Question Answering for Dutch using Dependency Relations*

Gosse Bouma, Jori Mur, Gertjan van Noord,
Lonneke van der Plas and Jörg Tiedemann
Information Science
Rijksuniversiteit Groningen
Postbus 716, 9700 AS Groningen
{gosse,mur,vannoord,vdplas,tiedeman}@let.rug.nl

Abstract

We describe Joost, our QA system for Dutch, which makes extensive use of dependency relations. We analyzed the full Dutch CLEF QA corpus syntactically and mined it off-line for information that may be useful for QA. Joost answers questions either by table look-up, or by searching for answers in paragraphs returned by an IR engine. In both cases, dependency relations are used to identify and rank potential answers.

1 Introduction

Joost is a monolingual QA system for Dutch which makes heavy use of syntactic information. Most questions are answered by retrieving relevant paragraphs from the document collection, using keywords from the question. Next, potential answers are identified and ranked using a number of clues. Apart from obvious clues (i.e. IR-score and the frequency with which the answer was found), we also use syntactic structure to identify and rank answer strings. A second strategy is based upon the observation that certain question types can be anticipated, and the corpus can be searched off-line for answers to such questions. Whereas previous approaches have used regular expressions to extract the relevant relations, we use patterns of dependency relations. To this end, the whole corpus has been analyzed syntactically.

In the next section, we describe the building blocks of our QA system, i.e. the Alpino dependency parser, utilities for reasoning with dependency relations, relation tables which are extracted off-line, and ISA-relations between named entities and concepts. In section 3, we describe Joost. Questions are parsed and classified using Alpino. Depending on the question class, questions are answered by means of table look-up or an IR-based method. General *which*-questions and definition questions are answered using the automatically acquired ISA-relations as an additional resource. In section 4, we discuss the results of Joost on the CLEF 2005 QA task.

2 Preliminaries

2.1 Syntactic Preprocessing

We have used the Alpino-system to parse the full text collection for the Dutch CLEF QA-task. The resulting dependency parse trees are stored as XML, and can be processed and searched in various ways, for instance, using XPath and XSLT.

The Alpino-system is a linguistically motivated, wide-coverage, grammar and parser for Dutch. The constraint-based grammar follows the tradition of HPSG (Pollard and Sag, 1994). It currently

*This research was carried out as part of the research program for *Interactive Multimedia Information Extraction*, IMIX, financed by NWO, the Dutch Organisation for Scientific Research.

consists of over 500 grammar rules (defined using inheritance) and a large and detailed lexicon (over 100.000 lexemes). To ensure coverage, heuristics have been implemented to deal with unknown words and ungrammatical or out-of-coverage sentences (which may nevertheless contain fragments that are analyzable). The grammar provides a 'deep' level of syntactic analysis, in which WH-movement, raising and control, and the Dutch verb cluster (which may give rise to 'crossing dependencies') are given a principled treatment. The output of the system is a dependency graph, compatible with the annotation guidelines of the Corpus of Spoken Dutch.

A left-corner chart parser is used to create the parse forest for a given input string. A manually corrected treebank of 140.000 words was used to train a maximum entropy disambiguation model. Beam-search is used as a heuristic to extract the most probable parse from the parse forest effeciently. (Malouf and van Noord, 2004) show that the accuracy of the system, when evaluated on a test-set of 500 newspaper sentences, is over 88%, which is in line with state-of-the-art systems for English.

A second extension of the system for QA, was the inclusion of a Named Entity Classifier. The Alpino system already includes heuristics for recognizing proper names. Thus, the classifier needs to classify strings which have been assigned a NAME part of speech by grammatical analysis, as being of the subtype PER, ORG, GEO or MISC.[1] To this end, we collected lists of person names (120K), geographical names (12K), organization names (26k), and miscalleneous items (2K). The data are primarily extracted from the Twente News Corpus, a collection of over 300 million words of newspaper text, which comes with annotation for the names of people, organizations, and locations, involved in a particular news story. For unknown names, a maximum entropy classifier was trained, using the Dutch part of the shared task for CONLL 2003.[2] The accuracy on unseen CONLL data of the resulting classifier (which combines dictionary look-up and a maximum entropy classifier) is 88.2%.

To this end, the text collection was tokenized (into 78 million words) and segmented into (4.1 million) sentences. Parsing this amount of text takes well over 500 CPU days. We used a Beowulf Linux cluster of 128 Pentium 4 processors[3] to complete the process in about three weeks. The dependency trees are stored as (25 Gb of) XML.

2.2 Reasoning over Dependency Relations

Several researchers have attempted to use syntactic information, and especially dependency relations, in QA. One approach is to look for an exact match between dependency tuples derived from the question and those present in a potential answer (Katz and Lin, 2003; Litkowski, 2004). Attardi et al. (2002) and Mollá and Gardiner (2005) compute the match between question and answer using a metric which basically computes the overlap in dependency relations between the two. Punyakanok, Roth, and Yih (2004) compute the tree edit distance between the dependency trees of the question and answer, and select answers from sentences which minimize this distance.

We have implemented a system in which dependency patterns derived from the question must be matched by equivalent dependency relations in a potential answer. The dependency analysis of a sentence gives rise to a set of dependency relations of the form ⟨`Head/HIx, Rel, Dep/DIx`⟩, where `Head` is the root form of the head of the relation, and `Dep` is the head of the constituent that is the dependent. `Hix` and `DIx` are string indices, which distinguish repeated occurrences of the same token in a string, and `Rel` is the name of the dependency relation. For instance, the dependency analysis of sentence (1-a) is (1-b).

(1) a. Mengistu kreeg asiel in Zimbabwe (*Mengistu was given asylum in Zimbabwe*)

b. $\left\{ \begin{array}{ll} \langle \texttt{krijg/2, su, mengistu/1} \rangle, & \langle \texttt{krijg/2, obj1, asiel/3} \rangle, \\ \langle \texttt{krijg/2, mod, in/4} \rangle, & \langle \texttt{in/4, obj1, zimbabwe/5} \rangle \end{array} \right\}$

A dependency pattern is a set of (partially underspecified) dependency relations:

[1] Various other entities which sometimes are dealt with by NEC, such as dates and measure phrases, can be identified using the information present in POS tags and dependency labels.

[2] `http://cnts.uia.ac.be/conll2003/ner/`

[3] which is part of the High-Performance Computing centre of the University of Groningen

(2) $\{$ ⟨krijg/K, obj1, asiel/A⟩, ⟨krijg/K, su, Su/S⟩ $\}$

A pattern may contain variables, represented here by (words starting with) a capital. A pattern P matches a set of dependency relations R if $P \subset R$, under some substitution of variables. The pattern in (2) matches with the set in (1-b), instantiating Su as mengistu.

Equivalences can be defined to account for the fact that in some cases we want a pattern to match a set of dependency relations that slightly differs from it, but nevertheless expresses the same semantic relation. For instance, the subject of an active sentence may be expressed as a PP-modifier headed by *door* (*by*) in the passive:

(3) a. Zimbabwe verleende asiel aan Mengistu (*Zimbabwe gave asylum to Mengistu*)
b. Aan Mengistu werd asiel verleend door Zimbabwe (*Mengistu was given asylum by Zimbabwe*)

The following equivalence accounts for this:

$$\{\langle \texttt{Vb/V,su,Su/S}\rangle\} \Leftrightarrow \{\ \langle \texttt{word/W,vc,Vb/V}\rangle,\ \ \langle \texttt{Vb/V,mod,door/D}\rangle,\ \ \langle \texttt{door/D,obj1,Su/S}\rangle\ \}$$

Here, the verb *word* is (the root form of) the passive auxiliary, which takes a verbal complement headed by the verb Vb.

Given an equivalence $Lhs \Leftrightarrow Rhs$, a pattern P containing Lhs is equivalent to a pattern P', which is identical to P, except that Lhs has been replaced by Rhs. A pattern P now also matches with a set of relations R if there is some equivalent pattern P', and P' is a subset of R, under some substitution of variables.

We have implemented 13 additional equivalence rules, to account for, among others, word order variation within appostions, the equivalence of genitives and *van*-PPs, equivalence between appositions and simple predicative sentence, coordination, and relative clauses. The equivalence rules we have implemented so far express linguistic equivalences, and thus are both general and domain independent. In Bouma, Mur, and van Noord (2005), we show that the inclusion of equivalence rules has a positive effect on various components of our QA system. In the future, we hope to extend this with equivalences which are restricted to specific relations (i.e. such as the equivalence between *X writes Y* and *X is the author of Y*), using techniques for acquiring such equivalences automatically from parsed corpora as in Lin and Pantel (2001).

2.3 Off-line Retrieval

Off-line methods have proven to be very effective in QA (Fleischman, Hovy, and Echihabi, 2003). Before actual questions are known, a corpus is exhaustively searched for potential answers to specific question types (capital, abbreviation, inhabitants, year of birth, ...). The answers are extracted from the corpus off-line and stored in a structured table for quick and easy access.

Jijkoun, Mur, and de Rijke (2004) show that extraction patterns defined in terms of dependency relations are more effective than regular expression patterns over surface strings. Following this observation, we used the module for dependency pattern matching to exhaustively search the parsed corpus for potential answers to frequently occurring question types. For instance, the pattern in (4) extracts information about organizations and their founders.

(4) $\{$ ⟨richt_op/R, su, Founder/S⟩, ⟨richt_op/R, obj1, Founded/O⟩ $\}$

The verb *oprichten* (*to found*) can take on a wide variety of forms (active, with the particle *op* split from the root, participle, and infinitival, either the founder or the organization can be the first constituent in the sentence, in passives the founder may be part of a *door* (*by*) phrase, and in control constructions the founder may be found as the subject of a governing clause. In all cases, modifiers may intervene between the relevant constituents:

(5) a. **Minderop** richtte **de Tros** op toen .... (*Minderop founded the Tros when...*)

Relation	tuples	uniq	Relation	tuples	uniq	Relation	tuples	uniq
Abbreviation	21.497	8.543	Currency	6.619	222	Function	77.028	46.589
Age	22.143	18520	Died Age	1.127	834	Inhabitants	708	633
Born Date	2356	1.990	Died Date	583	544	Nobel Prize	169	141
Born Loc	937	879	Died Loc	664	583			
Capital	2.146	515	Founded	1.021	953			

Table 1: Size of extracted relation tables.

b. **Kasparov** heeft **een nieuwe Russische Schaakbond** opgericht en... (*Kasparov has founded a new Russian Chess Union and...*)
c. ... toen **de Generale Bank** bekend maakte met de Belgische Post **een "postbank"** op te richten. (*when the General Bank announced to found a "postal bank" with the Belgian Mail*).

Such variation is almost impossible to capture accurately using regular expressions, whereas dependency relations can exploit the fact that in allmost all cases the organization and its founder can be identified as the object and subject of the verb with the root form *oprichten*. The pattern in (4) suffices to extract this relation from all of the examples above.

Equivalence rules can be used to deal with other forms of syntactic variation. For instance, once we define a pattern to extract the country and its capital from (6-a), the equivalence rules can be used to match this pattern against the alternative formulations in (6-b)- (6-d) as well.

(6) a. de hoofdstad van Afghanistan, Kabul (*the capital of Afghanistan, Kabul*)
b. Kabul, de hoofdstad van Afghanistan (*Kabul, the capital of Afghanistan*)
c. Afghanistans hoofdstad, Kabul (*Afghanistan's capital, Kabul*)
d. Kabul is de hoofdstad van Afghanistan (*Kabul is the capital of Afghanistan*)

Of course, the same holds for all other relations that are extracted off-line, and thus, the development effort per relation decreases, while recall typically increases.

Table 1 lists all the relations we extracted. Each second and third column list the overall number of extracted tuples and extracted unique tuples (types) respectively.

2.4 Extracting ISA relations

Fine-grained named entity classification is useful for answering WH-questions and definition questions. Both Pasça (2004) and Pantel and Ravichandran (2004) describe methods for acquiring labels for named entities from large text corpora and evaluate the results in the context of web search and question answering. Pantel and Ravichandran (2004) use the apposition relation to find potential labels for named entities. The apposition relation is the relation that holds between *Delors* and *president of the European Commission* in sentences like *Delors, president of the European Commission, arrived yesterday.*

From the fully parsed Dutch CLEF text collection, we extracted 295 unique apposition tuples, consisting of a noun (used as class label) and a named entity. The resulting table contains, for instance, 112 names of *ferry boats* (*Estonia, Anna Maria Lauro, Sally Star* etc.) and no less than 2951 national team coaches (*Bobby Robson, Jack Charlton, Menotti, Berti Vogts* etc.). By focussing on the most frequent label for a named entity, most of the noise can be discarded. For instance, *Guus Hiddink* occurs 17 times in the extracted apposition tuples, 5 times as *bondscoach* (*national team chef*), and once with various other labels (*boss, colleague, guest, newcomer, ...*). In van der Plas and Bouma (2005), we show that automatically acquired class labels for named entities improve the performance of our QA system on *which* questions and definition questions.

3 Joost

In this section, we describe the components of our QA system, Joost. Questions are analyzed and assigned a question class. If the class corresponds to a relation for which information has been extracted off-line, answers and corresponding document id's are retrieved from the relevant relation table. Otherwise, Information Retrieval is used to find paragraphs relevant to the question. Linguistic techniques are used to extract potential answers from these paragraphs. Potential answers are ranked on the basis of a score which combines, among others, IR-score, frequency of the answer, and the amount of overlap in dependency relations between question and the sentence from which the answer was extracted.

3.1 Question Analysis

Question analysis is the task of assigning a specific class (`person, location, date, ...`) to a question. Syntactic analysis helps to determine the question stem in complex WH-phrases (*With which Palestinian organization...*) and can help to identify additional properties of the question (i.e. *Give the name of a Japanese city that was struck by an earthquake* asks for the name of a city, not of an earthquake). Lexical semantic knowledge is required to recognize that *Which region in the US has ...* asks for a geographical named entity, whereas *Which car factory was bought by ...* asks for an organizational named entity.

Each incoming question is parsed by Alpino. To improve parsing accuracy on this specific task, the disambiguation model was retrained on a corpus which contained annotated and manually corrected dependency trees for 650 quiz questions.[4] The retrained model achieves an accuracy of 92.7% and 88.3% on the CLEF 2003 and 2004 questions, respectively. For CLEF 2005, we used a model which was trained on data which also included (manually corrected dependency trees of) the CLEF 2003 and 2004 questions. It achieved an accuracy of 97.6 on CLEF 2005 questions.

On the basis of the dependency relations returned by the parser the question class is determined. Joost distinguishes between 29 different question classes. 18 question classes are related to the relation tuples that were extracted off-line. Note that a single relation can often be questioned in different ways. For instance, whereas a frequent question type asks for the meaning of an acronym (*What does the abbreviation RSI stand for?*), a less frequent type asks for the abbreviation of a given term (*What is the abbreviation of Mad Cow Disease?*). The other 11 question classes identify questions asking for an amount, the date or location of an event, the (first) name of a person, the name of an organization, *how*-questions, WH-questions, and definition questions.

For each question class, one or more syntactic patterns are defined. For instance, the following pattern accounts for questions asking for the capital of a country:

(7) $$\left\{ \begin{array}{ll} \langle \texttt{wat/W, wh, is/I} \rangle, & \langle \texttt{is/I, su, hoofdstad/H} \rangle \\ \langle \texttt{hoofdstad/H, mod, van/V} \rangle, & \langle \texttt{van/V, obj1, Country/C} \rangle \end{array} \right\}$$

Depending on the question class, it is useful to identify one or two additional arguments . For instance, the dependency relations assigned to the question *Wat is de hoofdstad van Togo?* (*What is the capital of Togo?*) match with the pattern in (7), and instantiate `Country` as *Togo*. Therefore, the question class `capital` is assigned, with `Togo` as additional argument. Similarly, *Who is the king of Norway?* is classified as `function(king,Norway)`, and *In which year did the Islamic revolution in Iran start?* is classified as `date(revolution)`.

Some question classes require access to lexical semantic knowledge. For instance, to determine that *In which American state is Iron Mountain?* asks for a location, the systeem needs to know that *state* refers to a location, and to determine that *Who is the advisor of Yasser Arafat?* should be classifed as `function(advisor,Yasser Arafat)`, it needs to know that *advisor* is a function. We obtained such knowledge mainly from Dutch EuroWordNet (Vossen, 1998). The list of function words (indicating function roles such as *president, queen, captain, secretary-general, etc.*) was

[4] From the *Winkler Prins spel*, a quiz game. The material was made available to us by the publisher, *Het Spectrum, bv.*

expanded semi-automatically with words from the corpus that were distributionally similar to those extracted from EWN (see van der Plas and Bouma (2005) for details).

Question classification was very accurate for the CLEF 2005 questions. There were a few cases where the additional arguments selected by the system did not seem the most optimal choice. Two clear mistakes were found (e.g. *What is the currency of Peru?* was classified as `currency(of)` and not as `currency(Peru)`).

3.2 Information Retrieval

For questions which cannot be answered by the relation tables, traditional keyword-based information retrieval (IR) is used to narrow down the search space for the linguistically informed part of the QA system which identifies answers. On the basis of keywords from the question, the IR system retrieves relevant passages from the corpus.

Keywords are derived from the question using its content words. Function words and other irrelevant words are removed using a static stop word list. We implemented an interface to seven publicly available IR engines (Tiedemann, 2004). We selected Zettair (Zobel et al., 2004) as the underlying system in our experiments because of speed and recall performance. The entire CLEF QA corpus (in its tokenized plain text version) has been indexed using the IR engine with its standard setup.

Earlier experiments have shown that a segmentation into paragraphs is most efficient for IR performance in QA. We used the existing markup in the corpus to determine the paragraph boundaries. This resulted in about 1.1 million paragraphs (including headers that have been marked as paragraphs). We did experiments with additional pre-processing, e.g., including proper lemmatization (using Alpino root forms) but we could not improve the IR performance compared to the baseline using standard settings. However, we did include labels of named entities found by Alpino in each paragraph as additional tokens. This makes it possible to search for paragraphs including certain types of named entities (e.g. location names and organizations) and special units (e.g. measure names and temporal expressions) corresponding to question types found by the question analyses component.

For CLEF, Zettair returns the 40 most relevant paragraphs given a query. For the QA@CLEF 2003 data, this gives a recall of 75%.

3.3 Answer Identification and Ranking

For questions that are answered by means of table look-up, the relation table provides an exact answer string. For other questions, it is necessary to extract answer strings from the set of paragraphs returned by IR. Given a set of paragraph id's, we retrieve from the parsed corpus the dependency relations for the sentences occurring in these paragraphs.

Various syntactic patterns are defined for (exact) answer identification. For questions asking for the name of a person, organization, or location, or for an amount or date, a constituent headed by a word with the appropriate named entity class has to be found. As all of these occur frequently in the corpus, usually many potential answers will be identified. An important task is therefore to rank potential answers.

The following features are used to determine the score of a short answer A extracted from sentence S:

- **Syntactic Similarity** The proportion of dependency relations from the question which match with dependency relations in S.
- **Answer Context** A score for the syntactic context of A.
- **Names** The proportion of proper names, nouns, and adjectives from the query which can be found in S and the sentence preceding S.
- **Frequency** The frequency of A in all paragraphs returned by IR.

- **IR** The score assigned to the paragraph from which A was extracted.

The score for syntactic similarity implements a preference for answers from sentences with a syntactic structure that overlaps with that of the question. Answer context implements a preference for answers that occur in the context of certain terms from the question. Given a question classified as `date(Event)`, for instance, date expressions which occur as a modifier of `Event` are preferred over date expressions occurring as sisters of `Event`, which in turn are preferred over dates which have no syntactic relation to `Event`.

The overall score for an answer is the weighted sum of these features. Weights were determined manually using previous CLEF data for tuning. The highest weights are used for Syntactic Similarity and Answer Context. The highest scoring answer is returned as the answer.

Ranking of answers on the basis of various features was initially developed for IR-based QA only. Answers found by table look-up were ranked only by frequency. Recently, we have started to use the scoring mechanism described above also for answers stemming from table look-up. As the tables contain pointers to the sentence from which a tuple was extracted, we can easily go back to the full sentence, and apply the scoring mechanisms described above.[5] Using more features to rank an answer provides a way to give the correct answer to questions like *Who is the German minister of Economy?*. The function table contains several names for German ministers, but does not distinguish between different departments. The most frequent candidate is *Klaus Kinkel* (54 entries), who is minister of foreign affairs. The correct name, *Günter Rexrodt*, occurs only 11 times. Using Syntactic Similarity and Names as an additional features, Joost manages to give the correct answer.

3.4 Special Cases

Temporally Restricted Questions. The CLEF 2005 test set contained a number of questions which were temporally restricted:

(8) a. Which vulcano erupted in June 1991?
b. Who was the mayor of Moscow in 1994?

The temporal information in these questions was treated similarly to all other information in the question, and we did not try to implement techniques which deal specifically with temporal restrictions. The mechanism for scoring potential answers takes into account the syntactic similarity and the overlap in names (including date expressions) between question and answer sentence, and this implements a preference for answers which are extracted from contexts referring to the correct date. Note that, as the same scoring technique is used for answers found by table look-up, this strategy should also be able to find the correct answer for questions such as (8-b), for which the function table might contain more than one answer.

Which-questions. General WH-questions, such as (9), are relatively difficult to answer. Whereas for most question types, the type of the answer is relatively clear (i.e. it should the name of a person or organization, or a date, etc.), this is not the case for WH-questions.

(9) a. Which fruit contains vitamin C?
b. Which ferry sank southeast of the island Utö?

To improve the performance of our system on such questions, we make use of two additional knowledge sources. From EuroWordNet, we imported all hypernym relations between nouns. Question (9-a) is assigned the question class `which(fruit)`. We use the hypernym relations to assign a higher score to answers which are hypernyms of `fruit`.[6]

As EuroWordNet does hardly include proper names, we also used the ISA-relations extracted from appositions containing a named entity, as described in section 2.4. Question (9-b) is assigned

[5] As no IR is involved in this case, the IR score is set to 1 for all answers.

[6] Unfortunately, EuroWordNet only contains two hypernyms for the synset *fruit*, none of which could be used to identify an answer to (9-a).

Question Type	# questions	correct answers	
		#	%
Factoid	114	62	54.39
Temporally Restricted Factoid	26	7	26.92
Definition	60	30	50
Overall	200	99	49.5

Table 2: CLEF scores

the question class `which(ferry)`. Candidate answers that are selected by Joost are: *Tallinn, Estonia, Raimo Tiilikainen* etc. Since, according to our apposition database, *Estonia* is the only potential answer which ISA ferry, this answer is selected.

Definition Questions. An important category in CLEF 2005 are questions asking for the definition of a person or organization (i.e. *What is Sabena?, Who is Antonio Matarese?*). No less than 60 questions were of this type. Again, we used the ISA-relations extracted from appositions to answer such questions. More in particular, our strategy for answering definition questions consisted of two phases:

- Phase 1: The most frequent class found for a named entity is selected.
- Phase 2: The sentences which mention the named entity and the class are retrieved and searched for additional information which might be relevant. Snippets of information that are in a adjectival relation or which are a prepositional complement to the class label are selected.

Frequency is important to ensure that an appropriate class is chosen. The named entity *Sabena*, for instance, occurs frequently in the corpus, but often with class labels assigned to it, which are not suitable for inclusion in a definition (*possibility, partner, company,,....* By focussing on the most frequent class label assigned to a named entity (*airline company* in this case), we hope to select the most appropriate label for a definition. A disadvantage of this technique is that the class label by itself is not always sufficient for an adequate definition. Therefore, we expand the class labels with modifiers which typically need to be included in a definition. For the question *What is Sabena?*, our system produces *Belgian airline company* as answer.

4 Evaluation

The results of the CLEF evaluation are given in table 2. The scores are satisfactory for factoid questions and definitions. It is unclear to us at the moment what the explanation is for the fact that the system performed less well on temporally restricted questions.

Of the 140 factoid questions, 46 questions were assigned a type corresponding to a relation table. For 35 of these questions, an answer was actually found in one of the tables. The other 11 questions were answered by using the IR-based strategy as fall-back. 52 of the 60 definition questions were answered by the strategy described in section 3.4. For the other definition questions, the general IR-based strategy was used as fall-back. Three definition questions received NIL as an answer.

Parsing errors are the cause of some wrong or incomplete answers. The question *Who is Javier Solana?*, for instance, is answered with *Foreign Affairs*, which is extracted from a sentence containing the phrase *Oud-minister van buitenlandse zaken Javier Solana* (*Ex-minister of foreign affairs, Javier Solana*). Here, *Javier Solana* was erroneously analyzed as an apposition of *affairs*. Similarly, the wrong answer *United Nations* for the question *What is UNEP?*, which was extracted from a sentence containing *the environment programme of the United Nations (UNEP)*, which contained the same attachment mistake.

A frequent cause of errors were answers that were echoing (part of) the question. Currently, the system only filters answers which are a literal substring of the question. This strategy fails in cases like:

(10) a. Q: Where is Bonn located? A: `in Bonn`.
b. Q: In which city does one find the famous Piazza dei Miracoli? A: `at the Piazza dei Miracoli`
c. Q: In which American state is Iron Mountain located? A: `The United States`.

It seems cases like (10-a) and (10-b) could be easily filtered as well. Cases like (10-c) are harder, as they involve two (near) synonyms. Note finally that not all answers which overlap with the question should be filtered, as the answer in (11) is valid, eventhough the word *rocket* also occurs in the question.

(11) Q: What is the name of the rocket used to launch the satellite Clementine? A: `Titan rocket`

Our strategy for answering definition questions seemed to work reasonably well, although it did produce a relatively large number of inexact answers (of the 18 answers that were judged inexact, 13 were answers to definition questions). As we explained in section 3.4, this is due to the fact that we select the most frequent class label for a named entity, and only expand this label with adjectival and PP modifiers that are adjacent to the class label (a noun) in the corresponding sentence. Given the constituent *the museum Hermitage in St Petersburg*, this strategy fails to include *in St Petersburg*, for instance. We did not include relative clause modifiers, as these tend to contain information which is not appropriate for a definition. However, for the question, *Who is Iqbal Masih*, this leads the system to answer *twelve year old boy*, extracted from the constituent *twelve year old boy, who fought against child labour and was shot sunday in his home town Muritke*. Here, at least the first conjunct of the relative clause should have been included. Similarly, we did not include purpose clauses, which leads the system to respond *large scale American attempt* to the question *what was the Manhattan project*, instead of *large scale American attempt to develop the first (that is, before the Germans) atomic bomb*.

5 Conclusion

We have shown that dependency parsing of both questions and the full document collection is useful for developing an adequate QA system. Dependency patterns can be used to search the corpus exhaustively for answers to frequent question types and for class labels for named entities, which are used to improve the performance of the system on *which*-questions and definition questions. Selection of the most likely answer to a question uses a syntactic similarity metric based on dependency relations.

We have used a limited number of equivalences over dependency relations. An obvious next step is to expand this set with equivalences derived automatically from the parsed corpus (i.e. as in Lin and Pantel (2001)). The syntactic techniques we employ operate exclusively on individual sentences. In the future, we hope to extend this to techniques which operate on the paragraph level by integrating, among others, a component for coreference resolution. Finally, we want to explore the possibility of using dependency relations to boost the performance of the IR-engine, i.e. as in Cui et al. (2005).

References

Attardi, Giuseppe, Antonio Cisternino, Francesco Formica, Maria Simi, and Alessandro Tommasi. 2002. Piqasso: Pisa question answering system. In *Text REtrieval Conference (TREC) 2001 Proceedings*, pages 633–642, Gaithersburg, ML.

Bouma, Gosse, Jori Mur, and Gertjan van Noord. 2005. Reasoning over dependency relations for QA. In *Proceedings of the IJCAI workshop on Knowledge and Reasoning for Answering Questions (KRAQ)*, pages 15–21, Edinburgh.

Cui, Hang, Renxu Sun, Keya Li, Min-Yen Kan, and Tat-Seng Chua. 2005. Question answering passage retrieval using dependency relations. In *Proceedings of SIGIR 05*, Salvador, Brazil.

Fleischman, Michael, Eduard Hovy, and Abdessamad Echihabi. 2003. Offline strategies for online question answering: Answering questions before they are asked. In *Proc. 41st Annual Meeting of the Association for Computational Linguistics*, pages 1–7, Sapporo, Japan.

Jijkoun, Valentin, Jori Mur, and Maarten de Rijke. 2004. Information extraction for question answering: Improving recall through syntactic patterns. In *Coling 2004*, pages 1284–1290, Geneva.

Katz, Boris and Jimmy Lin. 2003. Selectively using relations to improve precision in question answering. In *Proceedings of the workshop on Natural Language Processing for Question Answering (EACL 2003)*, pages 43–50, Budapest. EACL.

Lin, Dekan and Patrick Pantel. 2001. Discovery of inference rules for question answering. *Natural Language Engineering*, 7:343–360.

Litkowski, Kenneth C. 2004. Use of metadata for question answering and novelty tasks. In E. M. Voorhees and L. P. Buckland, editors, *Proceedings of the eleventh Text Retrieval Conference (TREC 2003)*, pages 161–170, Gaithersburg, MD.

Malouf, Robert and Gertjan van Noord. 2004. Wide coverage parsing with stochastic attribute value grammars. In *IJCNLP-04 Workshop Beyond Shallow Analyses - Formalisms and statistical modeling for deep analyses*, Hainan.

Mollá, D. and M. Gardiner. 2005. Answerfinder - question answering by combining lexical, syntactic and semantic information. In *Australasian Language Technology Workshop (ALTW) 2004*, Sydney.

Pantel, Patrick and Deepak Ravichandran. 2004. Automatically labeling semantic classes. In Daniel Marcu Susan Dumais and Salim Roukos, editors, *HLT-NAACL 2004: Main Proceedings*, pages 321–328, Boston, Massachusetts, USA, May 2 - May 7. Association for Computational Linguistics.

Pasça, M. 2004. Acquisition of categorized named entities for web search. In *Proceedings of the Thirteenth ACM conference on Information and knowledge management*, pages 137 – 145.

Pollard, Carl and Ivan Sag. 1994. *Head-driven Phrase Structure Grammar*. Center for the Study of Language and Information Stanford.

Punyakanok, V., D. Roth, and W. Yih. 2004. Mapping dependency trees: An application to question answering. In *The 8th International Symposium on Artificial Intelligence and Mathematics (AI&Math 04)*, Fort Lauderdale, FL.

Tiedemann, Jörg. 2004. A comparison of off-the-shelf IR engines for question answering. Poster presentation at CLIN 2004, Leiden, The Netherlands.

van der Plas, Lonneke and Gosse Bouma. 2005. Automatic acquisition of lexico-semantic knowledge for question answering. In *Proceedings of Ontolex 2005 – Ontologies and Lexical Resources*, Jeju Island, South Korea. To appear.

Vossen, P. 1998. Eurowordnet a multilingual database with lexical semantic networks.

Zobel, Justin, Hugh Williams, Falk Scholer, John Yiannis, and Steffen Hein, 2004. *The Zettair Search Engine*. Search Engine Group, RMIT University, Melbourne, Australia, September.