesis the QASP corpus will be used as training data for a QA algorithm. This algorithm takes lessons learned from the experiments described in Chapter 3 and from the analysis provided in Section 4.4 in this chapter into account. It acquires syntactic and semantic knowledge from the answer sentences in the QASP corpus by analyzing the dependency relations between the answer and question constituents in the answer sentence. Because it learns from the QASP corpus, it is able to deal with direct and indirect forms of evidence, enabling it to recognize wider forms of paraphrases. We will evaluate this approach on the AQUAINT corpus and show that we achieve better performance than with the approach based on lexical resources.

The remainder of this chapter is organized as follows:

Section 4.2 "Background" provides information about resources and services used to create the QASP corpus, namely data sets released by TREC and Amazon's Mechanical Turk.

Section 4.3 "Creation of the Corpus" describes how the QASP corpus was created using Amazon's Mechanical Turk.

Section 4.4 "An Analysis of the Corpus" provides an analysis about a few key features of the corpus.

Section 4.5 "Conclusions" sums up this chapter.

Note that some of the work in chapter, especially Section 4.3, has been published as [Kaisser and Lowe, 2008].

4.2 Background

4.2.1 TREC data sets

One of the mayor providers of various data collections with the aim to foster research in Question Answering has been the Question Answering track of the Text REtrieval Conference (TREC), organized by the National Institute of Technology (NIST) each year since 1999 (see, for example, [Voorhees and Dang, 2005]). [1] TREC organized, oversaw or was involved the creation of the following resources: [2]

- Various sets of test questions, released on a yearly basis.
- Several corpora consisting of documents in which the answers to test questions are supposed to be found, e.g. the AQUAINT Corpus of English News Text [Graff, 2002].

[1] In 2008, TREC's QA track moved to the newly created Text Analysis Conference (TAC).

[2] This data, with the exception of the AQUAINT corpus, can be found here: http://trec.nist.gov/data/qamain.html

- Lists of top ranked documents as determined by an IR engine in the corpora for each question in the test sets.
- Judgment files indicating correct and incorrect answers returned by the systems.

This data has been invaluable for the QA community and it is hard to imagine the field without TREC and the wealth of data it has created over the years. Yet, most of the data created by TREC is better suited for research into IR than for research into NLP. For example, the smallest entities in the AQUAINT corpus marked by unique identifiers are documents. Similarly, the judgment files released by TREC each year list for each question, document-id/answer pairs. No data is available on sentence or paragraph level. The reason for this is that TREC (short for, as already mentioned, T*ext REtrieval Conference*, the name is no coincidence) sees QA mostly from an IR perspective: The traditional IR-inspired approach to QA starts with a document retrieval step, that is concerned with the identification of a set of documents that are likely to contain the answer. For such systems the resources provided indeed prove very useful.

From an NLP point of view, however, it seems appropriate to shift the focus from documents to smaller entities of text, especially sentences. A factoid question, usually, is not answered by a complete document, but instead by a single phrase. The sentence containing that answer phrase commonly provides the evidence to identify it as a valid answer sentence. [3] Thus a linguistically inspired QA system almost certainly, at some point in its processing pipeline, will match the question to a set of potential answer sentences by either syntactic or semantic means. In order to assist the development of such systems, a resource providing a large set of answer sentences for a large set of questions would be very useful–in somewhat similar ways than TREC data is for IR-inspired research into QA. For this reason the QASP corpus was created.

There is one other resource which we used beside TREC's test data, which is itself based on TREC data: [Lin and Katz, 2005]. Here, the authors describe how they manually created a small, reusable question answering test collection for research purposes. This collection is made up from 110 questions from the TREC 2002 test set for which the authors essentially try to identify ***all*** supportive documents in the AQUAINT corpus. To create the resource, they used known answers to TREC questions provided by NIST's judgment files, and manually crafted queries consisting of terms selected from

[3] We will elaborate on and diversify this claim is section 4.4.5

each question and its answers which they believed supportive documents are likely to contain. They then used these queries as an input for an IR system (Lucene) to retrieve all documents containing these terms. Each of these documents was then manually examined and those were marked that were considered as indeed being supportive for the answer to the question. They note, that while it is possible that this method still fails to retrieve all relevant documents, it can be assumed that the resulting set of judgments is much more complete than the presently available resources.

4.2.2 Mechanical Turk

We employed Amazon's Mechanical Turk (MTurk) [4] in order to create the QASP corpus. MTurk is a platform designed to enable computer programs to make use of human (as opposed to artificial) intelligence to perform tasks which computers are still, despite recent progress in AI, unable to perform. Amazon advertises MTurk as "Artificial Artificial Intelligence". From an alternative, somewhat simpler point of view, MTurk provides a platform for online experiments.

On Mechanical Turk, requesters (MTurk lingo for investigators) are able to define Human Intelligence Tasks (HITs) and upload them to the MTurk marketplace. There, workers (sometimes also referred to as turkers, both MTurk lingo for subjects) can browse among existing tasks, select one task and complete it for a (usually small) monetary reward set by the requester. Requesters can decide that workers have to fulfill certain qualifications before working on a task; they even are able define their own qualifications and test whether turkers meet them by creating tests. After a worker has completed one or more HITs, requesters can choose to accept or reject the results. (No money is paid if results are rejected.) Every such decision is logged and the statistics for each worker are accessible to requesters, who hence have a way to assess the trustworthiness of workers.

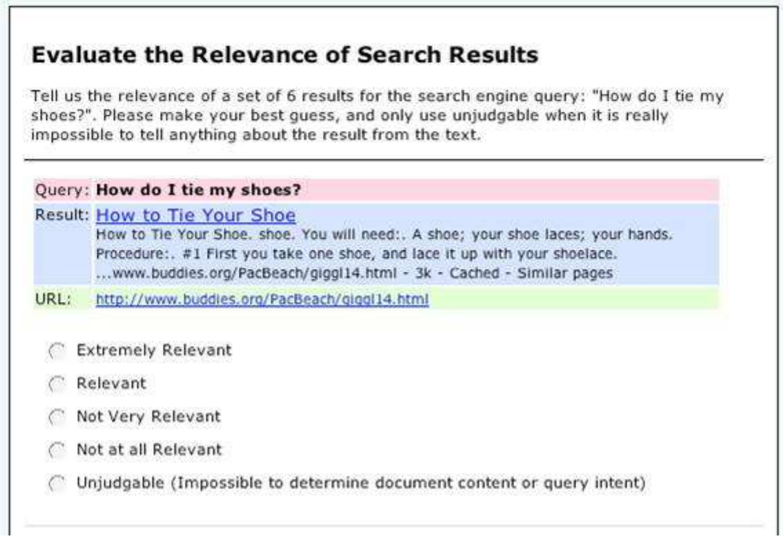
The HITs on MTurk can display a wide range of content (e.g. text and graphics) and provide many different input options, e.g. radio buttons, check boxes or input fields for free text. There also exists the possibility to create HITs based on Adobe Flash. Common tasks found on MTurk include but are not limited to:

- Please mark every face on this photo.
- Is the review of this product positive or negative?

[4] https://www.mturk.com/mturk/

- Do any of these photos contain illegal or offensive content?

Figure 4.1 gives one example of the many HITs that can be found on MTurk. Here the worker is asked to judge whether a web search result is relevant for a query.



Evaluate the Relevance of Search Results

Tell us the relevance of a set of 6 results for the search engine query: "How do I tie my shoes?". Please make your best guess, and only use unjudgable when it is really impossible to tell anything about the result from the text.

Query: **How do I tie my shoes?**

Result: How to Tie Your Shoe
How to Tie Your Shoe. shoe. You will need:. A shoe; your shoe laces; your hands. Procedure:. #1 First you take one shoe, and lace it up with your shoelace. ...www.buddies.org/PacBeach/giggl14.html - 3k - Cached - Similar pages

URL: http://www.buddies.org/PacBeach/giggl14.html

○ Extremely Relevant
○ Relevant
○ Not Very Relevant
○ Not at all Relevant
○ Unjudgable (Impossible to determine document content or query intent)

Figure 4.1: Sceenshot of a (partial) HIT as seen on MTurk. Workers are asked to judge the relevance of a search result.

The common perception by MTurk's requesters is that it makes experiments feasible that a few years ago would have been very difficult to carry out. The main reason for this is experiments on MTurk take place on the web, where people from all around the world can participate. Subjects do not have to physically be at the location where the experiment takes place. Other than other websites for online experiments often ran and used by Universities (e.g. http://www.language-experiments.org/), MTurk is not restricted to a certain research area and offers an easy way for participants to collect their money. The big advantage for investigators is that MTurk provides a large pool of subjects, which (literally) are just waiting to complete the tasks. It is not uncommon to upload a batch of several thousand HITs (e.g. involving a set of multiple choice questions) and having them all done one hour later, for USD 0.01 each. Mechanical Turk is fast and it is cheap.

Naturally, this has raised some concerns about the quality of the service. Yet, we are aware of several studies using MTurk for various tasks in the field of Natural Language Processing that report that high quality results can be obtained. [Snow et al., 2008], for example, evaluates the use of MTurk for five different tasks: affect recognition, word similarity, recognizing textual entailment, event temporal ordering, and word sense disambiguation. In all these cases high agreement between Mechanical Turk non-expert annotations and existing gold standard labels provided by expert labelers are achieved. In the study of expert and non-expert agreement for the affect recognition task they find that on average four non-expert labels per item are needed to emulate expert-level label quality. Because of the price difference between a non-expert on MTurk and an expert labeler in the real world, they conclude that many large labeling tasks can be effectively designed and carried out with this method at a fraction of the usual expense.

This is in line with our own experiences. We already had used MTurk for a study on customizing summary lengths for web search results [Kaisser et al., 2008] before using it to create the QASP corpus. We observed that as long as the task is simple (e.g. multiple choice, or selecting a sentence from a given document), and the number of turkers working on the same HIT is high enough to sort out turkers not taking the task seriously by checking inter-annotator agreement, MTurk delivers fast and good results for very little money. It is necessary though to check the results in order to identify and exclude turkers that do not perform the task properly. Further strengthening the case for MTurk, [Tietze et al., 2009] describe a study that examines the effect of linguistic devices on recall and comprehension in information presentation using recall and eye-tracking data. The authors use MTurk to validate results gained in a lab-based setting and find that average recall rate is nearly identical for MTurk with 0.76, when comparing it to subjects performing the reading experiment in the lab (0.77). The average time it took participants to complete the test was 23 minutes (MTurk) and 26 minutes (lab-based) per participant. They mention that they had to exclude results from three out of 60 participants because they performed the task in less than half of the average time and answered less than 50% of the questions.

4.3 Creation of the Corpus

4.3.1 TREC data and the QASP corpus

As mentioned, TREC has been a major provider of various valuable data collections. When building the QASP corpus, we used one of these collections, more precisely TREC's annual judgment files, as a starting point. These judgment files list all responses from all participating QA systems in one year and how they were judged by NIST assessor.

In the QA track, for each question in a given set of questions, participants' systems are expected to return an *answer, document id*-pair. These answers must be found in a provided document collection, but external sources (e.g. the Web) can be used to locate the answer as well. The document collection used from 2002 to 2006 was the *The AQUAINT Corpus of English News Text* [Graff, 2002]. The judgment files which TREC releases consist of *question id, document id, answer, judgment* quadruples. One line in these files looks like this:

```
1395 NYT19990326.0303 1 Nicole Kidman
```

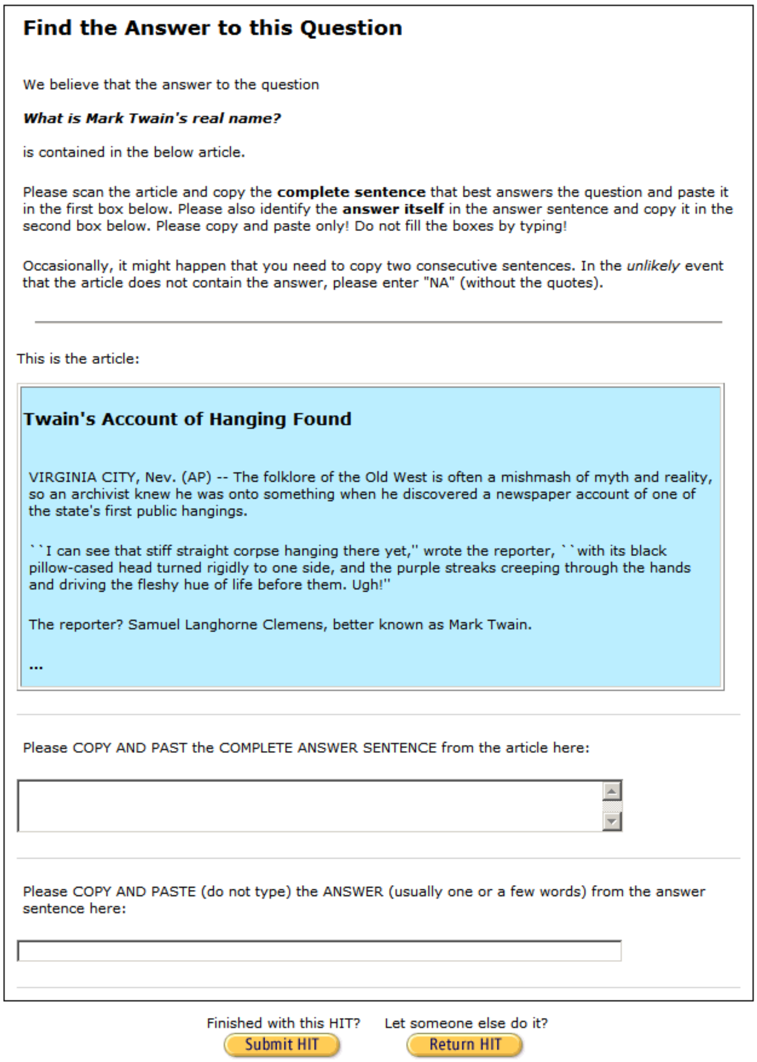
Here, Question 1395 (*Who is Tom Cruise married to?*) has been answered by one (undisclosed) participating system with the string "Nicole Kidman". `NYT19990326.0303` is the identifier of one particular document in the AQUAINT corpus (the 303rd document from the March 26, 1999 edition of the New York Times). The third column indicates whether the system returned the correct answer ("`1`", as in this case, means it did). This data has been used by researchers since then in a variety of ways; see for example [Echihabi et al., 2004] or [Monz, 2004].

But whenever researchers want to find the exact evidence for the answer provided, he or she has to look for it him/herself: no resource has been available that lists the *sentences* in these documents that provide evidence for the given answer. To address this gap, we collected the answer sentences for most *question id, document id, correct answer* triples for the years 2002 to 2006. There are 8,107 such triples in total that have been published by NIST during this period (counting only those that point to supporting documents). In addition, we identified most of the answer sentences for the *question id, document* pairs collected in [Lin and Katz, 2005]. As mentioned, in this paper the authors attempted to locate *every* document in the AQUAINT collection that contains the answer, whereas TREC publishes only incomplete lists based on the documents that the actual participating systems regarded as relevant.

4.3.2 Using MTurk to create the QASP corpus

As mentioned, we used Amazon's Mechanical to locate answer sentences to TREC questions in each of the AQUAINT documents judged relevant by NIST. Workers were asked to read a question and select from a displayed document the sentence that they thought answered it best. A screenshot of one of our HITs can be seen in Figure 4.2.

Every HIT was completed by three different turkers. This enables us to check inter-annotator agreement and thus have a measure for the plausibility of every collected answer sentence individually, as well as to evaluate the reliability of the complete collection. The actual execution of the experiment cost about USD 650 (Turkers received USD 0.02 for each completed HIT; 10% fees were paid to Amazon).



Find the Answer to this Question

We believe that the answer to the question

What is Mark Twain's real name?

is contained in the below article.

Please scan the article and copy the **complete sentence** that best answers the question and paste it in the first box below. Please also identify the **answer itself** in the answer sentence and copy it in the second box below. Please copy and paste only! Do not fill the boxes by typing!

Occasionally, it might happen that you need to copy two consecutive sentences. In the *unlikely* event that the article does not contain the answer, please enter "NA" (without the quotes).

This is the article:

Twain's Account of Hanging Found

VIRGINIA CITY, Nev. (AP) -- The folklore of the Old West is often a mishmash of myth and reality, so an archivist knew he was onto something when he discovered a newspaper account of one of the state's first public hangings.

``I can see that stiff straight corpse hanging there yet,'' wrote the reporter, ``with its black pillow-cased head turned rigidly to one side, and the purple streaks creeping through the hands and driving the fleshy hue of life before them. Ugh!''

The reporter? Samuel Langhorne Clemens, better known as Mark Twain.

...

Please COPY AND PAST the COMPLETE ANSWER SENTENCE from the article here:

Please COPY AND PASTE (do not type) the ANSWER (usually one or a few words) from the answer sentence here:

Figure 4.2: Example HIT, as shown to the subjects. (For this screenshot the text of the article was shortened from the original.)

Table 4.1 shows inter-annotator agreement when computing the similarity of responses by using strict string equality. One problem we encountered was that different browsers and/or operating systems use different copy-and-paste implementations. So even if two users intend to select exactly the same sentence, some implementations automatically include the closing punctuation mark while others do not. The same holds for opening/closing quotes and brackets. Table 4.2 shows inter-annotator agreement when allowing an Levenstein edit distance of 5, which treats sentences with minor differences as similar.

Three	3577	44.1%
Two	3248	40.1%
None	1282	15.8%

Table 4.1: Inter-annotator agreement for the 8107 TREC 2002-2006 QASPs when using strict string equality. The table shows how often all three turkers selected the same sentence (and the same answer), how often two turkers made the same selection, and how often none of the turkers agreed.

Three	4345	53.6%
Two	2907	35.9%
None	855	10.5%

Table 4.2: Inter-annotator agreement for the 8107 TREC 2002-2006 QASPs when allowing a Levenstein edit distance of 5.

There are several reasons why agreement is not higher, for example:

1. Turkers selected different sentences from a document which indeed includes more than one sentence that answers the question.
2. Sometimes it is not obvious to turkers where the selection boundaries should be.
3. Some selections made by turkers were suboptimal or simply wrong.

The second point can be illustrated with the example shown in Figure 4.2. We see in our data that, for the given text, two turkers selected the passage “The reporter? Samuel Langhorne Clemens, better known as Mark Twain.” while one selected the shorter “Samuel Langhorne Clemens, better known as Mark Twain.” In such a case there is no single correct answer, both possibilities seem justifiable. For these reasons, we consider the inter-annotator agreement reported in Table 4.2 as satisfactory. Nevertheless, in order to increase the quality of the final data, we decided to clean the data by hand.

4.3.3 Post Processing the Data

As noted before in the literature, the task to build a high quality research collection for QA, whether it contains documents, answer sentences or answers, is not always straightforward [Voorhees and Tice, 2000, Lin and Katz, 2005]. The most important issue here, beside the quantity of data involved, is that human judges tend to disagree about what constitutes a valid answer, answer sentence or supporting document.

In order to increase the quality of our data, we decided to let a second set of subjects check the results of the turkers. For this second round we did not employ MTurk. Instead, the subjects consisted of PhD students at the University of Edinburgh's School of Informatics. As a starting point the students received a file with all the judgments from round one, which included all sentences selected by the turkers. Each sentence was tagged to indicate how many turkers (one, two or three) had selected it. By default sentences which were tagged as *two* or *three* received an additional tag indicating that the sentence should become part of the final collection, whereas sentences selected by only one turker did not have this tag. The students' task then was to look at all sentences and add or remove the tag indicating that sentence should belong to the final selection if they thought that the turkers had made a mistake. Only one student looked at each sentence to make the final decision.

We used this opportunity to add tags to the QASP corpus. The following tags are included in the final version of the data set:

A indicates that the sentence does answer the question, but that the **a**nswer is inexact (e.g. only the last name of a person is mentioned).

C indicates that the sentence does answer the question, but that some important information is missing in the sentence. This information can most likely be found in the remainder of the document. (C stands for ***C****ontext missing*)

D indicates that it is **d**oubtful whether the sentence answers the question.

1 indicates that the sentence indeed does answer the question.

One sentence might be marked with more than one tag. Table 4.3.3 lists one example QASP for each tag.

Tag	Question	Answer Sentence
A	When did the shootings at Columbine happen?	The Columbine High School shootings April 20 also had an effect on ...
C	What is the capital of Kentucky?	The capital, Frankfort, is about 15 miles down river.
D	When was the internal combustion engine invented?	The first internal-combustion engine was built in 1867, but ...
1	How tall is Mount McKinley?	Together, they climbed Mount McKinley (20,320 feet), the highest peak in the United States.

Table 4.3: Examples to illustrate the tags used in the corpus: The first sentence gives only an inexact answer ("April 20" instead of "April 20, 1999"). The second sentence gives the correct answer, but does not mention "Kentucky". Most likely Kentucky is mentioned in a preceding sentence. Whether the third sentence answer the question is somewhat doubtful. The final sentence clearly answers the question.

4.3.4 Data Format

Our dataset comes in six files. Five files contain data based on TREC judgment files from 2002 to 2006. A sixth file is based on [Lin and Katz, 2005]. Each line in the files shows the data for one Question Answer Sentence Pair. The data in each line is comma separated. There are six rows in each line:

1. The TREC question id.
2. The AQUAINT document id.
3. The question itself (in quotes).[3]
4. The answer sentence (in quotes).
5. The answer (in quotes).
6. A tag (e.g. `1`) or possibly a list of tags, separated by semicolons (e.g. `A;C`), see Section 4.3.3.

The answer given in row five is always a substring of the answer sentence in row four. Note that the data in rows three, four and five may contain commas itself.

Figure 4.3 illustrates the data in our corpus. (Line breaks were added for better readability.)

[3] Here the data is slightly redundant, the question could of course be looked up in TREC's original question file, but we felt that including it increases human readability.

```
1396, NYT19981201.0229,"What is the name of the volcano that destroyed
the ancient city of Pompeii?", "Visiting tourists enter the excavated
ruins of the city - buried by the eruption of Mount Vesuvius - via a
tunnel through the defensive walls that surround it, just as visiting
traders did 2,000 years ago.", "Mount Vesuvius", C

1396, XIE19961004.0048,"What is the name of the volcano that destroyed
the ancient city of Pompeii?", "However, both sides made some gestures
of appeasement before Chirac set off for the Italian resort city lying
beside the Vesuve volcano which destroyed the Roman city of Pompeii.",
"Vesuve", 1

1396, NYT20000405.0216,"What is the name of the volcano that destroyed
the ancient city of Pompeii?", "His was a devout but somewhat empty
gesture, since Pompeii was pagan in A.D. 79, when Vesuvius erupted.",
"Vesuvius", D

1396, NYT19980607.0105,"What is the name of the volcano that destroyed
the ancient city of Pompeii?", "The ruins of Pompeii, the ancient city
wiped out in A.D. 79 by the eruption at Vesuvius, are Italy’s most
popular tourist attraction, visited by two million people a year.",
"Vesuvius", 1

1396, NYT20000912.0360,"What is the name of the volcano that destroyed
the ancient city of Pompeii?", "Ryan likened the discovery to finding
Pompeii, the ancient city buried by Mount Vesuvius.",
"Mount Vesuvius", 1
```

Figure 4.3: Five Question Answer Sentence Pairs, as contained in the corpus. (Line breaks were added for better readability.)

4.3.5 Numerical Overview

Table 4.4 presents a numeric overview over the original data sets and the data in our corpus. The first column shows the origin of the data, usually the year in which TREC released it. The next column shows the number of questions in the original data. Column three gives the numbers of supporting documents identified by TREC. Column four lists the number of questions for which we were able to find at least one answer sentence. This number is lower than the number of questions in the original data set for three reasons: a) There are so-called NIL questions in the question set, i.e. questions that do not have an answer in the document collection. b) For some non-NIL questions, TREC participants were unable to find the answer in the collection, although it exists. c) Our subjects were unable find a valid answer sentence in a document, judged as supportive in the original data set. The fifth column in the table shows how many sentences we could identify. There are three reasons why the number of sentences collected is lower than the number of document-ids in the original data set: a) The document itself might contain the answer, but no single text passage can be identified that answers the question. In such cases evidence from multiple passages would be needed to answer the question. b) Our subjects did not agree with TREC's judgment and decided that there is no answer in the document. c) There is a valid answer sentence in the document, but our subjects were unable to locate it. Finally, column six gives the average number of answer sentences we were able to identify for each question (i.e., column 4 divided by column 5).

year	No. factoid questions (original)	No. supporting documents identified	No. factoid questions remaining	No. question-answer sentence pairs	mean no. pairs per question
2002	500	2,177	429	2,006	4.67
2003	413	1,764	354	1,448	4.09
2004	231	919	204	865	4.24
2005	363	1,599	319	1,456	4.56
2006	404	1,648	352	1,405	3.99
2002-2006	1,911	8,107	1,658	7,180	4.33
2002 (Lin)	109	1,822	97	1,650	17.0

Table 4.4: Quantitative overview of the data contained in the QASP corpus.

Table 4.5 presents a numeric overview over tags used in the QASP corpus. For each subset of the data we list how many sentences are tagged as “1”, “A”, “C”, or “D”.

year	sentences	tagged “1”	“A”	“C”	“D”
2002	2,006	1,833 (91.4%)	44 (2.2%)	53 (2.6%)	76 (3.8%)
2003	1,448	1,352 (93.4%)	6 (0.4%)	17 (1.2%)	73 (5.0%)
2004	865	826 (95.5%)	8 (0.9%)	12 (1.4%)	19 (2.2%)
2005	1,456	1,228 (84.3%)	53 (3.6%)	152 (10.4%)	23 (1.6%)
2006	1,405	1,159 (82.5%)	53 (3.8%)	169 (12.0%)	24 (1.7%)
2002-2006	7,180	6,398 (89.1%)	164 (2.3%)	403 (5.6%)	215 (3.0%)
2002 (Lin)	1,650	1,128 (68.4%)	200 (12.1%)	202 (12.2%)	120 (7.3%)

Table 4.5: This table shows in the second column the total number of answer sentences for in the data set for each year. Subsequent columns show how many of these have been tagged “1”, “A”, “C” and “D”, respectively.

It still remains an open question whether the corpus is big enough to be used successfully in a QA system. (We will attempt to answer this question in Chapter 5.) By employing Amazon’s Mechanical Turk we chose a rather novel approach to create the QASP corpus. Of course there are other, more traditional ways to build such a corpus, some of which we potentially could use to extend its coverage. Bootstrapping possibly is the most established technique in this respect. Bootstrapping algorithms start with a small number of known seed instances (or patterns) which they use to iteratively discover more and more instances and patterns which express the same relation than the seed terms ([Agichtein and Gravano, 2000], [Ravichandran and Hovy, 2002]). When bootstrapping is applied to Question Answering the seed terms usually are question key words and the answer. The algorithm is supposed to find surface or syntactic patterns that are suitable to detect answers to yet unseen questions. We could have chosen to build a corpus of Question Answer pairs with bootstrapping methods, but such a fully automated procedure would likely have resulted in a less clean, more noisy corpus. By opting to use MTurk on the other hand we can expect that the use of humans to discover valid answer sentences leads to less unsupportive answer sentences in the corpus.

4.4 An Analysis of the Corpus

4.4.1 Why analyze the data?

The QASP corpus can serve several purposes. It can be used as input or training data for various kinds of QA algorithms. Beside that, when taking a closer look at the data ourselves and showing it to other researchers, we usually found that this was interesting in its own right. Even researchers working in QA for several years or decades were amazed at the sometimes complex relations between a question and its answer sentences. The conversation that arose usually have been along the lines of what methods one would need to link a particular answer sentence to its question, and what methods clearly would not be sufficient to do this.

Inspired by these conversation, a quantitive analysis of the relations between the questions and answer sentences in the corpus has been carried out, in order to assess how they relate to each other and how much common QA strategies can achieve. We also hope to find some valuable hints as to what methods would be suitable to locate these kind of answer sentences in the AQUAINT corpus. We were especially interested in the following questions, which we address in the remainder of this chapter:

Average Word Counts What is the average number of words of the questions, answer sentences and answers in the corpus?

Word Overlap How often do words from the question also occur in the answer sentence. Can different tendencies found for different words or classes of words?

Head Verbs How often is the head verb of the question also head verb of the answer sentence? If it is not, does the question's head verb occur in some variation somewhere in the answer sentence?

Role of Context What is the percentage of identified answer snippets that are in fact multi sentence constructs, i.e. where the answer is spread across multiple sentences?

4.4.2 Average Word Counts

This investigation is intended to determine the average number of words of the questions, answer sentences and answers in the QASP corpus. (A word here is defined as any detached character sequence separated by whitespace or punctuation marks in a string that is not a punctuation or whitespace character itself.) Results are shown in Table 4.6.

year	questions			answer sentences			answers	
	no.	av. len.	st. dev.	no.	av. len.	st. dev.	av. len.	st. dev.
2002	429	7.44	2.38	2006	28.53	12.70	1.78	1.14
2003	354	7.74	2.53	1448	29.18	15.49	1.78	1.12
2004	204	7.25	2.02	865	29.34	14.08	1.97	1.45
2005	318	8.67	3.08	1456	28.88	11.79	1.90	1.34
2006	352	9.42	3.14	1405	29.34	11.73	1.81	1.01
02-06	1657	8.14	2.81	7180	28.99	13.13	1.83	1.20
Lin	97	7.46	2.22	1650	30.36	12.80	1.75	0.93

Table 4.6: Average length (in words) of the questions, answer sentences and answers in the QASP corpus

The table lists the average length (in words) of the questions, answer sentences and answers in the QASP corpus. Columns two, three and four list the number of questions in one particular subset of the data, their average number of words and standard deviation. Columns five, six and seven give the same data for answer sentences. Columns eight and nine show results for answers; the number of answers in the corpus is omitted because it is the same as for answer sentences. As can be seen, for example, the average length of an answer sentence in the QASP corpus is close to 30 words. This is due to the fact that the AQUAINT corpus contains newspaper articles (for example from the New York Times) which usually show a rather complex sentence structure.

4.4.3 Word Overlap

Many traditional IR and QA methods are based on word overlap. In this experiment, we checked for every word in every question in our data whether it also occurred in the corresponding answer sentence. This test was performed in two variations:

1. We tested whether the word from the question occurs in the answer sentence in exactly the same way, i.e. in the same morphological form.

2. We tested whether the word occurs in the question in *some* morphological form.

For the first test all question and answer sentence words were transformed to lower case. After that strict string matching was applied. In the second case, we applied two different automatic ways to determine whether a word was the same. Firstly, we employed a stemmer based on the Snowball algorithm by Martin Porter (in the implementation that comes with Lucene's Sandbox, i.e. the `SnowballFilter` class [Porter, 2001], [Hatcher and Gospodnetić, 2004]). Secondly, we used the University of Pennsylvania's morphology database [Daniel et al., 1992] which contains 317,322 words in different morphological forms. The combination of both methods makes our stemming efforts fairly robust. While the morphology database contains no named entities, Snowball usually can deal with these. Snowball's weaknesses include irregular verbs–these however are contained in the morphology database.

Table 4.7 shows the results for the part of our data which is based on TREC judgment files. We only took answer sentences which are tagged as "1" into account. Stop words, question words and punctuation were removed. As can be seen, words starting with an upper case letter were evaluated separately from words starting in lower case. The numbers in the second column show on how many (non-stop) words the data is based. As there are 1550 questions in this part of the corpus with minimum one answer sentence judged as "1", it can be derived that each question has on average 4.34 non-stop words (6722/1550=4.34), 1.85 of which are upper case and 2.49 are lower case. It also can be seen that 72.6% of all upper case words in the questions can also be found in the corresponding answer sentence. For lower case words, this is only 39.5%. When stemming is performed, results increase only slightly for upper case words (from 72.6% to 74.2%, that is 1.6%), but more significantly for lower case words (from 39.5% to 48.1%, that is 8.6%). Overall, when looking at both upper and lower case words, 53.6% of all words in a question can also be found in the answer sentence; 59.2% when stemming is performed, which is 5.6% higher.

Case	**No.**	**No Stemming**	**Stemming**
Upper	2862	0.726	0.742
Lower	3860	0.395	0.481
Both	6722	0.536	0.592

Table 4.7: Numerical results of our analysis of question/answer sentence word overlap, based on TREC 2002-2006 data.

Table 4.8 shows the results for the part of our data based on [Lin and Katz, 2005]. Again, stop words, question words and punctuations were excluded. For this part of our data set, 64.9% of all upper case words in the questions could also be found in the corresponding answer sentence. For lower case words, this is only 40.5%. As with the data based on TREC data sets, when stemming is performed results increase slightly for upper case words (from 64.9% to 66.7%, that is 1.8%), but more significantly for lower case words (from 40.5% to 47.3%, that is 6.8%). Overall, when looking at both upper and lower case words, 49.3% of all words in a question can also be found in the answer sentence; 54.3% when stemming is performed which is 5.0% higher.

Case	**No.**	**No Stemming**	**Stemming**
Upper	132	0.649	0.667
Lower	236	0.405	0.473
Both	368	0.493	0.543

Table 4.8: Numerical results of our analysis of question/answer sentence word overlap, based Lin & Katz's data.

When comparing tables 4.7 and 4.8, we see that usually word overlap is smaller for the data based on [Lin and Katz, 2005] (an exception here are lower case words for which no stemming is performed). This makes sense when considering that Lin & Katz try to identify by hand *every* document in the AQUAINT collection that contains the answer. TREC data, on the other hand, is based on only those documents which automatic systems participating in TREC's QA track have successfully identified as containing the answer to the given question. As word overlap plays a large role in the document selection/answer finding strategies employed by many of these systems, we indeed should expect that documents/paragraphs/sentences with high word overlap occur more frequently in the data based on TREC's judgment files.

Table 4.9 provides a more detailed look at the data from Table 4.7–it gives figures for individual words. The first table shows lower case words, the second table lists upper case words, while the last table shows, for reasons of completeness, stop words. In each table, the first column shows in how many questions the word occurred, the next column lists the word, while the third and fourth columns list the fraction of answer sentences in which the word re-occurred. The two last columns differ in that column four gives the numbers when stemming is performed. It can be seen that, especially for lower case words, figures differ considerably for each word. An interesting contrast provide the verbs "die" and "born" in the first table. While "die", in this particular morphological form, only re-occurs in 4.6% of all cases in the question, this increases to 62.3% when stemming is performed. "born" on the other hand re-occurs in 70.5% of all cases, regardless of whether stemming is perfomed or not.

No.	Word	f	f (st.)
89	name	0.136	0.192
70	first	0.588	0.588
64	year	0.135	0.163
60	country	0.171	0.201
59	born	0.664	0.664
57	die	0.031	0.670
50	city	0.263	0.296
30	president	0.753	0.782
29	founded	0.500	0.705
25	located	0.059	0.063
23	date	0.000	0.011
22	world	0.728	0.728
22	made	0.211	0.256
20	old	0.295	0.295
19	play	0.151	0.325
19	company	0.382	0.435
17	people	0.648	0.648
16	won	0.368	0.428
15	died	0.734	0.734
15	held	0.167	0.167
15	long	0.100	0.100
15	called	0.102	0.102
15	win	0.093	0.725
15	movie	0.344	0.353
14	largest	0.676	0.676
14	occur	0.071	0.109
14	day	0.431	0.487
13	time	0.172	0.172
13	built	0.486	0.544
13	stand	0.060	0.157
12	invented	0.259	0.299
12	get	0.024	0.107
12	most	0.349	0.432
12	members	0.568	0.711
12	national	0.762	0.762
12	fast	0.114	0.190
12	population	0.822	0.822
12	real	0.294	0.294
12	famous	0.361	0.361
11	space	0.603	0.603
11	begin	0.000	0.583

No.	Word	f	f (st.)
36	U.S.	0.313	0.313
24	American	0.496	0.549
21	United	0.521	0.521
17	International	0.585	0.585
17	World	0.882	0.882
14	John	0.743	0.743
13	King	0.885	0.885
11	University	0.749	0.749
11	Paul	0.661	0.661
11	William	0.498	0.498
11	Great	0.835	0.835
10	River	0.891	0.991
10	New	0.671	0.711
10	Show	0.572	0.572
10	Baseball	0.271	0.271
10	Cup	0.913	0.913
9	President	0.737	0.785
9	Island	0.583	0.620
9	China	0.758	0.758
9	George	0.448	0.448
9	America	0.759	0.796
9	Miss	1.000	1.000
9	States	0.370	0.398

No.	Word	f	f (st.)
1027	the	0.902	0.902
743	what	0.019	0.019
493	is	0.272	0.663
395	of	0.688	0.688
387	was	0.342	0.722
336	in	0.667	0.667
277	how	0.007	0.007
265	did	0.024	0.028
218	when	0.122	0.122
167	who	0.161	0.161
126	many	0.038	0.038
122	where	0.056	0.056
90	does	0.003	0.048
90	a	0.602	0.602
86	for	0.451	0.454
85	to	0.601	0.601
69	are	0.211	0.740
57	on	0.523	0.523
43	from	0.502	0.502
40	which	0.085	0.085
39	and	0.719	0.719
38	at	0.399	0.399
30	have	0.295	0.472
29	were	0.211	0.639
27	an	0.187	0.187
21	that	0.243	0.243
20	there	0.173	0.173
20	do	0.000	0.000
19	by	0.368	0.368
19	much	0.039	0.039
18	has	0.275	0.430
17	with	0.263	0.263
16	his	0.564	0.564
15	as	0.246	0.246
13	its	0.334	0.461

Table 4.9: Word overlap for our data based on TREC 2002-2006 judgment files, when broken down for individual words, sorted by frequency of occurrence. The first table shows lower case words, the second table shows upper case words, while the last table shows, for reasons of completeness, stop words.

4.4.4 Head Verbs

Syntactic approaches to Question Answering often place high importance on the head words of questions and answer sentences because they largely determine the structure of the remaining sentence (see for example our own work on semantic roles in Chapter 3 or [Katz et al., 2002], [Wu et al., 2003] and [Novischi and Moldovan, 2006]). We thus performed an analysis based on questions' head words, concentrating on verbs, employing MiniPar for the necessary syntactic analyses. The results can be seen in Table 4.10. It lists numbers when all verbs are considered together ("Overall"), when sentences with the head "to be" are separated and additionally the 15 most frequent verbs in the part of our data which is based on TREC judgment files. Column 3 shows how often the questions's head verb also occurs as the head verb in the answer sentence. In addition we checked whether the question's head verb occurs somewhere in the answer sentence in some morphological form. The procedure used was the same as the one described in Section 4.4.3 "Word Overlap". (As a consequence, beside verbs, nouns and adjectives are also accepted.) The following QASPs illustrate two cases where the questions's head verb occurs in the answer sentence, but not as the head of the answer sentence:

Q: "Where was Bob Dylan **born**?"

A: "**Born** Robert Allen Zimmerman in Duluth, Minn., on May 24, 1941, Bob Dylan grew up in nearby Hibbing."

Q: "What university did Thomas Jefferson **found**?"

A: "**Founder** of the University of Virginia, Jefferson feared that its board might get carried away with political or religious enthusiasms rather than choose a recipient based on scholarly considerations."

Note that for 955 QASPs (13.3%) MiniPar returned no parse at all. In most cases (727), it was the parsing of the answer sentence that failed. This is due to the often very long and complex nature of the answer sentences in the corpus, see Section 4.4.2.

It should also be pointed out, that the data presented for "to be" is slightly misleading: Here the figure presented in the column "Variation found" lists answer sentences where "to be" is not the head verb but occurs in some morphological form (including "was", "been" etc.). Because the verb "to be" is very common in English, especially with longer sentences chances are high that it occurs somewhere in it. Whether it

	No.	Found as head	Variation found	Not found at all
Overall	6225	33.9%	18.2%	47.8%
be	2688	39.0%	30.1%	31.0%
other than “be”	3537	30.1%	9.2%	60.6%
die	207	61.4%	1.4%	37.2%
locate	198	4.0%	2.5%	93.4%
bear	144	52.8%	19.4%	27.8%
make	132	28.0%	0.0%	72.0%
found	117	47.9%	28.2%	23.9%
have	115	13.0%	13.0%	73.9%
occur	111	8.1%	0.9%	91.0%
stand for	104	4.8%	0.0%	95.2%
win	84	28.6%	17.9%	53.6%
play	76	25.0%	15.8%	59.2%
hold	70	20.0%	0.0%	80.0%
invent	57	33.3%	8.8%	57.9%
marry	56	41.1%	10.7%	48.2%
kill	52	78.8%	3.8%	17.3%
take place	49	4.1%	0.0%	95.9%

Table 4.10: Analysis of the question’s head verbs and whether they also occur in the answer sentence either as the head verb as well (column 3), or in some morphological variation somewhere else in the sentence (column 4). Column 5 shows how often the head verb could not be found at all in the answer sentence. Column 2 lists the number of answer sentences that the results are based on, that is the number of answer sentences in the corpus paired with questions that show the listed head verb.

stands in some semantic relation to the question’s head verb is a different matter altogether. Note however that, because the collection methology ensures that the question and answer sentences on which this data is based show some significant semantic overlap, we usually can assume that, for the vast majority of cases involving other verbs, there indeed is some sort of semantic relation between the question’s head verb and its occurrence in the answer sentence.

4.4.5 Role of Context

It is not always the case that the answer to a question can be found in one single sentence together with the evidence (words and phrases from the question) that mark this sentence as an answer sentence. Often such information is distributed across several sentences, as when a sentence uses an anaphora to refer to an earlier introduced entity.

When creating the QASP corpus, we deliberately allowed subjects to select two or more consecutive sentences should they think this was necessary. Table 4.11 lists how often an answer sentence in the corpus in fact consists of multiple sentences. The data was produced automatically by using a sentence splitter script especially developed for this task. The script treats every ".", "!" or "?" followed by a whitespace character and an uppercase letter a sentence terminator, unless one of the following conditions apply:

- "." is preceded by Mr, Ms, Mrs, Dr and 16 other abbreviations.
- "." is preceded by exactly one whitespace followed by an uppercase letter (as in "George W. Bush").

This fairly simple algorithm was evaluated by checking the first 100 positive results it returned for the TREC 2002 data by hand. 98 of these turned out to be true multi-sentence constructs.

Table 4.11 also lists how often a QASP was tagged "C", which stands for context. As explained earlier, in such a case, the answer sentence selected does answer the question, but some context is missing to derive this from the sentence alone, see Section 4.3.5. In such a case it usually can be assumed that the missing information is present somewhere else in the document, but this is not necessarily the preceding sentence. As can been seen in the table, for the part of the corpus that is based on TREC data, 13.8% of all answer snippets require some form of context going beyond one sentence. This data is significantly larger for the part based on Lin's data, where it is 21.5%. This makes sense when considering that Lin's data is based on *every* document that contains the answer. Sentences selected based on TREC data contain only results that participating systems in TREC have found. Because multi-sentence evidence is harder to find, we would indeed expect Lin's data to contain a higher percentage of such evidence.

year	no. pairs	no. multi	tagged “C”	overlap	sum
2002	2006	190	52	3	239 (11.9%)
2003	1448	145	15	0	160 (11.0%)
2004	865	72	11	1	82 (9.5%)
2005	1456	89	152	9	232 (15.9%)
2006	1405	108	169	3	274 (19.5%)
02-06	7180	604	399	16	987 (13.7%)
Lin	1650	160	202	6	356 (21.5%)

Table 4.11: Analysis of the number of multi-sentence answers contained in the QASP corpus. The first column indicates the subsection of the data set, the second lists the total number of QASPs in that subsection, the third column lists how often subjects selected more than one sentence, the fourth column lists the number of QASPS tagged as “C”. In some cases where more than one sentence was selected, this instance additionally was tagged as “C”; their number is given in column five. The final column shows the sum of multi-sentences and sentences tagged as “C”, minus the overlap (because otherwise some QASPs would be counted twice.)

4.4.6 Summary of Analysis

The data provided in this section can assist the design of good QA algorithms. It furthermore is suitable to explain why certain methods in QA work better than certain other methods. It can for example be seen that:

1. Questions are much shorter than their answer sentences. This illustrates that the difficulty in QA is not so much question processing, but rather candidate sentence analysis. This also can be seen to suggest that answer sentences are more complicated on a syntactic level.

2. Less than 50% of lower case non-stop words from question re-occur in answer sentences, even if stemming is performed (without stemming this figure is lower than 40%). This illustrates that lexical variation between question and answer sentences is very high. Simple key-word based IR techniques are not sufficient to deal with this.

3. In more than 60% of all cases, the head verb of a question (excluding “to be”) is not present in the answer sentence in any morphological variation. This additionally illustrates that lexical variation between question and answer sentences

is very high–at a point that is crucial for many syntax-based QA techniques.

4. In almost 14% of all cases more than one sentence in a document is needed to answer the question. In most of these cases, anaphora resolution and/or discourse analysis would be needed to be incorporated into a QA system.

Points two and three of the list above, in an obvious manner, further strengthen the argument that paraphrasing is a very central problem in Question Answering. Yet, this also holds for the fourth point. TREC-style factoid questions usually ask for a fact in a very brief and precise way. Yet in 14% of all cases the answer fact is distributed among more than one sentence–this stongly suggests that some form of paraphrasing must have been involved, especially on a syntactic level.

4.5 Conclusions

In this chapter we have described the creation of a corpus of **Q**uestion **A**nswer **S**entence **P**airs (QASPS) with Amazons Mechanical Turk. The corpus contains 8,830 such pairs and it publicly available. The creation of this resource was inspired by the fact the methods reported in Chapter 3, despite working well in a web-based setting, failed to deliver good result when evaluated on a local corpus, the AQUAINT corpus. One of the reasons identified for this is that in the AQUAINT corpus, answer sentences often show some form of indirect evidence; in other words: They do not answer the question from a strictly logical point of view. By collecting a large set of answer sentences from the AQUAINT corpus, we created a resource that contains many examples of indirect evidence. We analyzed a few key features of this corpus with automatic methods and found strong evidence towards the necessity to develop QA systems with strong paraphrasing capabilities.

In the next chapter, we will describe an algorithm for QA that acquires syntactic and semantic knowledge from the QASP corpus. Key features of this algorithm will take the findings that have been made when analyzing the QASP corpus into account. Because the algorithm is based on the QASP data it will be able to discover many forms of indirect evidence, something that was not possible with the methods described in Chapter 3 which are based on lexical resources.

Chapter 5

Acquiring Syntactic and Semantic Reformulation Rules from the QASP Corpus

5.1 Introduction

The experiments concerning lexical resources like FrameNet, PropBank and VerbNet described in Chapter 3 have shown the following:

1. FrameNet and the like are useful to find answer sentences that contain *direct evidence* to the question.

2. Developing a QA strategy based on finding answer sentences with *direct evidence* is a useful path to explore for web-based QA, but when working with a smaller corpus it will miss too many answer sentences containing *indirect evidence*.

Considering that the algorithms so far were not suitable to identify answer sentences that show indirect evidence, it is natural to ask, what one would need to develop a method that is suitable to deal with cases of indirect evidence. The line of research proposed here offers a natural answer to this question: An algorithm that acquires indirect evidence from a large set of answer sentences that show many examples of indirect evidence is devised. (This procedure is somewhat parallel to the approach for direct evidence described earlier in this thesis, which is based on annotated sentences in FrameNet, PropBank and VerbNet–and as such on data exemplifying potential answer sentence formulations showing direct evidence.) The QASP corpus described in

Chapter 4 contains 8,830 question answer sentence pairs, where the questions were used in former TREC evaluations and the answer sentences come from the AQUAINT corpus. Many of the sentences in it can be assumed to contain indirect evidence, as the example sentences listed in Section 4.3 show. Thus it is a natural dataset to work with for the purpose at hand.

As far as the algorithm to be developed is concerned, three features seem especially important:

1. Important question terms need to be related to their corresponding terms in the answer sentence. (An answer sentence must provide evidence that it is related to the question.)
2. Not all question terms may co-occur in the answer sentences: Some might be missing, other might stand in a less obvious semantic relation to the question's terms. (Consider "purchased" in "When was Alaska purchased?": Other than the verb "purchase" in some morphological form the answer sentence might show the verbs "buy", "sell", "acquired" or the nouns "purchase", "acquisition" etc.)
3. The syntactic structures of the answer sentences need to be taken into account, because the important words/terms need to stand in the correct relation to each other.

There has been a line of research in question answering that seems suitable for the task at hand: syntactic structures of question and answer sentences have been captured by dependency relations. Unseen answer sentences are expected to show similar dependency relations between question/answer terms as previously acquired structures. Here, two approaches can frequently be found in the literature (see also Section 5.2 "Related Work"):

1. Some work analyzed the syntactic relations between terms or constituents in the question and sought to find the same relations in the answer sentences, e.g. [Attardi et al., 2001], [Katz and Lin, 2003] or [Bouma et al., 2005a].
2. Bootstrapping approaches start with a small set of question answer pairs to automatically find valid answer sentences, e.g. [Lin and Pantel, 2001] or [Mur, 2008].

The algorithm proposed here differs in that it starts with a large set of question answer sentence pairs. This overcomes the following problems:

1. An approach based on analyzing syntactic relations present in the question alone may miss many positive answer sentences, because other, often more complicated syntactic structures than the usually simple ones found in the question can be used in valid answer sentences.

2. Bootstrapping relies on seed instances, which have to be manually created. To date, the vast majority of all work evaluates bootstrapping approaches in IR and QA only on small sets of a few handpicked question classes. It therefore remains unclear if and how such approaches can be used for large question sets containing a large variety of questions.

The main benefit of using the QASP corpus over the described methods is that it contains a large number of example sentences obtained and checked by humans, for a large number of factoid questions. As such, the data is not only of high quality, but also covers a many ways in which answer sentences can be formulated. While we cannot know if our data set is big enough for the task at hand before having carried out the experiment, we will evaluate our approach on standard TREC test sets–as opposed to a few carefully selected question classes.

Let us briefly comment on a few key decisions we have made when designing the algorithm:

- A central notion of the lexical resources described in Section 3.2 is that of a predicate, which largely shapes the syntactic structure of a sentence by determining the number and location of its arguments. In most cases this predicate is a verb. Section 4.4.4 however showed that in 60.6% of all cases a question's head verb does not occur in the answer sentences identified in the AQUAINT corpus (excluding "to be"). Yet, there is one (and just one) constituent of which we can be absolutely sure that it must be present in every valid answer sentence: the answer itself. Therefore we make the answer the central anchor point in our algorithm. What used to be arguments for the predicate, in our paradigm are important constituents in the question (just as the arguments of a predicate they differ in number) that we expect to also be present in the answer sentence–possibly with a different surface form.

- The link between question constituents and the answer in the answer sentence is realized via dependency paths, which have been used in question answering before, e.g. in [Lin and Pantel, 2001]. Their exact style however differs greatly in different works. Our paths are inspired by those parse tree paths used in shallow semantic parsers, e.g. [Gildea and Jurafsky, 2002] and [Xue and Palmer, 2004] (see Section 3.3.1 of this thesis). The paths used in these papers are paths in phrase structure trees. However, as [Xue and Palmer, 2004] themselves point out phrase structure paths show limitations. For example, unlike dependency paths, they are not suited to distinguish NPs following a ditransitive predicate (e.g. “give”), because both would be the same.

- In Section 4.4.3 we saw that many key words from the question do not re-occur in the answer sentence. In many cases however we expect a semantically related word to be present in the answer sentence. We added a special processing step to our algorithm to deal at with some forms of semantically closely related words, which is based on WordNet.

- From [Ravichandran and Hovy, 2002] we borrow the rule evaluation step in our algorithm and also their formula for rule precision (see this chapter’s related work section).

In the remainder of this chapter (after in Section 5.2 related work is presented), we will describe an algorithm devised to acquire reformulation rules from the QASP corpus. The algorithm can be divided into three main steps:

1. **Rule Creation** The Question-Answer-Sentence-Pairs in the corpus are used to create rules. This is described in Section 5.4.

2. **Rule Evaluation** Other text in the corpus is used to assign a confidence value to each of the rules created during step 1. This is described in Section 5.5.

3. **Rule Execution** The rules are applied. This is described in Section 5.6.

One crucial step in this algorithm’s processing pipeline is concerned with aligning words found in the question to words in the answer sentences. Sometimes the words that have to be aligned are not morphologically, but only semantically related. This step is a prerequisite for all of the algorithm’s three main steps. It is therefore described in a section preceding the main steps: Section 5.3.

Section 5.7 describes experiments designed to evaluate the performance of the algorithm are set out and results are given.

Finally, Section 5.8 reflects on what has been achieved and puts our results in context.

5.2 Related Work

This section describes work related to our approach which acquires possible answer sentence formulations from the QASP corpus. We use dependency relations, more precisely dependency paths, to express the necessary syntactic constraints on the answer sentences. There has been a strong tradition in Question Answering of using dependency relations to interpret answer sentences. We therefore start by discussing the most relevant papers in this respect. After this we will broaden the focus and take a look at different ways of how paraphrases in Question Answering have been acquired so far.

5.2.1 Dependency Relations for Question Answering

[Attardi et al., 2001] describes a Question Answering system, PIQASso, that, after a set of answer sentences has been identified, matches dependency relations to extract answers. Questions and answer sentences are parsed with MiniPar [Lin, 1998b] and the dependency output is analyzed in order to determine whether relations present in a question appear in a candidate sentence as well. For the question “Who killed John F. Kennedy”, for example an answer sentence is expected to contain the answer as subject of the verb “kill”, to which “John F. Kennedy” should be in object relation. PIQASso also infers new relations by applying a set of nine different rules to MiniPar’s output. One of them for example is: “A and B are related with the relation of genitive if A is the subject of a verb *to have* and B is the object.” The purpose of this rule is to enable the matching of “John’s car” with “John has a car”. The system did not perform particularly well in TREC’s 2001 QA track, where it achieved an MRR of 0.271.

In [Katz and Lin, 2003], the authors identified two phenomena, *semantic symmetry* and *ambiguous modification*, which are difficult to handle by linguistically uninformed systems. The first problem concerns questions like “What do frogs eat?” and “What eats frogs?”, which use the same (non-stop) words, but have different semantics and therefore ask for a different answer. The second problem is concerned with modifiers like “largest” or “in the solar system”, which can modify different head nouns. When only key-words are used, the information about what exactly it is that is for example “largest” is completely lost. Yet preserving this information could potentially improve a QA system’s performance considerably. The authors chose to capture these phenomena with ternary expressions, which intuitively can be seen as subject-relation-object triples, which can hold many types of relations, for example subject-verb-object or relations of possession. These ternary expressions are created by parsing text with MiniPar and simplifying its dependency output, so that it fits into the ternary expressions paradigm. The question “What do frogs eat?”, for example, becomes [frog eat ?x], whereas “What eats frogs?” becomes [?x eat frog]. An ternary expression extracted from an answer sentence of the form [frog eat insect] would answer the first question but not the second. The authors achieve large precision gains (0.84 compared to 0.29 for a keyword-based method) on their specially crafted test set consisting of questions exemplifying only the two mentioned phenomena.

[Punyakanok et al., 2004] present an approach for answer selection which represents both questions and candidate passages using dependency trees. Both are compared by approximate tree matching. In their model, the sentence that best answers a question is the one that minimizes the generalized edit distance between it and the question tree. Their measure of edit distance is adapted form the usual definition of edit distance. As such it measures the cost of a sequence of operations that are needed to transforms one labeled tree to another. The operations include deleting a node, inserting a node, and changing a node. As their algorithm combines questions with complete candidate sentence tress, the authors in fact do not identify answers. They judge a question as correctly answered if it identifies a sentence in a document that contains the answer. As baseline, they use a bag-of-words approach, which can identify supporting documents for 28.85% of all 454 questions from the TREC 2002 question set for which such a document exists. With their tree matching method this figure rises to 40.31%.

[Bouma et al., 2005a, Bouma et al., 2005b] describes a Question Answering System for Dutch, Joost, in which dependency relations are used in question analysis, off-line answer extraction, answer reranking and identification of potential answers. The system uses hand-written dependency patterns, essentially a set of (partially underspecified) dependency relations which are compared against a set of dependency relations derived from parse trees. It furthermore makes use of 14 equivalence rules, enabling the system to recognize a set of semantic equivalences, even if two dependency analyses differ. Examples for this include (the Dutch versions of) constructions like “Zimbabwe gave asylum to Mengistu” and “Mengistu was given asylum by Zimbabwe” or “the coach of Norway, Egil Olsen” and “Egil Olsen, the coach of Norway.”

The authors found that use of the mentioned equivalence rules considerably increases the number of facts retrieved by the extraction patterns. The system performs off-line answer extraction, meaning that potential answers to certain question types are extracted from the corpus before the actual questions are known and stored in a database, which can be accessed in a fast manner when a question is asked. For questions which cannot be found in the database when they are asked, the system also includes a fall-back strategy based on traditional keyword-based paragraph retrieval. The previously mentioned dependency patterns are then used to identify answers and also play a role during answer ranking. The system performed well (0.544 accuracy for factoid questions) in the 2005 CLEF evaluation. [Vallin et al., 2005]

All of the papers mentioned so far compare the syntactic structure present in a question with the syntactic structure present in candidate sentences. In the following we will take a look at work that takes known good examples of answer sentences into account as well.

[Cui et al., 2005] describe a fuzzy dependency relation matching approach to passage retrieval in Question Answering. Here, the authors present a statistical technique to measure the degree of overlap between dependency relations in candidate sentences with their corresponding relations in the question. Question/answer passage pairs from TREC-8 and TREC-9 evaluations are used as training data. [1] To illustrate their approach we repeat below from their paper three relations extracted from the question “What percent of the nation’s cheese does Wisconsin produce?” and the answer sentence “In Wisconsin, where farmers produce roughly 28 percent of the nation’s cheese, the outrage is palpable.”

[1] These passages sometimes are very short and contain just the answer, sometimes they consist of sentences, sometimes of text snippets starting in the middle of one sentence and ending in the middle of another. Therefore this kind of training data has to be considered as rather messy.

Path_ID	Node1	Path	Node2
<P$_{Q1}$>	Wisconsin	<subj>	produce
<P$_{Q2}$>	produce	<head, whn, prep, pcomp-n>	cheese
<P$_{Q3}$>	nation	<gen>	cheese
<P$_{S1}$>	Wisconsin	<pcomp-n, mod, i>	produce
<P$_{S2}$>	produce	<obj, mod, pcomp-n>	cheese
<P$_{S3}$>	nation	<gen>	cheese

Taking data such as this the system for example aligns the <subj> path from the question with the <pcomp-n, mod, i> path in the answer sentence. It then learns relatedness between paths based on a statistical translation model, IBM's Model 1 [Brown et al., 1993]. While IBM's Model 1 assigns probabilities to alignments of one word in one language to another word in another language, the method at hand learns from the training data probabilities for the alignment of dependency relations present in the question to dependency relations present in the answer sentence. Below are some of these learned translation probabilities, taken from a related paper [Cui et al., 2004]:

Relation-1	Relation-2	Similarity
whn	pcomp-n	0.43
whn	i	0.42
i	pcomp-n	0.39
i	s	0.37
pred	mod	0.37
appo	vrel	0.35
whn	nn	0.34
s	num	0.33

Here, *whn* for example stands for a nominal wh-phrase (e.g. "**who** escaped") and *pcomp-n* for a nominal complement of a preposition (e.g. "in the **garden**"). (For more detailed explanations about MiniPar's dependency relations see [Lin, 2003].) Note that these translation probabilities are completely independent of context (such as preceding or consequent relations in a path, other paths in the question or answer sentence or the question class). The system then calculates a score for path alignment by finding the most probable mapped relation in the path from the question for each relation in the aligned path from the sentence based on the relation translation probabilities. Finally, the sums of all path alignment scores for a question answer passage pair are summed up and the passages are ranked according to this score. The authors report an improvement of 31% in MRR when comparing their fuzzy relation matching technique with a strict matching variant.

While [Cui et al., 2005] is concerned with passage retrieval, [Cui et al., 2004] uses a very similar model for answer extraction. In each sentences returned by the IR module, all named entities of the expected answer types are treated as answer candidates. For questions with an unknown answer type, all NPs in the candidate sentence are considered. Then those paths in the answer sentence that are connected to an answer candidate are compared against the corresponding paths in the question, in a very similar fashion to [Cui et al., 2005]. The candidate whose paths show the highest matching score is selected. The system was evaluated in TREC 2004. Their baseline method, relying only on named entity information achieved an accuracy of 0.51. Two slight variations of their new answer extraction method achieved an accuracy of 0.62 and 0.60.

While both [Cui et al., 2005] and [Cui et al., 2004] are similar to our approach in that they utilize answer sentences, there are also considerable differences. Crucially, although the approach makes use of answer sentences, it still sticks to the general idea of comparing dependency relations present in the questions to those present in a candidate sentences. Like some of the earlier mentioned work, it implements a measure that evaluates how related these paths are to each other rather than requiring strict similarity. Different from earlier mentioned work, here the path relatedness measure is based on comparisons of paths from the question with paths found in valid answer passages. What the authors correctly have observed is that sometimes, especially with MiniPar, a relation in the question is not present in the same way in an answer sentence. A question starting with “Where” for example will contain an *wha* relation, indicating an adverbial wh-phrase. This wh-phrase of course will not be present in the answer sentence, instead a *pcomp-n* relation, a nominal complement of a preposition (e.g. “in **Prague**”) might be present. Taking this into account when comparing questions with candidate sentences is certainly helpful.

Yet the approach is very coarse. It is entirely based on one similarity matrix containing pairs of dependency relations, which is valid across all types of questions. In reality, these similarities might be very different for different question classes. Also, it treats every path in the question independently. This can be problematic. For example, if a constituent that is the subject in a question becomes the object in an answer sentence, obviously the object of the question cannot function as the object in the answer sentence as well. It might instead become the subject, as in passivation. This shows that, if one particular path between two constituents in a question changes in the answer sentence, other paths might very well (or even have to) change as well. In order to capture these kind of transformations, paths cannot be treated independently. The bottom line is that this method is not suitable to detect candidate sentences that show a completely different sentence structure from the question. Despite taking known answer passages into account, it still scores candidate sentences according to their similarity to the question.

[Shen and Klakow, 2006] describes a method similar to [Cui et al., 2004] and [Cui et al., 2005], also primarily based on similarity scores between dependency relation pairs. Their algorithm computes the similarity of paths between key phrases, not between words. Furthermore, it takes relations in a path not as independent from each other, but acknowledges that they form a sequence, by comparing two paths with the help of an adaption of the Dynamic Time Warping algorithm [Rabiner et al., 1991], which is often used in speech recognition to deal with different speaking speeds of voice input.

In the next section we will describe an algorithm that learns possible answer sentence formulations for syntactic question classes from the example sentences contained in the QASP corpus. Unlike the work described so far in this section, it acknowledges that:

- A valid answer sentence's syntax might be very different for the question's syntax.
- Several valid answer sentence structures, which might be completely independent from each other, can exist for one question.

We furthermore decided that, although fuzzy matching certainly is an interesting line of research and potentially could be combined with our approach, our method will be based on strict matching of dependency paths. The reason for this is that fuzzy matching, as we have seen, already has been tried many times and that we want to see what performance increase we can achieve by switching from analysing the structure of questions to analysing the structure of answer sentences alone.

Before describing our approach in more detail, we will review another body of relevant work in the next section.

5.2.2 Learning of Paraphrases for Question Answering

In the following we will take a look at work that is concerned with discovering paraphrases for Question Answering.

[Lin and Pantel, 2001] present an unsupervised algorithm to automatically discover inference rules (essentially paraphrases) from text in order to enhance a Question Answering system. These inference rules are based on paths between words in dependency trees, each of which connects two nouns. Their paths have the following form:

N:subj:V←find→V:obj:N→solution→N:to:N

This path represents the relation "X finds a solution to Y".

The authors start by parsing 1GB of newspaper text with MiniPar and extract 7 million paths like the one above from the resulting parse trees (231,000 of which are unique). These are stored in a database, which contains frequency counts for triples consisting of the path itself, a word that was found either at the start or at the end of the path in the corpus (the X or Y in the relation above), and a variable indicating whether that word was found at the beginning or the end (whether that word fills the X or Y slot). Then the similarity between the collected paths is computed, by adapting the Mutual Information measure, often used to measure association strength between two words, to paths. Paths that often can be found in the corpus with the same word at their ends receive a high similarity value. (Their formula also takes the frequency of the words in the corpus into account.) For certain paths, other paths in the corpus are sorted according to their similarity value, and the paths with the highest values are used as paraphrases. For "X solved Y", for example, the top-5 most similar paths the authors identify are:

- "Y is solved by X"
- "X resolves Y"
- "X finds a solution to Y"
- "X tries to solve Y"

- “X deals with Y”

The authors do not evaluate their algorithm by using it in a QA system. Rather, they compare their paraphrases to a set of human-generated paraphrases for the first 15 questions used in TREC’s 1999 QA track and also manually inspect the 40 highest scoring automatic paraphrases on whether they are suitable for finding answers to the questions. The intersection between human-generated paraphrases (between 2 and 14 for each question) and automatically generated paraphrases was found to be quite low. On the other hand, many of the system’s paraphrases turned out to be valid (e.g. 92.5% for “X manufactures Y” or 52.5% for “X is author of Y”.)

[Ravichandran and Hovy, 2002] explore the use of surface text patterns for a Question Answering system. For a small set of question types the authors learn regular expressions that describe potential answer templates, e.g. “<NAME> was born in <BIRTHDATE>” which matches strings like “Mozart was born in 1756” and is suitable to answer questions of the form “When was X born?” Their algorithm starts by submitting a set of seeds, here known question term and answer term pairs, (“Mozart” and “1756” for the example) to a search engine. The top 1000 documents are downloaded, broken into sentences and only those are retained that contain both seed terms. Common re-occurring substrings are searched for in these sentences and again only those are kept that contain the seed terms. Then the word in these phrase for the question term is replaced by <NAME>, and the answer term by <ANSWER>. This is repeated with different seed terms, for the birthdates they for example also use “Gandhi 1869” and “Newton 1642”. The overall most common substrings are stored, for birthdates some of these are:

```
born in <ANSWER> , <NAME>
<NAME was born on <ANSWER>,
<NAME> ( <ANSWER> -
<NAME> ( <ANSWER> - )
```

Their approach however does not treat these patterns as the final result. A subsequent processing step determines the precision of each pattern. Again, a search engine is queried, this time however by only using the question term (“Mozart”) and the top 1000 documents are extracted and broken into sentences. Only those that contain the question term are retained and for each pattern determined earlier it is checked whether it is contained in these strings, either with the correct answer found at the <ANSWER> slot, or with some other term. Pattern precision is then calculated by dividing the number of instances where the pattern matches with the correct answer term by the number of instances where the pattern matched with any answer term. For the birthdates example they obtain the following values:

```
1.0  <NAME> ( <ANSWER> - )
0.85 <NAME was born on <ANSWER>,
0.6  <NAME was born in <ANSWER>
0.59 <NAME was born <ANSWER>
0.53 <ANSWER> <NAME> was born
```

These patterns are then used to find answers. Patterns with a high precision are given precedence over pattern with a lower precision. Beside the mentioned birthdate question type the same approach was also applied to questions about inventors and discoverers, definitions and locations and questions about why a person is famous. The system was evaluated in a web-based setting and separately by searching a local corpus. Six different question types are used for evaluation: birthdate, location, inventor, discoverer, definition and why-famous. Results vary a lot across these question types. Their best score (MRR 0.88) is for the discoverer questions when querying the web; their worst score (0.00) is for Why-famous questions, also in a web-based setting.

[Ibrahim et al., 2003] present an approach to automatically learn paraphrases from aligned monolingual corpora. They use different translations of foreign novels, for example two translations of *20,000 Leagues Under the Sea*, two translations of *The Kreutzer Sonata*, and three translations of *Madame Bouvary*. Sentences are aligned using the Gale and Church algorithm [Gale and Church, 1991], and parsed by the dependency-based Link parser [Sleator and Temperley, 1993]. Anchors are identified within the aligned sentence pairs, which can only be nouns or pronouns. Exact string matches qualify as anchors. Other matches based on, for example, the longest common substrings penalizes the score by 50%. A breadth-first search is used to find the shortest dependency path between the anchor words. (These dependency paths, which they treat as paraphrases are very similar to [Lin and Pantel, 2001].) If valid paths can be found between anchor pairs in both of the aligned sentences, they are considered candidate paraphrases. These are then scored, taking the frequency of anchors and their variety for each paraphrase into account. The authors are able to extract 5,502 unique paraphrase pairs from the corpora. 130 of these are randomly chosen and then judged by three human assessors, their average precision is 41.2%. The paper then discusses the use of these paraphrases for a Question Answering system, but no evaluation is given in this respect.

[Snow et al., 2005] present an algorithm for learning hypernym (is-a) relations from text. The work carried out here follows the work of [Hearst, 1992] in that is uses lexico-syntactic patterns as cues that indicate a particular semantic relationship between two nouns. While Hearst proposed the use of a set of hand-crafted extraction rules, [Snow et al., 2005] propose a machine learning paradigm that automatically learns these rules. For training, they extract all pairs of words in a hypernym/hyponym relation from WordNet. For each pair sentences are searched in a newspaper corpus in which both words occur. (Note that for some pairs, no sentences might be found.) These are parsed with MiniPar and patterns based on the dependency paths between these two words are extracted. A hypernym classifier is trained using the extracted dependency paths as features. For testing, the classifier makes a boolean decision whether a pair of words from an unseen sentence is in a hypernym/hyponym relation or not. If this word pair is contained in WordNet, it is easy to check whether the classifier has made the right decision or not. When comparing their automatically learned patterns with Hearst's manually created patterns in this way, they achieve a 132% relative improvement. Interestingly, [McNamee et al., 2008] uses a slightly modified version of the method described in [Snow et al., 2005] to improve the semantic type checking component of a Question Answering system. This is useful because WordNet, which is often used in QA to check an answer on the correct semantic type, contains very little information about Named Entities. Beside using dependency paths between the hypernym and hyponym as features in this paper, the authors additionally employ twelve other kind of features including capitalization, common suffixes (e.g. -ation, -ment, -ology) etc. When evaluated on TREC 2005 & 2006 data, their system achieves 27.3% accuracy, compared to 18.26% based on WordNet alone. (The paper contains no data about how the system would perform when using a Named Entity Recognizer for answer type checking.)

Our work differs from the above work in that it actually provides an algorithm suitable to extract answers for potentially all types of factoid question. Most of the above approaches, although described as being for Question Answering, in fact detect paraphrases between declarative sentences. How these should be connected to the question is not detailed. Accordingly, [Lin and Pantel, 2001] and [Ibrahim et al., 2003] evaluate only the validity of their paraphrases, not how they can help to detect answers. [Ravichandran and Hovy, 2002] uses six different factoid question types for evaluation (see above), where for each question type a manual selection of example seed terms is required. For many other question types their approach would not work. This is because [Ravichandran and Hovy, 2002] (the same holds for [Lin and Pantel, 2001]) can only handle paraphrases that express relations between exactly two words. [2] For factoid question answering, this is a simplified assumption: Many questions contain more than two important keywords.

5.3 Word Alignment

For the approach at hand, in order to be able to create rules, for every question/answer sentence pair in the QASP corpus, corresponding constituents in each question and answer sentence have to be aligned. Consider the following example:

Question: "When was the Alaska territory purchased?" Answer: "1867"

Answer sentence: "The acquisition of what would become the Territory of Alaska took place in 1867."

The mapping that has to be achieved looks like this:

Question Term	Answer Sentence Term
"Alaska territory"	"Territory of Alaska"
"purchased"	"acquisition"
ANSWER	"1867"

The algorithm described in this section is concerned with the alignment of semantically related words from the question and the answer sentence. This falls one step short of what we actually need to achieve, which is an alignment of constituents (like

[2] The same holds for [Snow et al., 2005], but here the authors are concerned with the extraction of hyponymy relations, which always exist between exactly two words.

"Alaska territory" and "Territory of Alaska" in the example above). Alignment of constituents is performed after words have been aligned and the constituent alignment step takes the word alignment results as an input. This section describes the alignment of semantically related words.

Word Alignment is important in the field of Machine translation where words in parallel, bilingual corpora have to be aligned. There, it has been extensively studied, see for example [Och and Ney, 2003] for a comparison of various statistical alignment models. In our setting however, we have to align questions to answer sentences that are in the same language (English). This enables us to use means that would not be possible for bilingual alignment. Firstly, we expect many question words to be present in the answer sentence as well, possibly with exactly the same surface appearance or alternatively in some morphological variant. It is fairly straightforward to assign such words as we will see. Furthermore, for the remaining words we can use tools that can tell us how semantically related two words are, most notably WordNet. One disadvantage that we face over methods as described in [Och and Ney, 2003], is that the size of the data available for training is significantly smaller. [Och and Ney, 2003] use two training sets, with 34K and 1470K aligned sentences. Would we want to use the QASP corpus as training data for one of these approaches we would only have roughly 8K of QASPs available. Because of these differences we decided to implement a custom build alignment strategy.

In our approach, every word in the question which is not a stop word or a wh-word will be subject to alignment. For the above example this means "when", "was" and "the" will not be aligned. As far as the remaining words are concerned, the alignment of "Alaska" and "territory" is trivial, the same holds for the answer "1867". (All three are similar on string level.) Yet, although not the case in the example, a question might use different surface forms of a word than the answer sentence (e.g. "purchase" and "purchased".) To detect such cases we employ a stemmer based on the Snowball algorithm by Martin Porter (in the implementation that comes with Lucene's Sandbox, i.e. the `SnowballFilter` class [Porter, 2001, Hatcher and Gospodnetić, 2004]). Additionally we use the University of Pennsylvania's morphology database [Daniel et al., 1992] which contains 317,322 words in different morphological forms. If one of these tools reports that two strings are morphological variants of the same words they are aligned. (The same approach is used in Section 4.4.2.)

The alignment of "purchased" to "acquisition" is more problematic. These words are semantically related, not morphologically. In the following we will describe several approaches that have been implemented and evaluated to this end. The approaches are similar in that each picks one word that has to be aligned from the question at a time and compares it to all of the (non-stop) word in the answer sentence. Each of the answer sentence words is assigned a value between zero and one expressing its relatedness to the question word. The highest scoring word, if above a certain threshold, is selected as the closest semantic match. Most of the measures employed make use of WordNet::Similarity, a Perl software package based on WordNet that makes it possible to measure semantic similarity (or relatedness) between a pair of word senses by returning a numeric value that represents the degree to which they are similar or related.[Pedersen et al., 2004] [3] Additionally, we developed a method bespoke for the problem at hand. In the following, all measures will be described and be evaluated on a small corpus created especially for that purpose.

5.3.1 Word Alignment with WordNet::Similarity

WordNet::Similarity currently contains ten measures, which are briefly described below. (For more details, please refer to the provided citations or go to http://wn-similarity.sourceforge.net/, where a nice web interface to try out the different measures can be found.)

Three of the similarity measures in WordNet::Similarity are based on path lengths between concepts:

Path length A simple measure, where the relatedness score is inversely proportional to the number of nodes along the shortest path between the synsets.

Leacock & Chodorow takes the maximum depth of the taxonomy into account. [Leacock and Chodorow, 1998]

Wu & Palmer calculates relatedness by considering the depths of the two synsets in the WordNet taxonomies, along with the depth of the least common subsumer (LCS), the most specific concept they share as an ancestor. [Wu and Palmer, 1994]

Three similarity measures are based on information content (IC):

[3] Measures of similarity quantify how much two concepts are alike, based on information contained in WordNet's ISA hierarchy. Measures of relatedness are based on additional sources of information, e.g. other relations beside ISA or WordNet's glosses. As such they can be applied to a wider range of concept pairs.

Resnik returns IC(LCS), where IC(x) is the information content of x and LCS the least common subsumer. [Resnik, 1995]

Jiang & Conrath returns 1 / (IC(synset1) + IC(synset2) - 2 * IC(LCS)).
[Jiang and Conrath, 1997]

Lin returns 2 * IC(LCS) / (IC(synset1) + IC(synset2)). [Lin, 1998a]

Four measures are relatedness measures:

Hirst & St-Onge finds lexical chains linking the two word senses. Three classes of relations are considered: extra-strong, strong, and medium-strong.
[Hirst and St-Onge, 1998]

Adapted Lesk (Extended Gloss Overlaps) works by finding overlaps in the glosses of the two synsets. [Banerjee and Pedersen, 2003]

Gloss Vector forms second-order co-occurrence vectors from the glosses or WordNet definitions of concepts. It takes glosses of adjacent Synsets into account.
[Patwardhan, 2003]

Gloss Vector (pairwise) is a slight variation of the Gloss Vector measure.

To test the usefulness of each of the measures for the purpose at hand they were evaluated on a small hand-annotated subsection of the QASP data. To create this corpus (in the following called the “alignment corpus”) the first 75 questions in the TREC 2002 subsection of the QASP corpus were taken into account. In these, we automatically determined all answer sentences tagged as “1” (thus all undoubtedly supportive sentences) for which at least one question word could not be aligned to a word from the corresponding answer sentence purely by taking morphological relatedness into account. This resulted is 101 question word/answer sentence pairs. In the sentences, all words that could potentially serve as candidates for the alignment process were automatically determined. This was done by excluding stop words and words belonging to certain parts of speech (e.g. proper names, numbers, symbols etc).

In the following QASP, for example, the term “purchased” could not be aligned:

Question: “When was Alaska purchased?” **Answer:** “1867”

Answer sentence: "In Seward, the town named for Secretary of State William Seward, who bought Alaska for $7.2 million in 1867, a multimillion-dollar industry has developed around ships that take visitors to the bird rookeries and glaciers of Kenai Fjords National Park."

The list of alignment candidates, here presented with their lexical category (WordNet-style, as determined by parser), is:

1. town#n
2. name#v
3. buy#v
4. million#n
5. multimillion-dollar#a
6. industry#n
7. develop#v
8. ship#n
9. take#v
10. visitor#n
11. bird#n
12. rookery#n
13. glacier#n

This data, for all 101 unaligned question words and their corresponding sentences, were written to a file and, for each sentence, the word that is correct alignment choice was manually marked. As it turns out, in many cases a question word simply cannot be found at all in an answer sentences. (In such a case the alignment algorithm needs to decide to not align the question word at all.) In other cases, a related word is present, but it is somewhat doubtful whether the alignment algorithm should pick it up. Consider the following example:

Question: "How do you say house in Spanish?" **Answer:** "casa"

Answer sentence: "Justice Stephen G. Breyer, in apparent support of Backstrom's argument, noted that home owners often use the Spanish phrase "mi casa es su casa" - my house is your house - to make their social guests feel at home."

Arguably, the closest semantically related word to "say", somewhat fulfilling the same function as in the question, is "phrase", which indeed might be somewhat useful as an indication that this sentence contains the answer to the question. Yet is it doubtful whether we want to use this alignment in a rule; the relation is somewhat vague. On the other hand it would be unreasonable to penalize an algorithm for picking it up. Therefore it was decided to mark certain words in the alignment corpus as possible (yet not necessary) matches. Note also, that in a few cases it was necessary to mark more than one word in one sentence in such a way. Following the described procedure for all 101 sentences in the test set, in 37 exactly one word was marked that should **definitely** be matched to the question word (the tag “D” was used), in 21 sentences no such word was marked, but at least one word was marked that **potentially** could be matched (as “P”), and in 43 no word was marked at all. [4]

The alignment corpus was then used to evaluate the performance of the WordNet::Similarity measures described earlier. Each of the 101 question words was aligned to the candidate words from the 101 answer sentences automatically by using the different measures. This was done by having the measures assign a similarity value to each candidate word. The word with the highest value was selected as the result. There are nine possible outcomes of this process per question word. For example, in a sentence containing a word that definitely should be matched (let us call this an D-sentence), the algorithm could have selected that word (an D-Word). But it also could have selected a word marked as a potential, but not necessary match (a P-word), a word which is not marked at all (a C-word, C for candidate), or the algorithm could have returned no word at all (“-”). Table 5.1 shows all nine possibilities together with the scores that were assigned to each of the outcomes. High scores represent desirable alignments, while low scores represent wrong alignments.

Using this scoring scheme, the best possible result that an algorithm can achieve is (based on the sentence types in the corpus and the best possible score that can achieved for each): 2*37 + 1*21 + 2*43 = 181. Here, in all 37 answer sentences that contain a word that definitely should be aligned that word has been found, and therefore a score of 2 is assigned; in all 21 sentence containing a potential (but not necessary) match, that potential match is correctly identified, resulting in a score of 1; finally in all 43 sentences that do not show any word related to the listed question word, the algorithm correctly returned no result, achieving a score of 2 for each of these 43 sentences. Similarly, the worst possible value an algorithm can achieve is -1*37 + 0*21 + -1*43

[4] Our approach here is somewhat similar to [Och and Ney, 2003], where the authors perform a comparison of various statistical alignment models for Machine Translation. They also use necessary and possible alignments in their reference alignment set.

Sentence Type	**Word selected**	**Score**
D	D	+2
D	P	+1
D	C	-1
D	-	-1
P	P	+1
P	C	0
P	-	0
C	C	-1
C	-	+2

Table 5.1: Possible outcomes of the word alignment process.

= -80.

Performance of all ten measures described earlier was evaluated on the alignment corpus in the described way. Most measures return similarity values between zero and one. For those that do not, their output was converted to be in the range between zero and one (by dividing the result by the highest result they ever returned while being tested on the alignment corpus). In the evaluation corpus, there is a considerable number of sentences that contain no word that should be aligned (43 out of 101). Yet, most measures will always return a positive value (albeit a small one), even if the compared words are not closely related. Thus most algorithms will almost always return one word–a behaviour which is not desired. To counteract this, each measure was evaluated with a cutoff value that would only take values above or on a certain value into account.

Table 5.2 shows the results of all measures being tested on the alignment corpus. Results for a random baseline are also reported. This baseline assigns a randomly generated value larger than 0.0 and smaller than 1.0 to each word. As with all other measures the word with the highest value is chosen. For all measures, results are given for different cutoff values. As can be seen, setting the cutoff value in a reasonable way is important. With no cutoff value (Column 2, “0.0”, taking all results greater or equal 0.0 into account) results are a lot worse than results from the same measure with a higher cutoff. Note that the ideal setting for the cutoff value differs considerably for each measure.

Measure	Cutoff											Processing
	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0	Time
Path length	-29	-23	21	31	58	55	55	55	55	55	49	2m 24s
Leacock & Chodorow	-26	-26	-26	-26	-23	-20	-6	25	52	55	55	2m 23s
Wu & Palmer	-26	-26	-26	-26	-25	-9	3	19	40	58	49	4m 00s
Resnik	-34	-29	-26	-5	27	40	52	49	49	46	46	2m 27s
Jiang & Conrath	-26	-26	-26	-26	-20	-18	16	52	55	55	55	2m 29s
Lin	-23	-17	-14	-8	24	45	49	**61**	**61**	55	49	2m 22s
Hirst & St-Onge	-17	30	33	58	58	58	58	58	58	49	49	9h 55m 37s
Adapted Lesk	-38	-18	-1	19	37	55	**61**	58	58	**61**	58	10m 09s
Gloss Vector	-36	-36	-36	-36	-36	-36	-34	-28	-24	35	52	1h 37m 40s
Gloss Vector (pw)	-36	-36	-36	-36	-36	-36	-34	-28	-24	35	52	1h 43m 29s
Random baseline	-62	-62	-62	-62	-62	-61	-60	-57	-51	-32	49	appox. 50ms

Table 5.2: Performance of the ten different WordNet::Similarity measures when used to find the semantically closest related word in an answer sentence to a word from a question.

It can be seen that all measures outperform the random baselines by a large margin. The exception is the random baseline with a cutoff value of 1.0. In this case however, the "random" baseline is not random anymore. With a cutoff value of 1.0, this baseline (which was implemented to return values smaller than 1.0) will always return no result, which, taking the nature of the test set into account, is a reasonable strategy that is tough to outperform: On average there are 11.76 words for each question term in the alignment corpus between which a decision has to been made. In 43 out of 101 cases all of them are wrong choices, because no semantically related word is present in the answer sentence. Here returning no result is the correct decision. In the remaining 58 cases one word (in four of these cases two words) is correct while the others are wrong. In this situation retuning no results is penalized, but so would returning a wrong result. Hitting a correct result by chance is not very likely. From this follows that only cautious, very informed choices can be expected to perform better that the always-return-no-result strategy. Note that this, because the alignment corpus was created from a subset of actual cases occurring when processing data from the QASP corpus, reflects the nature of the problem we are dealing with: In many cases the word in question has no match in the answer sentence, and choosing a wrong word is always a bad choice, but unfortunately there are more wrong possibilities that correct ones.

5.3.2 A Bespoke Strategy for Word Alignment

These observations led to the development of a custom made alignment strategy also based on WordNet. It takes the following considerations into account:

1. Many of the measures in the WordNet::Similarity package take only hyponym/ hypernym relations into account. This makes aligning word of different parts of speech difficult or even impossible. However, such alignments are important for our needs.

2. Many of the measures return results, even if the two words are not closely related at all. For our purposes however, only strong semantic relations should be taken into account.

Our strategy, also based on WordNet, is given below as pseudocode:

```
INPUT: w1, w2 (words, as strings)
SET n = 0
FOR each sense s1 of w1 in WordNet
  FOR every item i connected via a pointer to s1
    IF i is or contains any sense of w2
      INCREASE n
    END IF
  END FOR
END FOR
SET n = (n/( number of senses of w1 in WordNet
           * number of senses of w2 in WordNet))*3
IF (n > 1) THEN n = 1
RETURN n
```

Figure 5.1: Pseudocode for the word alignment strategy.

Crucially, this only accepts two words as related if any of their senses are directly connected by a pointer in WordNet. On the other hand it takes all pointers in WordNet into account, not just hyponymy/hypernymy. The multiplication of the result with three near the end is done to increase the result, which often showed to be very small during experimentation.

Results of this measure are reported in the first row (“WN Pointers”) in Table 5.2. It can be seen that the best result returned (90) is significantly higher than the best result

Measure	Cutoff											Processing
	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0	Time
WN pointers l=1	**90**	**90**	**90**	87	87	87	87	87	87	84	49	13s
WN pointers l=2	60	71	68	52	46	44	47	47	47	47	49	2m 42s
WN pointers + Lin	15	21	24	30	62	85	90	**101**	96	93	49	2m 33s

Table 5.3: Performance of the two additional relatedness measures when used to find the semantically closest related word in an answer sentence to a word from a question.

obtained from the WordNet::Similarity measures (61). In the second row, we slightly modified the approach by allowing two senses of two input words to be connected by a chain of pointers of length two, not one. As can be seen this hurts performance. For the values in the third row, the "WN Pointers l=1" method was combined with WordNet::Similarity's "Lin" measure. How this was done can be seen in formula 5.1. Here r stands for the returned result, wnp for the result of the "WN Pointers l=1" method (which has already been multiplied with three, see pseudocode) and lin for the result of the Lin measure.

$$r = \begin{cases} 1 & \text{if } wnp*2 >= 1 \\ wnp*2 & \text{if } wnp*2 < 1 \ \& \ wnp*2 >= lin \\ lin & \text{if } wnp*2 < 1 \ \& \ wnp*2 < lin \end{cases} \tag{5.1}$$

This in fact substitutes small WN Pointers values with Lin values, if the latter are greater. As can be seen, combining both measures achieves a further increase (to 101). Different ways of combining WN Pointers with Lin, as well as with other WordNet::Similarity measures were tested, but the method reported here returned best results. It is this method which will be used in the experiments described in the next section.

5.4 Rule Creation

After question words are aligned to words in the answer sentence, both question and answer sentence are parsed with the StanfordParser [Klein and Manning, 2003b], [Klein and Manning, 2003a]. Questions are matched against a set of simple, predefined, syntactic patterns like the ones in Table 5.4. [5] These patterns are borrowed from the QuALiM system, see Chapter 2 of this thesis.

[5] In these patterns COP stands for copula; X for any sequence of words with a length greater than one.

Pattern name	Example question
What+COP+NP	What is the French national anthem?
Who+COP+NP	Who is the governor of Colorado?
Where+COP+NP	Where is Devil's Tower?
What+X+COP+NP+in/by/on	What county is Wilmington, Del in?
When+COP+NP	When was the Rosenberg trial?
Where+was+NP+VERB	Where was Abraham Lincoln born?
When+was+NP+VERB	When was Alaska purchased?

Table 5.4: Examples of question patterns used by the algorithm.

Currently 244 such patterns exist. They serve to inform the algorithm about important segmentations present in the question. While, in the last section a method to align question words to answer sentence words was described, this falls one step short of what is actually necessary: Aligning phrases. In the first example given in Table 5.4, the NP "French national anthem" should be somewhat preserved in the answer sentence. A sentence like "The French military band played the British national anthem to welcome Gordon Brown." contains all three words from the NP but not in a single phrase. This is a strong hint towards not treating it as a sentence containing the answer to the question.

The first decision to be made when aligning question constituents is to decide which of them have to be aligned and which do not. In order to do this, every question constituent is checked as to whether it contains at least one word that is not a stop or wh-word. If this is the case, a matching phrase is sought in the answer sentence.

In order to align a question constituent, every word it contains is aligned to a word in the answer sentence using the method described in Section 5.3. (This might result in some words having no alignments.) If the question constituent contains only one word, the process is finished. If it consists of two or more words, every phrase in the answer sentence is checked on how many of these words it contains and a value is computed by dividing this number by the total number of non-stop-words in that phrase. The answer sentence phrase that returns the highest value is retained, together with the top node of that phrase. We now have a set of mappings of important question constituents to nodes in the parse tree of the answer sentence. Note that it might happen that one individual question constituent is not mapped to any node, or that it is mapped to more than one nodes (representing different alignment alternatives).

Once this is done dependency paths from all relevant phrases to the answer are extracted and stored. Below is the dependency output for the sentence “The acquisition of Alaska happened in 1867.”, together with the dependency paths extracted from it:

```
1: The         (the,DT,2)         [det]
2: acquisition (acquisition,NN,5) [nsubj]
3: of          (of,IN,2)          [prep]
4: Alaska      (Alaska,NNP,3)     [pobj]
5: happened    (happen,VBD,null)  [ROOT]
6: in          (in,IN,5)          [prep]
7: 1867        (1867,CD,6)        [pobj]
```

Alaska⇒1867: ⇑[4]pobj⇑[3]prep⇑[2]nsubj⇓[5]prep⇓[6]pobj
acquisition⇒1867: ⇑[2]nsubj⇓[5]prep⇓[6]pobj

The numbers in square brackets point to the corresponding nodes in the dependency tree. They are added here so that the paths can be easier understood; they are however not stored by the algorithm. Figure 5.2 shows the algorithm as pseudocode.

These dependency paths then become the consequent of a rule. The pattern the question matched earlier on becomes the antecedent. For the given example the rule would be:

Rule:
Pattern: When[1]+was[2]+NP[3]+VERB[4]
Path 3: ⇑pobj⇑prep⇑nsubj⇓prep⇓pobj
Path 4: ⇑nsubj⇓prep⇓pobj

As can be seen the pattern is stored, together with numbers assigned to each constituent. The numbers for constituents for which alignments in the answer sentence were sought for are listed together with the resulting dependency paths. If no alignment could be found for a constituent, *null* is stored instead of a path. Should there be two or more constituents that were identified as alternative possibilities for one question constituent, we create additional rules, so that each rule contains one of the possibilities.

The described procedure is repeated for all QASPs in the corpus and for each, one or more rules are created.

```
INPUT: set of QASPs qasps
CREATE empty rule set rs
FOR each qasp in qasps
  FIND matching pattern p for question q in qasp
  FOR each sentence s in qasp
    FIND constituent ca in s that corresponds to answer a from qasp
    CREATE empty constituent list cs
    FOR each constituent c in q as determined by p
      FIND constituent c2 in s that corresponds to c
      IF c2 exists THEN ADD c2 to cs
    END FOR
    DETERMINE paths hs between ca and all constituents in cs
    CREATE rule r from pattern p and paths hs
    ADD r to rs
  END FOR
END FOR
RETURN rule set rs
```

Figure 5.2: Pseudocode for the rule creation algorithm.

5.5 Rule Evaluation

It might seem like the rules created in the last section could be used straightaway to locate answers. After all, they were created from hand-checked, valid answer sentences and so the syntactic structures they contain should be valid as well. This, however, is not necessarily the case. Assuming that the word alignment module fails to spot the relatedness between “married” and “spouse”, the question “Who is Tom Cruise married to?” and the answer sentence “Actor Tom Cruise and his spouse Nicole Kidman ...” might, for example, result in following rule:

Rule:
Pattern: Who[1]+is[2]+NP[3]+married[4]+to[5]
Path 3: ⇓conj
Path 4: null

This rule however returns all sorts of wrong results, e.g. “Anthony Hopkins” based on the sentence “The movie stars Anthony Hopkins and Tom Cruise and is directed by John Woo.” The reason for this is that the above rule simply requires “Tom Cruise” to be the second argument of a conjunction. It does not require the presence of any word somehow related to the concept of marriage and does not require “Tom Cruise” to be connected to it.

In order to overcome the problem of rules that are too general, an additional processing step between rule creation and rule application was implemented: Rule evaluation. Similar approaches have been described in the relevant literature, many of them concerned with bootstrapping, starting with [Ravichandran and Hovy, 2002], for more details see Section 5.2. The general purpose of this step is to use the available data about questions and their correct answers to evaluate how often each created rule returns a correct or an incorrect answer. Typically, this data is stored with each rule and the result of the equation (often called pattern precision, see [Ravichandran and Hovy, 2002] as well) can be used during retrieval stage, for example to exclude unreliable rules or to give more reliable rules precedence over less reliable ones. Pattern precision is defined as:

$$p = \frac{No\_correct\_answers}{No\_correct\_answers + No\_incorrect\_answers} \tag{5.2}$$

How rule evaluation is carried out in our setup can be seen as pseudocode in Figure 5.3.

The pseudocode misses some of the finer points, therefore the algorithm will be explained in more detail (in the following numbers in brackets refer back to lines in the pseudocode): First, a set of questions from the QASP corpus is matched against the question patterns (2,3). Usually, the same set of questions that were used during rule creation are used, thereby ensuring that the same questions and therefore the same patterns that were used during rule creation are re-visited. Then, for each question, the AQUAINT corpus is searched for paragraphs likely to contain potential answer sentences (4). This is done by using Lucene [Hatcher and Gospodnetić, 2004] and the Snowball stemmer [Porter, 2001], which were set up to index all of the AQUAINT corpus, treating each paragraph of each article as a *Lucene document*. As a result, Lucene, when queried, will return paragraphs, not articles. The search query is created by using all the question terms that were used during rule creation (For the “Who is Tom Cruise married to?” example this would be “Tom” “Cruise” and “married”) and additionally all known answers to the question contained in the QASP corpus (here

```
INPUT: rule set rs, question set qs
FOR each question q in qs
  MATCH questions against patterns
  RETRIEVE top 100 paragraphs ps relevant for q with Lucene
  PARSE each p in ps
  FOR each rule r in rs that matches question q
    FOR each paragraph p in ps
      IDENTIFY question constituents cs from q in p
      IF all cs which have a path in r are found THEN
        IDENTIFY paths from r in p
        IF all paths are found in p THEN
          IDENTIFY answer candidate node
          EVALUATE answer candidate node
          IF answer candidate node is valid
            INCREASE variable correct in r by 1
          ELSE
            INCREASE variable incorrect in r by 1
          END IF
        END IF
      END IF
    END FOR
  END FOR
END FOR
RETURN rs
```

Figure 5.3: Pseudocode for the rule evaluation algorithm.

“Nicole Kidman” and “Ms. Kidman”). These are combined to a final query of the form:

```
Tom Cruise married ("Ms. Kidman" OR "Nicole Kidman")
```

The reason for including answers in the search query is that there are in general not that many valid answers sentences to TREC questions present in the corpus, and we want to make sure to find the few ones which are there. For the TREC 2002-2006 data, on average, there are 4.33 known answers in the AQUAINT corpus (see Section 3.3.5. Note that there of course are also unknown answer instances in the corpus–it is difficult to estimate how many.) Furthermore, Section 5.7.1.3 will show that, when using only question key words, in more than 20% of the cases none of the returned paragraphs contains both a valid answer and at least one question key word. Evaluatig rules with a set of sentences that contains only a few or even zero valid answer sentence examples will not enable the system to determine which of the acquired rules are good rules. Instead, the system will only able to determine which rules are particularily bad. We can expext a more balanced proportion of valid and invalid answer sentences if we include the answer in the query string. On the one hand, most of the positive examples in the corpus should be found with this method. On the other hand, because we are retrieving 100 paragraphs and for the vast majority there are only a handful of positive examples contained in the corpus in the first place, we will also inevitably retrieve many negative examples as well.

Answer terms in the query are combined using the OR operator, because different answers for one and the same question might exist, and in such a case, is is sufficient if the returned documents contain one of these possibilities. (The question, “Who is the governor of Colorado?” for example has two valid answers in the AQUAINT corpus, “Roy Romer” and “Bill Owens”. Searching for documents that contain both terms would narrow the search space unnecessarily.)

The top 100 paragraphs returned by Lucene are parsed, again using the Stanford-Parser (5). Note that the paragraphs contained in the AQUAINT corpus are in the majority of all cases rather short, usually comprising of one to three sentences–although a few much longer exceptions exist. Now, all rules are found whose antecedent exhibits the same pattern which matched the question currently being processed (6). After that, constituents from all paragraphs are aligned to question constituents in the exact same way as for the sentences during rule creation (described earlier in Sections 5.3 and 5.4) (7,8). As during rule creation, some question constituents might not be found.

Those paths in these rules that are not *null* are searched for in the paragraphs’ parse trees (9). In order to do this, each question constituent that has been identified in the paragraphs is used as a starting point and the paths in the rules are followed from there. If one path specified in the rule can be found, it will lead to a target node (10). If all paths in one rule lead to the same target node (11), this node is identified as an answer candidate node (12).

It is then checked whether this node’s surface structure (in a dependency tree, each node is a leaf node and thus represents one word) is in some morphological form present in the answer strings associated with the question in the QASP corpus (13). For each rule, the results of this process are stored together with that rule in the following way: Each time, if for one specific rule all paths are found, and if they point to the same node and if that node shows a surface present in one of the known answers to the question (14), a variable named *correct* is increased by one (15). Each time, if for one specific rule all paths are found, and if they point to the same node and if that node does not show a surface present in one of the known answers (16), a variable named *incorrect* is increased by one (17). [6]

After the evaluation process is finished the example rule given earlier now looks like this:

Rule:

Pattern:	Who[1]+is[2]+NP[3]+married[4]+to[5]
Path 3:	⇓conj
Path 4:	null
correct:	8
incorrect:	13

After all rules have been created and evaluated, they are stored and can be used to identify answers.

5.6 Rule Execution

The rule execution algorithm is very similar to the rule evaluation algorithm. After all, both algorithms identify candidate answers in unseen answer sentences. When consulting Figure 5.3, which gives pseudocode for the rule evaluation algorithm, we find that the rule execution algorithm works in the same way except in the section between lines 13 to 18. We will now detail with what these lines are replaced in the answer extraction step.

In step 12 in Figure 5.3 an answer candidate node was identified. In the rule execution step it is necessary is to determine the boundaries of the answer. (We need to return an answer string, we cannot make do with an answer node.) This however, is not always trivial. Consider the question “Which office did Bill Clinton assume in 1993”? If the correct answer node could be located, it will have the surface “president”, as, for

[6] Of course, this evaluation procedure is only an approximation. It cannot be 100% guaranteed that all decisions on whether a correct answer has been found or not are correct. Nevertheless, because answers are usuallly very short (Section 4.3.5 earlier in this thesis reports average answer length of 1.83 words), it is highly unlikely that if the node’s surface structure occurs in the answer as well, the node is not part of a correct answer string.

example, node four in the parse of the sentence “Bill Clinton became president of the United States in 1993.” below.

```
1:  Bill       (Bill,NNP,2)         [nn]
2:  Clinton    (Clinton,NNP,4)      [nsubj]
3:  became     (become,VBD,4)       [cop]
4:  president  (president,NN,null) [ROOT]
5:  of         (of,IN,4)            [prep]
6:  the        (the,DT,8)           [det]
7:  United     (United,NNP,8)       [nn]
8:  States     (States,NNPS,5)      [pobj]
9:  in         (in,IN,4)            [prep]
10: 1993       (1993,CD,9)          [pobj]
```

But what exactly constitutes the answer? “president”, “president of the United States” or “president of the United States in 1993”. In order to receive the correct result “president of the United States”, the first PP “of the United States” has to be included in the answer string, but not the second PP “in 1993”. We use a heuristic that for every node that is a child of the answer node decides whether this node should become part of the answer and whether its children should be included or excluded straightaway. This heuristic is mainly based on dependency categories and POS tags. Direct children with NNP and NNPs tags are always included, dependency categories indicating the start of subordinate clauses (e.g. “rcmod”) are indicators to not include a node and all of its children. PPs are usually excluded, unless they contain (beside stop words), only “NNP” or related tags. While this heuristic is not perfect, it returns correct results in the vast majority of all cases.

After all answer candidate strings have been determined, each one receives a confidence value that is equal to the precision value of the rule from which it derived (see also Section 5.5):

$$p = \frac{No\_correct\_answers}{No\_correct\_answers + No\_incorrect\_answers}$$

If we receive the same answer more than once, their individual confidence values are added. (So if we would have one answer candidate c_1 resulting from one successful application with a rule precision of 0.5, and another answer candidate c_2 resulting from three successful rule applications, each with a precision of 0.25, these latter values are added up: $0.25 + 0.25 + 0.25 = 0.75$. Thus answer candidate c_2 would be ranked higher than c_1. This procedure is the same as described in Section 2.2.4.) We now have a ranked list of answer candidates, where the ranking is based on a) the precision of the rules that originated the candidate, and b) the number of rules that were involved in originating the candidate.

Finally, answer candidates are checked on their correct semantic type. We use the type checking mechanism of the QuALiM system, as described in Section 2.2.5 to do this. Note that this step often is necessary to determine the correct answer–an approach solely based on syntax alone is not suitable to answer a subset of questions. As an example consider When- and Where-questions, for whom the answer often is contained in a PP adjunct:

Questions:

- When was Franz Kafka born?
- Where was Franz Kafka born?

Answer sentences:

- Franz Kafka was born in 1883.
- Franz Kafka was born in Prague.

The parser output for the first answer sentence is:

```
1:  Franz (Franz,NNP,2)[nn]
2:  Kafka (Kafka,NNP,4)[nsubjpass]
3:  was (be,VBD,4)[auxpass]
4:  born (bear,VBN,null)[ROOT]
5:  in (in,IN,4)[prep]
6:  1883 (1883,CD,5)[pobj]
```

This results in the following rule:

Rule:

Pattern: When[1]+was[2]+NP[3]+born[4]

Path 3: ⇑nsubjpass⇓prep⇓pobj

Path 4: ⇓prep⇓pobj

The parser output for the second answer sentence is very similar to the parser output for the first sentence given above, it only differs in node six. As a result, we would get a rule with the exact same paths for the second of the above questions combined with the second answer sentence. Thus, both rules would match both answer sentences and therefore the answer candidates "Prague" and "1883" would be returned for both questions. Checking that the answer candidate is of the correct semantic type is a simple way to eliminate this source of wrong answer candidates. (See also Section 3.4, where this point already had been raised.)

After the confidence values of the answer candidates have been modified by the semantic type checking module, the candidate with the highest value is returned as the final answer.

```
1   INPUT: question q, rule set rs
2*  CREATE empty answer bag b
3   MATCH question q against patterns
4   RETRIEVE top 100 paragraphs ps relevant for q with Lucene
5   PARSE each p in ps
6   FOR each rule r in rs that matches question q
7     FOR each paragraph p in ps
8       IDENTIFY question constituents cs from q in p
9       IF all cs which have a path in r are found THEN
10        IDENTIFY paths from r in p
11        IF all paths are found in p THEN
12          IDENTIFY answer candidate node
13*         DETERMINE answer candidate boundaries
14*         PUT answer candidate in answer bag b
15          END IF
16        END IF
17      END IF
18    END FOR
19  END FOR
20* CHECK all candidates in b on correct answer type
21* RETURN answer candidate in b with highest weight
```

Figure 5.4: Pseudocode for the rule execution algorithm.

Pseudocode for the rule execution algorithm can be seen in Figure 5.4. Other than

Figure 5.3 here the input is just one question, not a list of questions. For this reason Figure 5.4 misses one loop. Otherwise, both pseudocodes are very similar. Differences in Figure 5.4 from Figure 5.3 are marked with an asterisk.

5.7 Experiments and System Evaluation

We now provide an evaluation of the algorithm described earlier in this chapter. The key questions we are interested in are the following:

Base performance What is the general performance of the algorithm? This is evaluated using three different evaluation sets.

Effect of semantic alignment How does Semantic Word Alignment (see Section 5.3) affect the results? The system is evaluated once with the corresponding module turned on and once with it switched off.

Comparison with the methods based on lexical resources How does the algorithm described in this chapter compare to the methods described in Chapter 3?

Baseline Performance How do our methods perform when compared to a method that extracts dependency paths from the question?

Effect of corpus size What is the effect of the corpus size on performance? Can we estimate how performance would change with a larger corpus? We evaluate the system with less training data, determine performance increase when increasing the size of the training set and estimate how performance would further increase with even bigger training sets.

However, before these questions can be answered, the evaluation setup needs to be described.

5.7.1 Evaluation Setup

5.7.1.1 TREC question sets

We use the factoid questions in the TREC QA test sets from 2002 to 2006 for evaluation. Question series, as used from 2004 on are manually resolved. (This has been discussed in more detail in Section 3.6.1 of this thesis.) Furthermore, we only used those questions for which a answer sentence tagged with “1” exists in the QASP corpus. For these questions we can be sure that there is a valid answer sentence present in the corpus. (TREC deliberately includes some questions each year to which no answer can be found in the AQUAINT corpus. For these questions, a QA system participating in TREC is supposed to return “NIL” instead of an answer, indicating that the system was able to identify that the document collection contains no answer.)

5.7.1.2 Cross Validation

In order to evaluate performance, we adopt a cross validation approach. The fact that the QASP corpus is already segmented into six parts makes this a natural choice. Five of the six QASP files contain data based on TREC judgment files from 2002 to 2006, whereas the sixth file is based on [Lin and Katz, 2005]. One crucial detail that needs to be considered here is that the dataset based on [Lin and Katz, 2005] itself is based on TREC’s 2002 data: All questions in [Lin and Katz, 2005]’s data set are also part of TREC’s 2002 question set. Thus training on the Lin dataset and testing on TREC02 and via-versa is not an option. Taking this into account we arrive at a five-fold evaluation procedure, as can be seen in Table 5.5.

Fold	Training Sets	Test Set
1	TREC03, TREC04, TREC05, TREC06	TREC02
2	TREC02, TREC04, TREC05, TREC06, Lin02	TREC03
3	TREC02, TREC03, TREC05, TREC06, Lin02	TREC04
4	TREC02, TREC03, TREC04, TREC06, Lin02	TREC05
5	TREC02, TREC03, TREC04, TREC05, Lin02	TREC06

Table 5.5: Splits into training and tests sets of the QASP data set used for evaluation.

5.7.1.3 Evaluation Sets

In order to evaluate the created rules we need a set of sentences to which they can be applied. In a complete QA system, with a traditional architecture (see [Prager, 2006] and [Voorhees, 2003], for descriptions of such architectures), the document or passage retrieval step performs this function. This step is crucial to a QA system’s performance, because it is impossible to locate answers in the subsequent answer extraction step if the passages returned during passage retrieval do not contain any occurrences of the answer. The problem we are facing in our setup is similar: It cannot be expected for our rules to return a correct answer, if none of the sentences used as input contain the correct answer. We therefore decided to use three different evaluation sets to test the rules:

1. The first set contains for each question the top 100 paragraphs returned by Lucene when using simple queries made up from the question's key words. It cannot be guaranteed that answers to every question are present in this test set.

2. The second set contains the same data as the first set, but all known, valid answer sentences for the question from the QASP corpus are added to the top 100 paragraphs returned by Lucene. This is done to ensure that at least some sentences containing the correct answer are present in the evaluation set.

3. The third set is similar to the second, but the IR search query used includes all known correct answers to the question. This further increases the chance that the evaluation set actually contains valid answer sentences.

Let us take a look at these sets in more detail:

The Lucene query to create the first evaluation set is based on words in the question that are not stop words or wh-words. Each such word is added to query. No sophisticated methods like combining phrases from the question into quoted search phrases or even query expansion are used. Thus, for the question "Who is Tom Cruise married to?", the query is:

```
Tom Cruise married
```

For the second evaluation set we use the exact same method and retain all 100 paragraphs from the first evaluation set. Additionally, all known answer sentences to the current question from the QASP corpus files based on TREC 2002-2006 data are included, should they not already be present in the test set. As reported in Section 4.3.5 earlier in this thesis, on average this part of the QASP corpus contains 4.33 known answer sentences per question. We did not add sentences from the QASP data based on Lin et al. The reason for this is that for some questions it contains more than 100 answer sentences and therefore we felt it would have distorted the balance between valid and invalid candidate sentences in the test set. Note that, because of the way the data set is split into training and test sets, it never is the case that we add answer sentences that were used during rule creation and evaluation.

The third evaluation set is similar to the second, except for the query that was used to retrieve the top 100 paragraphs. This time, all known answers contained in the QASP corpus for one particular question were added to the query–in brackets and separated by the `OR` operator. Thus, for the question “Who is Tom Cruise married to?” and the two answers present in the QASP files “Nicole Kidman” and “Ms. Kidman”, the final query is of the form:

```
Tom Cruise married ("Ms. Kidman" OR "Nicole Kidman")
```

This way of creating the queries is is the same as used during rule evaluation, see Section 5.5. All known answer sentences to the current question from the QASP corpus files based on TREC 2002-2006 data are also included, should they not already have been picked up by Lucene and therefore already be present in the test set (just as in evaluation set 2). Note that for all experiments described in this chapter neither the score returned by the IR module nor the IR rank are used in further processing steps.

In order to provide a quantitative characterization of the three described evaluation sets we ran a check on each of them in order to approximate the number of correct answer sentences they contain. For each paragraph it was automatically determined whether it contained one of the known answer strings and a minimum of one of the question word. (These are also the minimum requirements for this approach to locate a correct answer: It needs a least one question word to match one of the rules, and, in order to return a correct answer, it of course has to be present in the sentence.)

Test	**Number of Correct Answer Sentences (approximation)**										Mean	Med
set	= 0	<= 1	<= 3	<= 5	<= 10	<= 25	<= 50	>= 75	>= 90	>= 100		
2002	0.203	0.396	0.580	0.671	0.809	0.935	0.984	0.0	0.0	0.0	6.86	2.0
2003	0.249	0.429	0.627	0.732	0.828	0.955	0.997	0.003	0.003	0.0	5.67	2.0
2004	0.221	0.368	0.539	0.637	0.799	0.936	0.985	0.0	0.0	0.0	6.51	3.0
2005	0.245	0.404	0.574	0.665	0.777	0.912	0.987	0.0	0.0	0.0	7.56	2.0
2006	0.241	0.389	0.568	0.665	0.807	0.920	0.966	0.006	0.0	0.0	8.04	3.0

Table 5.6: Fraction of sentences that contain correct answers in Evaluation Set 1 (approximation).

Tables 5.6 to 5.8 shows for each part of each evaluation set how many answers on average it contains per question. The column “= 0” for example shows the fraction of questions for which no valid answer sentence (as determined by the approximation method just described) is contained in the evaluation set, column “<= 5” lists the fraction of questions with five or fewer answer sentences, column “>= 90” gives the

Test	Number of Correct Answer Sentences (approximation)										Mean	Med
set	= 0	<= 1	<= 3	<= 5	<= 10	<= 25	<= 50	>= 75	>= 90	>= 100		
2002	0.0	0.177	0.385	0.515	0.699	0.904	0.965	0.002	0.0	0.0	10.20	5.0
2003	0.0	0.234	0.429	0.565	0.757	0.924	0.997	0.003	0.003	0.003	8.60	5.0
2004	0.0	0.216	0.397	0.534	0.686	0.916	0.980	0.0	0.0	0.0	9.29	5.0
2005	0.0	0.251	0.429	0.545	0.667	0.890	0.962	0.006	0.0	0.0	10.48	4.0
2006	0.0	0.207	0.412	0.534	0.741	0.895	0.955	0.017	0.003	0.0	10.65	5.0

Table 5.7: Fraction of sentences that contain correct answers in Evaluation Set 2 (approximation).

Test	Number of Correct Answer Sentences (approximation)										Mean	Med
set	= 0	<= 1	<= 3	<= 5	<= 10	<= 25	<= 50	>= 75	>= 90	>= 100		
2002	0.0	0.074	0.158	0.235	0.342	0.561	0.748	0.172	0.116	0.060	33.46	21.0
2003	0.0	0.099	0.203	0.254	0.356	0.573	0.720	0.161	0.090	0.031	32.88	19.0
2004	0.0	0.073	0.137	0.211	0.328	0.598	0.779	0.142	0.069	0.034	30.82	20.0
2005	0.0	0.163	0.238	0.279	0.410	0.589	0.759	0.141	0.097	0.069	30.87	17.0
2006	0.0	0.125	0.207	0.281	0.415	0.596	0.727	0.173	0.122	0.088	32.93	17.5

Table 5.8: Fraction of sentences that contain correct answers in Evaluation Set 3 (approximation).

fraction of questions with 90 or more valid answer sentences. The last two columns show mean and median values.

Note that these values give upper bound approximations. A candidate might contain the correct answer and one question keyword and still not be a valid answer sentence. While this is rather unlikely for most questions, for a few questions, especially those with answers that are very frequent words or phrases in the AQUAINT corpus, this very well might be the case. Consider for example the question “Where is the United Nations headquarters located?” and the answer “New York”. A sentence might easily contain both “New York” and “United”, as for example “Franklin D. Roosevelt was born in New York, United States of America.” does, but still not be a valid answer sentence to this question.

5.7.2 Base Performance

For this set of experiments we created and evaluated a set of rules as described in Sections 5.4 and 5.5. Word Alignment was simply based on morphological similarity. No semantic alignments as described in Section 5.3 were used.

Results for all three evaluation sets are presented in Tables 5.9, 5.10 and 5.11. In all three tables the first column lists the test set used (see Table 5.5 for the corresponding training sets). Column two indicates the number of questions in each test set, column three the number of questions for which a minimum of one rule exists. Note that even very unreliable rules are counted here. Column four indicates for how many an-

Test set	Question number	Question with rules	Min one correct	Overall correct	Accuracy overall	Accuracy if ∃ rule
2002	429	321	141	48	0.112	0.150
2003	354	237	72	21	0.059	0.089
2004	204	142	69	25	0.126	0.176
2005	319	214	92	44	0.138	0.206
2006	352	208	78	29	0.082	0.139
Sum	1658	1122	452	167	0.101	0.149

Table 5.9: Performance based on evaluation set 1.

Test set	Question number	Question with rules	Min one correct	Overall correct	Accuracy overall	Accuracy if ∃ rule
2002	429	321	181	70	0.163	0.218
2003	354	237	104	45	0.127	0.190
2004	204	142	87	40	0.196	0.281
2005	319	214	122	63	0.197	0.294
2006	352	208	101	43	0.122	0.206
Sum	1658	1122	595	261	0.157	0.233

Table 5.10: Performance based on evaluation set 2.

swers a minimum of one rule (out of potentially many) returned a correct result. (Note that there might also have been rules, potentially even with a higher confidence value, which returned wrong results.) Column five indicates how often the overall answer returned was correct. Column six indicates the top-1 accuracy the system achieves, when all questions in the data set are taken into account (column five divided by column two). Column seven indicates top-1 accuracy when computed only for questions for which a minimum of one rule exists (column five divided by column three).

As can be seen, results improve considerably when the answer is part of the IR query, which was to be expected. Note that it should be possible to improve the results presented in Table 5.9 by constructing more sophisticated search queries (e.g. by combining phrases into quoted search phrases or by using query expansion).

Test set	Question number	Question with rules	Min one correct	Overall correct	Accuracy overall	Accuracy if ∃ rule
2002	429	321	228	127	0.296	0.396
2003	354	237	141	83	0.234	0.350
2004	204	142	111	62	0.304	0.437
2005	319	214	153	88	0.276	0.411
2006	352	208	128	79	0.224	0.380
Sum	1658	1122	761	439	0.265	0.391

Table 5.11: Performance based on evaluation set 3.

5.7.3 Effect of Semantic Alignment

For the figures presented so far no semantic alignment (Section 5.3) was used. Table 5.12 presents results based on evaluation set 1 when using semantic alignment.

Test set	Question number	Question with rules	Min one correct	Overall correct	Accuracy overall	Accuracy if ∃ rule
2002	429	321	147	50	0.117	0.156
2003	354	237	76	22	0.062	0.093
2004	204	142	74	26	0.127	0.183
2005	319	214	97	46	0.144	0.215
2006	352	208	85	31	0.088	0.149
Sum	1658	1122	452	176	0.106	0.156

Table 5.12: Performance with Semantic Alignment based on evaluation set 1.

As can be seen, overall, when taking all test sets into account, the number of correct answers returned increases from 167 to 175 for evaluation set 1, that is 4.7%.

5.7.4 Baseline Performance

As a baseline to compare our algorithm against we repeated the same experiment, except for one difference. During the rule creation step, we acquired the dependency paths contained in the rules not from answer sentences, but from the questions directly. The question word (e.g. “Who”) or question question phrase (e.g. “Which city”) was taken as the place where the answer is supposed to be located. The reasoning behind this is that each question illustrates one way how an answer sentence can be

Test set	Question number	Question with rules	Min one correct	Overall correct	Accuracy overall	Accuracy if ∃ rule
2002	429	321	43	14	0.033	0.044
2003	354	237	28	10	0.028	0.042
2004	204	142	19	6	0.029	0.042
2005	319	214	21	7	0.022	0.033
2006	352	208	20	7	0.020	0.034
Sum	1658	1122	131	44	0.027	0.039

Table 5.13: Performance of the baseline method which extracts dependency paths from questions. Figures are based on evaluation set 1.

formulated. This basic observation has been used in many QA systems to date, e.g. [Attardi et al., 2001], [Katz and Lin, 2003] or [Bouma et al., 2005a]. Results can be seen in Table 5.12. As can be seen performance is much lower. (0.039 accuracy compared to 0.156 for all question sets, based on evaluation set 1, when rules are available provides a 300% increase in performance.) Of course, the baseline used is very simple. It does not make use of none of the enhancements commonly used in other work, see Section 5.2.1, for example equivialence rules or fuzzy matching. However, our method using the answer sentences from the QASP corpus uses none of these enhancements either. We can expect that employing one of these enhancements leads to an increase in performance of both the baseline method and of the method based on the QASP corpus detailed in this chapter.

There are two main reasons for this:

1. In the QASP corpus, on average there are more than four answer sentences per question, thus when learning from answer sentences we have more data available. [7]

2. Questions show a different syntax than answer sentences. For some question classes the syntax of English questions differs considerably than those of declarative sentences. Also, answer sentences tend to be longer and therefore show a more complex syntactic structure.

Both these points come as no surprise. These observations in fact were part of the motivations for the approach described in this chapter, see Sections 5.1 and 5.2.1.

[7] Note that this is much more than four times as much, because the algorithm is based on rules for question classes, not for individual questions.

5.7.5 Comparison with Methods based on Lexical Resources

One reason to develop the approach laid out in this chapter was that the methods based on lexical resources described in Chapter 3, showed poor performance when evaluated on the AQUAINT corpus. In Section 3.6 we reported the following figures (top-1 accuracy): On all (complete) test sets from 2002 to 2006, for method 1, FrameNet alone obtains a score of 0.030, PropBank alone 0.028, VerbNet alone 0.027. All resources combined obtain a score of 0.032, this rises to 0.036 when also using the automatic coverage expansion strategy. For method 2 we receive scores of 0.015 and 0.044 for FrameNet and PropBank respectively, 0.049 when both resources are combined. The combination of both methods results in a score of 0.061.

When comparing these figures with the method detailed in this chapter, we face the problem that the method described here is an answer extraction strategy only, not a complete QA strategy (which is the case for the first method based on lexical resources, see Section 3.4). But even if we use the results obtained with evaluation set 1, which uses a very simple paragraph retrieval module, we obtain better values for the strategy described in this chapter: 0.101, with semantic alignment 0.106. (A relative increase of 74% when compared to both methods based on lexical resources combined.) We can expect much better performance with a better paragraph retrieval module, as the results for evaluation sets 2 and 3 show. In this context it should also be mentioned that the number of sentences in the QASP corpus is below 9,000; This is much lower than the data in PropBank and FrameNet, which both list more than 100,000 sentences.

5.7.6 Performance Impact on a pre-existing QA System

In Section 3.6.5 of this thesis we evaluated what performance increase the methods based on lexical resources bring to a pre-existing QA system. In a similar manner we are now interested in finding out how much the method described in this chapter improves performance of a pre-existing QA system. The system we have available to this end is the QuALiM system (see Chapter 2) which was also used in Section 3.6.5. It should be noted however, that this time there are, other than in Section 3.6.5, a few reasons that put the baseline system at an advantage over the approach at hand:

- The baseline system, QuALiM, is web-based. The methods detailed in this chapter on the contrary are developed to work on a small local corpus. Generally, we can expect that web based QA produces better results than QA based on local corpora (due to, for example, redundancy).

- The approach described in this chapter is an answer extraction approach only. Yet, in order to compare it to the baseline, we need a complete system which includes an information retrieval module. We use the same retrieval approach which we used for evaluation set 1, described in Section 5.7.1.3. As can be seen there, in more than 20% of the cases, this method returns not a single paragraph that contains both the answer and at least one question keyword.

Test Set	QuALiM	QASP	combined	increase
2002	0.503	0.117	0.524	5.2%
2003	0.367	0.062	0.390	6.2%
2004	0.426	0.127	0.451	5.7%
2005	0.373	0.144	0.389	4.2%
2006	0.341	0.088	0.358	5.0%
02-06	0.405	0.106	0.425	4.9%

Table 5.14: Evaluation results of the QuALiM system on its own and when combined with the QA appoach based on QASPs. Top-1 accuracy based on all questions with known answers in TREC's test sets.

In Table 5.14 results are shown when the QA method detailed in this chapter is combined with the baseline system QuALiM. (See Section 2.2.4 for explanations about how algorithm results are combined.) The first column indicates the question set used. The second column gives top-1 accuracy figures for QuALiM's performance. Column four shows performance of the method detailed in this chapter, based on evaluation set 1, with semantic alignment (see Section 5.7.3). Column five shows top-1 accuracy when both methods are combined, whereas the last column shows the relative increase obtained, when comparing the combined run with the run based on QuALiM alone. As can be seen in the last row, overall, for all test sets combined we achieve a 4.9% increase in top-1 accuracy.

5.7.7 Effect of Corpus Size

This section aims to assess the effects of corpus size on performance. We are mainly interested in the impact the size of the corpus reserved for training purposes makes. Tables 5.9 to 5.11 presented earlier in this chapter show that an average of 32.2% of the questions have no matching rules. This is because the data used for training contained no examples for a significant subset of question classes. It can be expected that, if more training data would be available, performance would increase. Thus, ideally, we would like to repeat the described experiments with more training data. We would expect to find additional instances of question classes and also expect the number of answer sentences for already known question classes to increase. Unfortunately, for the described experiments, all data available already has been used, thus increasing the amount of training data is not an option we can easily take. What however can be done is to repeat the experiments with **less** training data. We then can gradually add more training data and see how this affects results. From that, it should be possible to see whether we can expect additional training data to further improve the results.

Test	Training	Q	QASP	Questions	Overall	Accuracy	Accuracy
set	sets	No.	No.	with rules	correct	overall	if ∃ rule
2002	03	354	1352	250	31	0.072	0.124
2002	03, 04	558	2178	284	38	0.089	0.133
2002	03, 04, 05	877	3406	311	43	0.100	0.138
2002	03, 04, 05, 06	1229	4565	321	48	0.112	0.150

Table 5.15: Performance increase when training set size is increased. Based on 2002 data used as test set.

Test	Training	Q	QASP	Questions	Overall	Accuracy	Accuracy
set	sets	No.	No.	with rules	correct	overall	if ∃ rule
2003	02	429	1833	187	17	0.048	0.091
2003	02, Lin	429	2961	192	18	0.051	0.094
2003	02, Lin, 04	633	3787	212	19	0.054	0.090
2003	02, Lin, 04, 05	952	5015	224	19	0.054	0.085
2003	02, Lin, 04, 05, 06	1304	6174	237	21	0.059	0.089

Table 5.16: Performance increase when training set size is increased. Based on 2003 data used as test set.

Results of this process are shown in Tables 5.15 to 5.19. They are based on evaluation set 1. For each available test set, the experiment outlined in Section 5.7 was repeated, but this time initially with only one QASP file as training data. At each subsequent iteration, one additional file was added.

As can be seen, overall accuracy improves in 16 out of 19 times when a new QASP file is added to the training set; there are two occasions where accuracy stays the same and once it drops (when adding the 2004 training set to the experiment based on the

Test set	Training sets	Q No.	QASP No.	Questions with rules	Overall correct	Accuracy overall	Accuracy if ∃ rule
2004	02	429	1833	107	18	0.088	0.168
2004	02, Lin	429	2961	110	19	0.093	0.173
2004	02, Lin, 03	783	4313	124	20	0.098	0.161
2004	02, Lin, 03, 05	1102	5541	138	24	0.118	0.174
2004	02, Lin, 03, 05, 06	1454	6700	142	25	0.123	0.176

Table 5.17: Performance increase when training set size is increased. Based on 2004 data used as test set.

Test set	Training sets	Q No.	QASP No.	Questions with rules	Overall correct	Accuracy overall	Accuracy if ∃ rule
2005	02	429	1833	165	26	0.082	0.158
2005	02, Lin	429	2961	176	26	0.082	0.148
2005	02, Lin, 03	783	4313	186	32	0.100	0.172
2005	02, Lin, 03, 04	987	5139	198	38	0.119	0.192
2005	02, Lin, 03, 04, 06	1339	6298	214	44	0.138	0.206

Table 5.18: Performance increase when training set size is increased. Based on 2005 data used as test set.

2006 test set).

Figure 5.5 presents this data in graphical form. The x-axis shows the number of Question Answer Sentence Pairs that were used as training data (row four in Tables 5.15 to 5.19), whereas on the y-axis we can see overall accuracy (row seven in Tables 5.15 to 5.19). Each colour represents a different test set (see legend).

Test set	Training sets	Q No.	QASP No.	Questions with rules	Overall correct	Accuracy overall	Accuracy if ∃ rule
2006	02	429	1833	166	20	0.057	0.120
2006	02, Lin	429	2961	169	22	0.063	0.130
2006	02, Lin, 03	783	4313	188	24	0.068	0.127
2006	02, Lin, 03, 04	987	5139	193	23	0.065	0.120
2006	02, Lin, 03, 04, 05	1306	6367	208	29	0.082	0.140

Table 5.19: Performance increase when training set size is increased. Based on 2006 data used as test set.

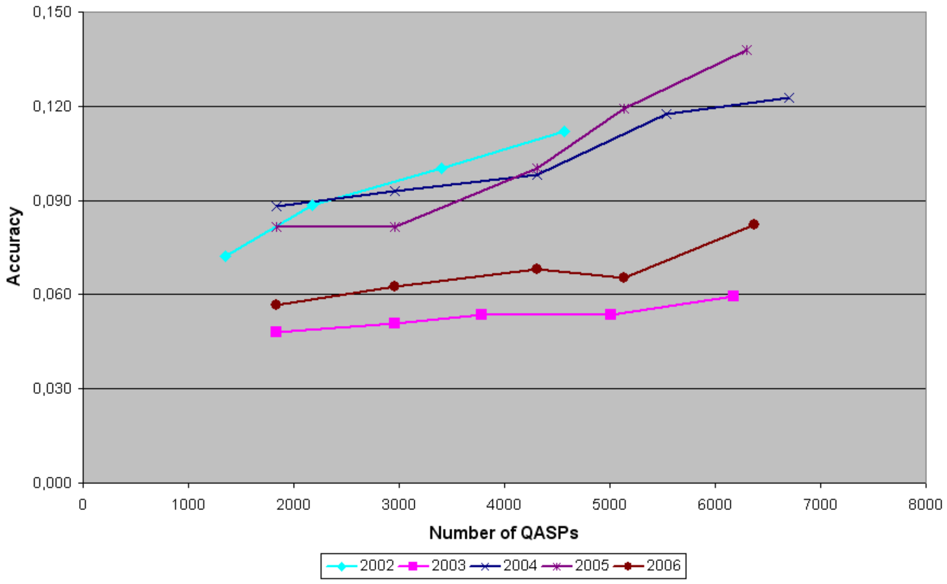


Figure 5.5: Effect of the amount of training data on system performance

5.8 Conlusions

We described a method how, by utilizing dependency paths, syntactic relations from a corpus of Question Answer Sentence Pairs can be aquired. These can than be used to process a set of candidate sentences with the aim of deciding for each sentence whether it contains the answer and, if this is the case, to extract the answer. The method was developed to work on a small local corpus. This was done because the methods based on lexical resources presented in Chapter 3, although performing well on the web, could not be successfully applied to a local corpus. The main reason for this, beside redundancy, we argued is that small, journalistic corpora (like the AQUAINT corpus) show many sentences exemplifying *indirect evidence*. The methods in Chapter 3 are suitable to recognize stict semantic similarity, or *direct evidence*, but not *indirect evidence*, and thus failed on the AQUAINT corpus. The methos presented in this chapter acquire syntactic knowlege from a corpus containing many examples of indirect evidence and as such can be assumed to better identify form of indirect evidence. That this indeed is the case was shown in the evaluation section. The algorithm based on dependency paths outperforms the methods from Chapter 3 on standard TREC test sets. Additionally, it outperforms a baseline which uses dependency relations extracted from the questions.

The algorithm described here, however, is an algorithm for answer extraction only; in order to work as a full-fleged QA system, a paragraph or sentence retrieval module is needed as well. The implementation we used for our experiments is very simple and hinders performance. While this certainly is a shortcomming of the current implmentation and therefore an item on the future work list, it is not directly related to what we are interested in in this chapter. (One possibility to solve this problem would be to parse the whole AQUAINT corpus off-line, and devise an IR module that indexes dependency relations beside the surface strings of the words. During retrieval stage we could only extract phrases between which certain dependency relations exist. Approaches like this have already been undertaken, see for example [Tiedemann, 2005].)

Chapter 6

Conclusions

In this chapter we summarise this thesis and its main findings. We then address questions that remain unsolved and which provide topics for further research.

6.1 Main Findings

This thesis is concerned with discovering new ways of how to deal with paraphrases in QA. Paraphrases involving lexical variation between questions and answer sentences (or passages) have already been examined many times, for example by using WordNet, e.g. in [Harabagiu et al., 2001]. In this thesis we shift the focus to paraphrases involving grammatical variations. We argue that such paraphrases can be obtained from corpora, more specifically from lexical resources like FrameNet [Baker et al., 1998] and PropBank [Palmer et al., 2005] or from a corpus of Question Answer Sentence Pairs (QASPs).

To this end we developed, in Chapter 3, an approach based on the mentioned lexical resources FrameNet and PropBank (and also VerbNet [Schuler, 2005]) all of which comprise information about semantic and syntactic combinatory possibilities (valences) of words. FrameNet and PropBank both contain more than 100,000 annotated sentences, that can be used, for example by a QA system, to recognize different ways in which one and the same fact can be expressed. VerbNet does not contain annotated sentences, but rather contains a symbolic representation of valences. We are able to show that the information in these resources can be used beneficially for Question Answering. This is done in a number of ways (all experiments were carried out on the factoid questions contained in TREC's test sets from 2002-2006):

- When utilizing all three resources we outperform a baseline that only uses syntactic information available in the question by 14.7%, although coverage of the resources is sketchy (see Section 2.6.2).

- Our methods based on these resources improve performance of a state-of-the-art QA system (QuALiM, see Chapter 3) that performed well in TREC evaluations from 2004 to 2006, by 16.3% and 21.9%, depending on whether complete or partial TREC test sets are used (see Section 3.6.5).
- Our methods based on these lexical resources alone achieve a performance that compares well with the best performing systems' scores in the corresponding TREC evaluations, if using partial TREC test sets (see Section 3.6.4).

However, we find that all three resources, at the time of writing, show significant coverage issues. We achieve much better results when evaluating our methods on partial TREC test sets consisting only of questions for which data is available in these resources (accuracy 0.318 for all five used TREC test sets combined) compared to evaluating them on complete TREC test sets (accuracy 0.187), see Section 3.6.4. This strongly suggests that more effort has to be made by the research community to create more complete resources.

Furthermore, when porting our methods from a web-based setting (for which they where developed) to a setting where answers are supposed to be found in a much smaller, local corpus (we used the AQUAINT corpus [Graff, 2002]), results are much worse (accuracy 0.061 on complete test sets), see Section 3.6.6. We argue that this is due to one limitation of these resources: They are suitable to detect *direct evidence*, that is cases where an answer sentence can be said to answer a question in a strict logical way. They are however not suitable to detect forms of *indirect evidence*.

Based on the mentioned evaluation results, we argue that in a web-based setting, where one can expect to find a large number of answer sentences for a given question, it in fact is a desirable strategy to concentrate on sentences that show direct evidence for the question. When working with a small local corpus however, because there are much less answer sentences available we cannot afford to overlook sentence exemplifying *indirect evidence*, thus in such a case we need methods that are able to detect forms of *indirect evidence* (see Section 3.7).

It is furthermore important to note that our experiments concerning FrameNet, PropBank and VerbNet constituted the first analysis of the potential use of these resources for Question Answering. Our work was the first to show the benefit of these resources for Question Answering and the problems that arise when they are used, most notably coverage. The little work that preceded our work does not assess these resources in isolation, but in combination with other methods. Thus their effect on performance is not evaluated in isolation and as a result no conclusions about their usefulness for QA can be made (see Section 3.3).

The observation that our methods based on FrameNet and the like perform well on the web but not on a local corpus led to the creation a corpus of **Q**uestion **A**nswer **S**entence **P**airs (QASPs), described in Chapter 4. This corpus contains more than 8,000 answer sentences to 1,900 factoid TREC questions from 2002 to 2006. As such, it documents the relations between a large number of TREC questions and their answer sentences. We have made the QASP corpus available to the public in April 2008. At the time of writing, there is no other corpus of this type available. In the context of this thesis, the QASP corpus is important because many of the sentences it contains exhibit forms of *indirect evidence*. We also provide a numerical analysis of the QASP corpus in Section 4.4. We concentrate on some selected properties of the questions and answer sentences it contains and the relations between them. Our results provide further evidence for the need of strong paraphrasing capabilities in Question Answering.

In Chapter 5, we employ the QASP corpus in an answer extraction strategy for factoid Question Answering, which acquires syntactic information about potential formulations of answer sentences (syntactic paraphrases) for classes of question from the answer sentences in the QASP corpus. This information is coded by using dependency paths and stored in rules, which can be used to extract answers from unseen candidate sentences. Because this strategy acquires knowledge from the QASP corpus, which contains many questions and answer sentences linked by *indirect evidence*, it is suitable to locate answers in sentences exemplifying such forms of evidence. We evaluate this approach by again using factoid questions from the TREC test sets from 2002-2006. We use the AQUAINT corpus as the document collection where answer have to be found and are able to show that:

- Our approach outperforms a baseline method that acquires syntactic information from questions (as opposed to answer sentences from the corpus) by 300% (see Section 5.7.4).

- Our approach outperforms the methods based on lexical resources on the AQUAINT corpus by 74%, although we use a very basic paragraph retrieval module and much less training data (see Section 5.7.5).

- Performance steadily increases when as a start using only parts of the available data for training and then increasing the training set size gradually. This strongly suggests that performance would further rise, if more training data would be available (see Section 5.7.7).

6.2 Open Questions and Future Work

6.2.1 The Need for More Data

Performance of the algorithms detailed in Chapters 3 and 5 is hindered by the fact that training data is sparse. While this affects performance, it cannot be used as an argument against the validity of the algorithms. In both cases, we evaluated performance on partial TREC test sets only containing questions for which data is available and the results were very encouraging, although we ideally would have had more training instances available per questions. Yet, we think this is the proper way of advancing research in the field. Surely, it would be unreasonable to create massive resources (and spend lots of money while doing so), and then, after they are completed, test whether they are actually useful. It seems much more appropriate to run experiments on a partial data, and to then decide, depending on the outcome, whether to invest in creating bigger resources or not. This is what we did in this thesis and for both kinds of data used (lexical resources like FrameNet, PropBank and VerbNet on the one hand and a corpus of Question Answer Sentence Pairs on the other) we arrived at the conclusion that it would be highly desirable to create more data. In most cases this means employing humans for this task, which is expensive. However, one current, promising line of research in this respect is extending coverage of resources, for example FrameNet and PropBank with automatic means, see for example [Pennacchiotti et al., 2008] and [Gordon and Swanson, 2007], respectively. Another noteworthy line of research utilizes large online, manually-created QA collections on sites like Yahoo! Answers [1] for Question Answering, see for example [Lee et al., 2008] and [Jeon et al., 2005]. Data from such sites could potentially be used to extend the QASP corpus.

6.2.2 What Kind of Data?

While we just argued for the need of more data, a related and probably even more important questions is concerned with what kind of data we ideally need. The resources

[1] http://answers.yahoo.com/

we looked at are suitable to detect different kinds of evidence, namely *direct* and *indirect evidence*. In Natural Language Processing we need both. In a Question Answering system, direct evidence should be preferred whenever possible, because it delivers answers that undoubtfully answer the question. Yet, we cannot rely on it alone, especially when working with small corpora, because in such cases indirect evidence often is the only available form of evidence.

As far as the work concerning lexical resources (and thus direct evidence) in Chapter 3 goes, FrameNet offers unique characteristics that we think show the way forward. It combines lexical and syntactic knowledge in such a way that it can not only be used to determine that “buy” and “sell” are semantically related (like in WordNet) or to look up example usages of these words (like in PropBank), but also to determine how a sentence using the one can be transformed to a sentence using the other by preserving its meaning. This is because in FrameNet different entires are organized in frames between which different kinds of relations exist, which, in theory, enables various kinds of NLP systems to perform wide-ranging, detailed inference. ([Baumgartner and Burchardt, 2004] describe an infrastructure for such a system, but the authors provide no implementation.) However at this point FrameNet coverage still is problematic. Currently, PropBank contains more annotated sentences than FrameNet and WordNet much more links between semantically related words.

While FrameNet and the like are resources that can be used in many subfields of NLP, the QASP corpus is more centered towards Question Answering. (Which does not mean it could potentially also be useful in other subfields, e.g. textual entailment or paraphrase detection.) We could show that the QASP corpus is useful for QA and provides good performance with our algorithm as long as training data is available. We also showed that system performance consistently increases when training the system with only parts of the data in the beginning and then gradually adding more data until using all of it. This strongly suggests that performance would further increase, if more data would be available.

However, we did not evaluate how the acquired rules perform when they are used to identify answers in a corpus other than AQUAINT, for example the BLOG corpus [Macdonald and Ounis, 2006], a test collection of blog data used in TREC since 2007 [Dang et al., 2007]. We would expect blog articles to show different properties than a newspaper articles. This might manifest itself on many levels, e.g. in average sentence length or the sentences’ syntactic complexity or even regularity. Performance of the rules acquired from the AQUAINT corpus might deteriorate when they are applied to such a document collection. To counteract this, we could, when extending the QASP corpus, include new example sentences from corpora that differ in nature, e.g. from the BLOG corpus. Once we have a QASP corpus containing answer sentences from different genres, we can expect it to perform well across genre boundaries.

6.2.3 Non-Strict Matching

Most of the work carried out in this thesis relies on strict matching of questions and answer sentences. A considerable amount of recent work in QA has focused on fuzzy, statistical methods to link questions and answer sentences, e.g. [Punyakanok et al., 2004, Cui et al., 2005, Shen and Klakow, 2006]. There are a lot of trade-offs at work here which have to be considered. Firstly, when employing less strict matching methods one can expect recall to go up, because more correct candidate sentences can be matched, but one also would expect precision to deteriorate, because it is difficult to not have more incorrect candidate sentences matched at the same time. Often the use of fuzzy methods is (directly or indirectly) motivated with a lack of data from which the algorithm can learn. [Punyakanok et al., 2004] for example match dependency structures in questions against answer sentences. As argued before in this thesis (see for example Section 5.2), this is not sufficient to capture many ways in which answer sentences can be formulated. [Punyakanok et al., 2004] employ fuzzy matching to solve this problem, we propose using more and better suited data (e.g. a corpus of answer sentences). Our reasoning is that fuzzy matching is only suitable to identify additional valid answer sentences that are somehow (how depends on the algorithm used) similar to known correct instances. There are not suitable to identify completely new structures, e.g. if a verb shows different syntactic frames. Yet of course it should be beneficial to combine both approaches. We cannot necessarily expect that there will be a point where we have enough high quality training data available that allows us to match all questions with their answer sentences with via strict matching algorithms. Especially for minor variations, fuzzy methods will always be needed to fill the gaps.

6.2.4 Syntactic and/or Semantic Indices

Performance of all methods presented in this thesis was hampered by the fact that the indices we used for Information Retrieval were indices containing (stemmed, non-stop) words. Although the resources we used contain various forms of syntactic and semantic information which we exploited with our algorithms, passage retrieval is a bottleneck because it functions on string level. (In 5.7.1 we have seen that by using standard key-word based retrieval, for more than 20% of all questions the top 100 returned paragraphs contain not a single valid answer sentence.)

To remedy this, in theory, it would be possible to store any kind of syntactic or semantic information in the index. In practice this means that every sentence in the corpus has to be preprocessed with whatever tools' output we would like to store. If we would like to have syntactic information from parse trees, e.g. dependency relations or paths in the index, we would have to parse every sentence. If we would like to store semantic role information, every sentence has to be processed by a role labeler. The same holds for named entity information, coreferences etc. (Some of these approaches have already been implemented, e.g. [Prager et al., 2000] for named entities or [Tiedemann, 2005] for parsing.) Such approaches however require large amounts of computing power. For our experiments, it took more than 120 hours to parse all necessary sentences for the experiments described in Section 5.7 alone (we used the Stanford Parser [Klein and Manning, 2003b]), yet this accounts only for a fraction (less than 1%) of the sentences contained in the AQUAINT corpus. In case of a web-based approach to Question Answering system, (as our approach described in Chapter 3) this becomes even more problematic because of the huge size of textual information available on the web. With the resources available to us at the time of writing, we simple cannot parse the web. Yet better and faster hardware with more storage capabilities has been getting much cheaper in recent years and will continue to do so. Thus it probably is only be a question of time until we can index the web in semantic ways.

Bibliography

[Agichtein and Gravano, 2000] Agichtein, E. and Gravano, S. (2000). Snowball: Extracting Relations from Large Plain-text Collections. In *Proceedings of the 5th ACM International Conference on Digital Libraries*.

[Attardi et al., 2001] Attardi, G., Cisternino, A., Formica, F., Simi, M., and Tommasi, A. (2001). PIQASso: Pisa question answering system. In *Proceedings of the 2001 Edition of the Text REtrieval Conference (TREC-01)*.

[Baker et al., 1998] Baker, C. F., Fillmore, C. J., and Lowe, J. B. (1998). The Berkeley FrameNet Project. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and the 17th International Conference on Computational Linguistics (COLING/ACL-98)*.

[Banerjee and Pedersen, 2003] Banerjee, S. and Pedersen, T. (2003). Extended Gloss Overlaps as a Measure of Semantic Relatedness. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI-03)*.

[Bar-Haim et al., 2006] Bar-Haim, R., Dagan, I., Dolan, B., Ferro, L., Giampiccolo, D., Magnini, B., and Szpektor, I. (2006). The second PASCAL Recognising Textual Entailment Challenge. In *Proceedings of the Second PASCAL Challenges Workshop*.

[Baumgartner and Burchardt, 2004] Baumgartner, P. and Burchardt, A. B. (2004). Logic Programming Infrastructure for Inferences on FrameNet. In *Proceedings of the European Conference on Logics in Artificial Intelligence (LNAI-04)*.

[Bouma et al., 2005a] Bouma, G., Mur, J., and van Noord, G. (2005a). Reasoning over Dependency Relations for QA. In *Proceedings of the IJCAI workshop on Knowledge and Reasoning for Answering Questions (KRAQ-05)*.

[Bouma et al., 2005b] Bouma, G., Mur, J., van Noord, G., van der Plas, L., and Tiedemann, J. (2005b). Question Answering for Dutch using Dependency Relations. In *Cross Language Evaluation Forum: Working Notes for the CLEF 2005 Workshop (CLEF-05)*.

[Brown et al., 1993] Brown, P. E., Pietra, S. A. D., Pietra, V. J. D., and Mercer, R. L. (1993). The Mathematics of Statistical Machine Translation: Parameter Estimation. *Computational Linguistics*, 19(2).

[Burchardt et al., 2006] Burchardt, A., Erk, K., Frank, A., Kowalski, A., Pado, S., and Pinkal, M. (2006). The SALSA Corpus: A German Corpus Resource for Lexical Semantics. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC-06)*.

[Burchardt and Frank, 2006] Burchardt, A. and Frank, A. (2006). Approaching Textual Entailment with LFG and FrameNet Frames. In *Proceedings of the Second PASCAL Challenges Workshop*.

[Cardie et al., 2000] Cardie, C., Ng, V., Pierce, D., and Buckley, C. (2000). Examining the Role of Statistical and Linguistic Knowledge Sources in a General-Knowledge Question-Answering System. In *Proceedings of the Sixth Applied Natural Language Processing Conference (ANLP-00)*.

[Clarke et al., 2001] Clarke, C. L. A., Cormack, G. V., and Lynam, T. R. (2001). Exploiting Redundancy in Question Answering. In *Proceedings of the 24th ACM-SIGIR International Conference on Research and Development in Information Retrieval (SIGIR-01)*.

[Cui et al., 2004] Cui, H., Li, K., Sun, R., Chua, T.-S., and Kan, M.-Y. (2004). National University of Singapore at the TREC-13 Question Answering Main Task. In *Proceedings of the 2004 Edition of the Text REtrieval Conference (TREC-04)*.

[Cui et al., 2005] Cui, H., Sun, R., Li, K., Kan, M.-Y., and Chua, T.-S. (2005). Question Answering Passage Retrieval Using Dependency Relations. In *Proceedings of the 28th ACM-SIGIR International Conference on Research and Development in Information Retrieval (SIGIR-05)*.

[Cunningham et al., 2002] Cunningham, H., Maynard, D., Bontcheva, K., and Tablan, V. (2002). GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications. In *Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics (ACL-02)*.

[Dang et al., 2006] Dang, H. T., Lin, J., , and Kelly, D. (2006). Overview of the TREC 2006 Question Answering Track. In *Proceedings of the 2006 Edition of the Text REtrieval Conference (TREC-06)*.

[Dang et al., 2007] Dang, H. T., Lin, J., , and Kelly, D. (2007). Overview of the TREC 2007 Question Answering Track. In *Proceedings of the 2007 Edition of the Text REtrieval Conference (TREC-07)*.

[Daniel et al., 1992] Daniel, K., Schabes, Y., Zaidel, M., and Egedi, D. (1992). A Freely Available Wide Coverage Morphological Analyzer for English. In *Proceedings of the 14th International Conference on Computational Linguistics (COLING-92)*.

[Dumais et al., 2002] Dumais, S., Bankom, M., Brill, E., Lin, J., and Ng, A. (2002). Web Question Answering: Is More Always Better? *Proceedings of the 25th ACM-SIGIR International Conference on Research and Development in Information Retrieval (SIGIR-02)*.

[Echihabi et al., 2004] Echihabi, A., Hermjakob, U., Hovy, E., Marcu, D., Melz, E., and Ravichandran, D. (2004). Multiple-Engine Question Answering in TextMap. In *Proceedings of the 2003 Edition of the Text REtrieval Conference (TREC-03)*.

[Erk and Pado, 2006] Erk, K. and Pado, S. (2006). Shalmaneser - A Flexible Toolbox for Semantic Role Assignment. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC-06)*.

[Fang, 2008] Fang, H. (2008). A Re-examination of Query Expansion Using Lexical Resources. In *Proceedings of the 46th Annual Meetings of the Association for Computational Linguistics (ACL-08)*.

[Fillmore, 1968] Fillmore, C. J. (1968). *The Case for Case*, pages 1–88. Holt, Rinehart and Winston.

[Fillmore, 1976] Fillmore, C. J. (1976). Frame Semantics and the Nature of Language. In *Annals of the New York Academy of Sciences: Conference on the Origin and Development of Language and Speech*, volume 280.

[Fliedner, 2004] Fliedner, G. (2004). Towards Using FrameNet for Question Answering. In *Proceedings of the LREC 2004 Workshop on Building Lexical Resources from Semantically Annotated Corpora*.

[Gale and Church, 1991] Gale, W. A. and Church, K. W. (1991). A Program for Aligning Sentences in Bilingual Corpora. In *Proceedings of the 29th Annual Meeting of the Association for Computational Linguistics (ACL-91)*.

[Gildea and Jurafsky, 2000] Gildea, D. and Jurafsky, D. (2000). Automatic Labeling of Semantic Roles. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics (ACL-00)*.

[Gildea and Jurafsky, 2002] Gildea, D. and Jurafsky, D. (2002). Automatic Labeling of Semantic Roles. *Computational Linguistics*, 28(3):245–288.

[Gildea and Palmer, 2002] Gildea, D. and Palmer, M. (2002). The Necessity of Parsing for Predicate Argument Recognition. In *Proceedings of the 40th Meeting of the Association for Computational Linguistics (ACL-02)*.

[Giuglea and Moschitti, 2006] Giuglea, A.-M. and Moschitti, A. (2006). Towards Free-text Semantic Parsing: A Unified Framework Based on FrameNet, VerbNet and PropBank. In *Proceedings of the Workshop on Learning Structured Information for Natural Language Applications, 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL-06)*.

[Gordon and Swanson, 2007] Gordon, A. and Swanson, R. (2007). Generalizing Semantic Role Annotations across Syntactically Similar Verbs. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics (ACL-07)*.

[Gove, 1961] Gove, P. B., editor (1961). *Webster's Third New International Dictionary of the English Language, Unabridged*. Merriam-Webster.

[Graff, 2002] Graff, D. (2002). The AQUAINT Corpus of English News Text.

[Grinberg et al., 1995] Grinberg, D., Lafferty, J., and Sleator, D. (1995). A Robust Parsing Algorithm for Link Grammars. In *Proceedings of the Fourth International Workshop on Parsing Technologies*.

[Harabagiu et al., 2001] Harabagiu, S., Moldovan, D., Pacsa, M., Mihalcea, R., Surdeanu, M., Buneascu, R., Girju, R., Rus, V., and Morarescu, P. (2001). FALCON:

Boosting Knowledge for Answer Engines. In *Proceedings of the 2001 Edition of the Text REtrieval Conference (TREC-01))*.

[Hatcher and Gospodnetić, 2004] Hatcher, E. and Gospodnetić, O. (2004). *Lucene in Action*. Manning Publications Co.

[Hearst, 1992] Hearst, M. (1992). Automatic Acquisition of Hyponyms from Large Text Corpora. In *Proceedings of the Fourteenth International Conference on Computational Linguistics (COLING-92)*.

[Hickl et al., 2006] Hickl, A., Bensley, J., John Williams, K. R., Rink, B., , and Shi, Y. (2006). Recognizing Textual Entailment with LCC's GROUNDHOG System. In *Proceedings of the Second PASCAL Challenges Workshop*.

[Hirst and St-Onge, 1998] Hirst, G. and St-Onge, D. (1998). Lexical Chains as Representations of Context for the Detection and Correction of Malapropisms. In Fellbaum, C., editor, *WordNet: An electronic Lexical Database*. MIT Press.

[Hovy et al., 2000] Hovy, E., Gerber, L., Hermjakob, U., Junk, M., and Lin, C.-Y. (2000). Question Answering in Webclopedia. In *Proceedings of the 2000 Edition of the Text REtrieval Conference (TREC-00)*.

[Huddleston and Pullum, 2002] Huddleston, R. and Pullum, G. K. (2002). *The Cambridge Grammar of the English Language*. Cambridge University Press.

[Ibrahim et al., 2003] Ibrahim, A., Katz, B., and Lin, J. (2003). Extracting Structural Paraphrases from Aligned Monolingual Corpora. In *Proceedings of the International Workshop on Paraphrase (IWP-2003)*.

[Jeon et al., 2005] Jeon, J., Croft, W. B., and Lee, J. H. (2005). Finding Similar Questions in Large Question and Answer Archives. In *Proceedings of the 14th ACM International Conference on Information and Knowledge Management (CIKM-05)*.

[Jiang and Conrath, 1997] Jiang, J. and Conrath, D. (1997). Semantic Similarity based on Corpus Statistics and Lexical Taxonomy. In *Proceedings of the International Conference on Research in Computational Linguistics*.

[Kaisser, 2005] Kaisser, M. (2005). QuALiM at TREC 2005: Web-Question Answering with FrameNet. In *Proceedings of the 2005 Edition of the Text REtrieval Conference (TREC-05)*.

[Kaisser, 2006] Kaisser, M. (2006). Web Question Answering by Exploiting Wide-Coverage Lexical Resources. In *Proceedings of the 11th ESSLLI Student Session*.

[Kaisser, 2008] Kaisser, M. (2008). The QuALiM Question Answering Demo: Supplementing Answers with Paragraphs drawn from Wikipedia. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics (ACL-08)*.

[Kaisser and Becker, 2004] Kaisser, M. and Becker, T. (2004). Question Answering by Searching Large Corpora with Linguistic Methods. In *Proceedings of the 2004 Edition of the Text REtrieval Conference (TREC-04)*.

[Kaisser et al., 2008] Kaisser, M., Hearst, M., and Lowe, J. (2008). Improving Search Result Quality by Customizing Summary Lengths. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics (ACL-08)*.

[Kaisser and Lowe, 2008] Kaisser, M. and Lowe, J. (2008). Creating a Research Collection of Question Answer Sentence Pairs with Amazon's Mechanical Turk. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC-08)*.

[Kaisser et al., 2006] Kaisser, M., Scheible, S., and Webber, B. (2006). Experiments at the University of Edinburgh for the TREC 2006 QA track. In *Proceedings of the 2006 Edition of the Text REtrieval Conference (TREC-06)*.

[Kaisser and Webber, 2007] Kaisser, M. and Webber, B. (2007). Question Answering based on Semantic Roles. In *Proceedings of the ACL 2007 Deep Linguistic Processing Workshop (DLP-07)*.

[Katz et al., 2002] Katz, B., Felshin, S., Yuret, D., Ibrahim, A., Lin, J., Marton, G., McFarland, A. J., and Temelkuran, B. (2002). Omnibase: Uniform Access to Heterogeneous Data for Question Answering. In *Proceedings of the 7th International Workshop on Applications Of Natural Language to Information Systems (NLDB-02)*.

[Katz and Lin, 2003] Katz, B. and Lin, J. (2003). Selectively Using Relations to Improve Precision in Question Answering. In *Proceedings of the EACL 2003 Workshop on Natural Language Processing for Question Answering (NLP4QA)*.

[Kingsbury and Palmer, 2002] Kingsbury, P. and Palmer, M. (2002). From Treebank to PropBank. In *Proceedings of the 3rd International Conference on Language Resources and Evaluation (LREC-02)*.

[Kingsbury et al., 2002] Kingsbury, P., Palmer, M., and Marcus, M. (2002). Adding Semantic Annotation to the Penn TreeBank. In *Proceedings of the Human Language Technology Conference (HLT-02)*.

[Klein and Manning, 2003a] Klein, D. and Manning, C. D. (2003a). Accurate Unlexicalized Parsing. In *Proceedings of the 41st Meeting of the Association for Computational Linguistics (ACL-03)*.

[Klein and Manning, 2003b] Klein, D. and Manning, C. D. (2003b). Fast Exact Inference with a Factored Model for Natural Language Parsing. In *Advances in Neural Information Processing Systems 15*.

[Kwok et al., 2001] Kwok, C. C. T., Etzioni, O., and Weld, D. S. (2001). Scaling Question Answering to the Web. In *Proceedings of the 10th World Wide Web Conference (WWW-01)*.

[Leacock and Chodorow, 1998] Leacock, C. and Chodorow, M. (1998). Combining Local Context and WordNet Similarity for Word Sense Identification. In Fellbaum, C., editor, *WordNet: An electronic lexical database*. MIT Press.

[Lee et al., 2008] Lee, J.-T., Kim, S.-B., Song, Y.-I., , and Rim, H.-C. (2008). Bridging Lexical Gaps between Queries and Questions on Large Online Q&A Collections with Compact Translation Models. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing (EMNLP-08)*.

[Levin, 1993] Levin, B. (1993). *English Verb Classes and Alternations: A Preliminary Investigation.* University of Chicago Press, Chicago.

[Lin, 1998a] Lin, D. (1998a). An Information-theoretic Definition of Similarity. In *Proceedings of the International Conference on Machine Learning (ICML-98)*.

[Lin, 1998b] Lin, D. (1998b). Dependency-based Evaluation of MINIPAR. In *Workshop on the Evaluation of Parsing Systems*.

[Lin, 2003] Lin, D. (2003). The Format of Dependency Trees Output from MiniPar. www.cs.ualberta.ca/˜lindek/minipar.htm.

[Lin and Pantel, 2001] Lin, D. and Pantel, P. (2001). Discovery of Inference Rules for Question-Answering. *Natural Language Engineering*, 7(4):343–360.

[Lin and Katz, 2005] Lin, J. and Katz, B. (2005). Building a Reusable Test Collection for Question Answering. *Journal of the American Society for Information Science and Technology (JASIST)*.

[Macdonald and Ounis, 2006] Macdonald, C. and Ounis, I. (2006). The TREC Blogs06 Collection: Creating and Analysing a Blog Test Collection. Technical Report DCS Technical Report TR-2006-224, Department of Computing Science, University of Glasgow.

[Magnini et al., 2006] Magnini, B., Giampiccolo, D., Forner, P., Ayache, C., Jijkoun, V., Osenova, P., Peas, A., Rocha, P., Sacaleanu, B., and Sutcliffe, R. (2006). Overview of the CLEF 2006 Multilingual Question Answering Track. In *Working Notes for the CLEF 2006 Workshop (CLEF-06)*.

[Magnini et al., 2004] Magnini, B., Vallin, A., Ayache, C., Erbach, G., Penas, A., de Rijke, M., Rocha, P., Simov, K., and Sutcliffe, R. (2004). Overview of the CLEF 2004 Multilingual Question Answering Track. In *Results of the CLEF 2004 Cross-Language System Evaluation Compaign (CLEF-04)*.

[Marcus et al., 1994a] Marcus, M., Kim, G., Marcinkiewicz, M., MacIntyre, R., Bies, A., Ferguson, M., Katz, K., and Schasberger, B. (1994a). The Penn Treebank: Annotating Predicate Argument Structure. In *Proceedings of the 1994 ARPA Human Language Technology Workshop*.

[Marcus et al., 1994b] Marcus, M. P., Santorini, B., and Marcinkiewicz, M. A. (1994b). Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2).

[McNamee et al., 2008] McNamee, P., Snow, R., Schone, P., and Mayfield, J. (2008). Learning Named Entity Hyponyms for Question Answering. In *Proceedings of the Third International Joint Conference on Natural Language Processing (IJCNLP-08)*.

[Miller et al., 1993] Miller, G. A., Beckwith, R., Fellbaum, C., Gross, D., and Miller, K. (1993). Introduction to WordNet: An On-Line Lexical Database. *Journal of Lexicography*, 3(4):235–244.

[Monz, 2004] Monz, C. (2004). Minimal Span Weighting Retrieval for Question Answering. In *Proceedings of the SIGIR Workshop on Information Retrieval for Question Answering (IR4QA)*.

[Moschitti et al., 2005] Moschitti, A., Giuglea, A.-M., Coppola, B., and Basili, R. (2005). Hierarchical Semantic Role Labeling. In *Proceedings of the Ninth Conference on Computational Natural Language Learning (CoNLL-05)*.

[Moschitti et al., 2007] Moschitti, A., Quarteroni, S., Basili, R., and Manandhar, S. (2007). Exploiting Syntactic and Shallow Semantic Kernels for Question/Answer Classification. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL-07)*.

[Mur, 2008] Mur, J. (2008). *Off-line Answer Extraction for Question Answering*. PhD thesis, Rijksuniversiteit Groningen.

[Narayanan and Harabagiu, 2004] Narayanan, S. and Harabagiu, S. (2004). Question Answering Based on Semantic Structures. In *Proceedings of the 20th international conference on Computational Linguistics (COLING-04)*.

[Novischi and Moldovan, 2006] Novischi, A. and Moldovan, D. (2006). Question Answering with Lexical Chains Propagating Verb Arguments. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL (COLING/ACL-06)*.

[Och and Ney, 2003] Och, F. J. and Ney, H. (2003). A Systematic Comparison of Various Statistical Alignment Models. *Computational Linguistics*, 29(1):19–52.

[Ofoghi et al., 2009] Ofoghi, B., JohnYearwood, , and Ma, L. (2009). The Impact of Frame Semantic Annotation Levels, Frame-Alignment Techniques, and Fusion Methods on Factoid Answer Processing. *Journal of the American Society for Information Science and Technology*, 60(2):247–263.

[Palmer et al., 2005] Palmer, M., Gildea, D., and Kingsbury, P. (2005). The Proposition Bank: An Annotated Corpus of Semantic Roles. *Computational Linguistics*, 31(1):71–106.

[Pasca and Harabagiu, 2001] Pasca, M. and Harabagiu, S. (2001). The informative role of WordNet in Open-Domain Question Answering. In *Proceedings of NAACL-01 Workshop on WordNet and Other Lexical Resources*.

[Patwardhan, 2003] Patwardhan, S. (2003). Incorporating Dictionary and Corpus Information into a Context Vector Measure of Semantic Relatedness. Master's thesis, University of Minnesota, Duluth.

[Pedersen et al., 2004] Pedersen, T., Patwardhan, S., and Michelizzi, J. (2004). WordNet::Similarity - Measuring the Relatedness of Concepts. In *Proceedings of the Nineteenth National Conference on Artificial Intelligence (AAAI-04)*.

[Pennacchiotti et al., 2008] Pennacchiotti, M., Cao, D. D., Basili, R., Croce, D., and Roth, M. (2008). Automatic Induction of FrameNet Lexical Units. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing (EMNLP-08)*.

[Porter, 2001] Porter, M. (2001). Snowball: A Language for Stemming Algorithms. http://snowball.sourceforge.net.

[Pradhan et al., 2005a] Pradhan, S., Hacioglu, K., Krugler, V., Ward, W., Martin, J. H., and Jurafsky, D. (2005a). Support Vector Learning for Semantic Argument Classification. *Machine Learning*, 60(1):11–39.

[Pradhan et al., 2004] Pradhan, S., Ward, W., Hacioglu, K., and Martin, J. H. (2004). Shallow Semantic Parsing using Support Vector Machines. In *Proceedings of the Human Language Technology Conference/North American Chapter of the Association for Computational Linguistics annual meeting (HLT/NAACL-2004)*.

[Pradhan et al., 2005b] Pradhan, S., Ward, W., Hacioglu, K., and Martin, J. H. (2005b). Semantic Role Labeling Using Different Syntactic Views. In *Proceedings of the Association for Computational Linguistics 43rd Annual Meeting (ACL-2005)*.

[Prager, 2006] Prager, J. (2006). Open-Domain Question-Answering. *Foundations and Trends in Information Retrieval*, 1(2).

[Prager et al., 2000] Prager, J., Brown, E., Coden, A., and Radev, D. (2000). Question-Answering by Predictive Annotation. In *Proceedings of the 23rd ACM-SIGIR International Conference on Research and Development in Information Retrieval (SIGIR-00)*.

[Punyakanok et al., 2004] Punyakanok, V., Roth, D., , and Yih, W.-T. (2004). Mapping Dependencies Trees: An Application to Question Answering. In *Proceedings of the 8th International Symposium on Artificial Intelligence and Mathematics*.

[Rabiner et al., 1991] Rabiner, L. R., Rosenberg, A. E., and Levinson, S. E. (1991). Considerations in Dynamic Time Warping Algorithms for Discrete Word Recognition. In *Proceedings of IEEE Transactions on Acoustics, Speech and Signal Processing*.

[Ravichandran and Hovy, 2002] Ravichandran, D. and Hovy, E. (2002). Learning Surface Text Patterns for a Question Answering System. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL-02)*.

[Resnik, 1995] Resnik, P. (1995). Using Information Content to Evaluate Semantic Similarity in a Taxonomy. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI-95)*.

[Rinaldi et al., 2003] Rinaldi, F., Dowdall, J., Kaljurand, K., Hess, M., and Molla, D. (2003). Explioting Paraphrases in a Question Answering System. In *The Second International Workshop on Paraphrasing: Paraphrase Acquisition and Applications (IWP-03)*, page 2532.

[Ruppenhofer et al., 2006] Ruppenhofer, J., Ellsworth, M., Petruck, M. R. L., Johnson, C. R., and Scheffczyk, J. (2006). FrameNet II: Extended Theory and Practice. Version of August 25, 2006.

[Schlaefer et al., 2007] Schlaefer, N., Ko, J., Betteridge, J., Sautter, G., Pathak, M., and Nyberg, E. (2007). Semantic Extensions of the Ephyra QA System for TREC 2007. In *Proceedings of the 2007 Edition of the Text REtrieval Conference (TREC-07)*.

[Schuler, 2005] Schuler, K. K. (2005). *VerbNet: A Broad-Coverage, Comprehensive Verb Lexicon*. PhD thesis, University of Pennsylvania.

[Shen and Klakow, 2006] Shen, D. and Klakow, D. (2006). Exploring Correlation of Dependency Relation Paths for Answer Extraction. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL (COLING/ACL-06)*.

[Shen and Lapata, 2007] Shen, D. and Lapata, M. (2007). Using Semantic Roles to Improve Question Answering. In *Proceedings of the 2007 Conference on Empirical Methods in Natural Language Processing (EMNLP-07)*.

[Shi and Mihalcea, 2005] Shi, L. and Mihalcea, R. (2005). Putting Pieces Together: Combining FrameNet, VerbNet and WordNet for Robust Semantic Parsing. In *Proceedings of the 6th International Conference on Computational Linguistics and Intelligent Text Processing (CICLing-05)*.

[Simpson and Weiner, 1989] Simpson, J. and Weiner, E., editors (1989). *Oxford English Dictionary*. Oxford University Press.

[Sleator and Temperley, 1993] Sleator, D. and Temperley, D. (1993). Parsing English with a Link Grammar. *Third International Workshop on Parsing Technologies (IWPT-93)*.

[Snow et al., 2005] Snow, R., Jurafsky, D., and Ng, A. Y. (2005). Learning Syntactic Patterns for Automatic Hypernym Discovery. In *Proceedings of Advances in Neural Information Processing Systems (NIPS-05)*.

[Snow et al., 2008] Snow, R., O'Connor, B., Jurafsky, D., and Ng, A. Y. (2008). Cheap and Fast - But is it Good? Evaluating Non-Expert Annotations for Natural Language Tasks. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing (EMNLP-08)*.

[Subirats and Sato, 2004] Subirats, C. and Sato, H. (2004). Spanish FrameNet and FrameSQL. In *Proceedings of the LREC Workshop on Building Lexical Resources from Semantically Annotated Corpora*.

[Sun et al., 2005] Sun, R., Jiang, J., Tan, Y. F., Cui, H., Chua, T.-S., and Kan, M.-Y. (2005). Using Syntactic and Semantic Relation Analysis in Question Answering. In *Proceedings of the 2005 Edition of the Text REtrieval Conference (TREC-05)*.

[Surdeanu et al., 2003] Surdeanu, M., Harabagiu, S., Williams, J., and Aarseth, P. (2003). Using Predicate-Argument Structures for Information Extraction. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics (ACL-03)*.

[Tiedemann, 2005] Tiedemann, J. (2005). Integrating Linguistic Knowledge in Passage Retrieval for Question Answering. In *Proceedings of the 2005 Conference on Empirical Methods in Natural Language Processing (EMNLP-05)*.

[Tietze et al., 2009] Tietze, M. I., Winterboer, A., and Moore, J. D. (2009). The Effect of Linguistic Devices in Information Presentation Messages on Recall and Comprehension. In *Proceedings of Proceedings of the 12th European Workshop on Natural Language Generation (ENLG-09)*.

[Tufis and Mason, 1998] Tufis, D. and Mason, O. (1998). Tagging Romanian Texts: a Case Study for QTAG, a Language Independent Probabilistic Tagger. In *Proceedings of the First International Conference on Language Resources and Evaluation (LREC-98)*.

[Vallin et al., 2005] Vallin, A., Magnini, B., Giampiccolo, D., Aunimo, L., Ayache, C., Osenova, P., Peas, A., de Rijke, M., Sacaleanu, B., Santos, D., and Sutcliffe, R. (2005). Overview of the CLEF 2005 Multilingual Question Answering Track. In *Cross Language Evaluation Forum: Working Notes for the CLEF 2005 Workshop (CLEF-05)*.

[Vicedo and Ferrandez, 2000a] Vicedo, J. L. and Ferrandez, A. (2000a). A Semantic Approach to Question Answering Systems. In *Proceedings of the Ninth Text Retrieval Conference (TREC-9)*.

[Vicedo and Ferrandez, 2000b] Vicedo, J. L. and Ferrandez, A. (2000b). Importance of Pronominal Anaphora Resolu-tion in Question Answering Systems. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics (ACL-00)*.

[Voorhees, 1994] Voorhees, E. M. (1994). Query Expansion Using Lexical-semantic Relations. In *Proceedings of the 1994 ACM-SIGIR Conference on Research and Development in Information Retrieval (SIGIR-94)*.

[Voorhees, 2003] Voorhees, E. M. (2003). Overview of the TREC 2003 Question Answering Track. In *Proceedings of the 2003 Edition of the Text REtrieval Conference (TREC-03)*.

[Voorhees, 2004] Voorhees, E. M. (2004). Overview of the TREC 2004 Question Answering Track. In *Proceedings of the 2004 Edition of the Text REtrieval Conference (TREC-04)*.

[Voorhees and Dang, 2005] Voorhees, E. M. and Dang, H. T. (2005). Overview of the TREC 2005 Question Answering Track. In *Proceedings of the 2005 Edition of the Text REtrieval Conference (TREC-05)*.

[Voorhees and Tice, 2000] Voorhees, E. M. and Tice, D. M. (2000). Building a Question Answering Test Collection. In *Proceedings of the 23rd ACM-SIGIR International Conference on Research and Development in Information Retrieval (SIGIR-00)*.

[Wu et al., 2003] Wu, L., Huang, X., Zhou, Y., Du, Y., and You, L. (2003). Fduqa on trec2003 qa task. In *Proceedings of the 2003 Edition of the Text REtrieval Conference (TREC-03)*.

[Wu and Palmer, 1994] Wu, Z. and Palmer, M. (1994). Verb Semantics and Lexical Selection. In *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics (ACL-94)*.

[Xue and Palmer, 2004] Xue, N. and Palmer, M. (2004). Calibrating Features for Semantic Role Labeling. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing (EMNLP-04)*.

[Zheng et al., 2007] Zheng, Q., Tian, Z., and Wang, Y. (2007). Knowledge-Oriented Answer Extraction in Chinese Question Answering System for E-Learning. In *Proceedings of the 11th International Conference on Computer Supported Cooperative Work in Design, (CSCWD-07)*.