Reasoning over Dependency Relations for QA*

Gosse Bouma and **Jori Mur** and **Gertjan van Noord**
Information Science
Rijksuniversiteit Groningen
Postbus 716, 9700 AS Groningen
{gosse,mur,vannoord}@let.rug.nl

Abstract

We present a QA system in which question analysis, off-line answer extraction and reranking and identification of potential answers from an IR all make use of syntactic dependency relations. We show that the addition of equivalence rules over patterns of dependency relations, which capture cases where different syntactic patterns express the same semantic relationship, improves the performance of various modules of the system.
Keywords: Reasoning, Language Processing.

1 Introduction

English QA systems can make use of robust wide-coverage parsers [Lin, 1994; Collins, 1999; Briscoe and Carroll, 2002; Clark and Curran, 2004], which either produce dependency relations directly (i.e. tuples of the form ⟨`Head, Rel, Dep`⟩, where `Head` is the head word, `Rel` the name of a dependency relation, and `Dep` the head word of the dependent), or which can be used to derive such tuples. For question analysis (i.e. the task of determining, among others, what the category of the answer to a question is), the use of dependency relations is relatively wide-spread as it tends to produce more accurate results than regular expressions.

The use of dependency information for answer identification (i.e. the task for finding an exact answer in a list of passages returned by an IR system) is much less common. One reason is the fact that parsing of large amounts of text remains a challenge, even for English. Most QA systems make use of an IR step to retrieve a reasonably sized subset of text fragments relevant to the question from the full document collection. Full parsing of these fragments is often not attempted, however. Instead, keyword matching, regular expressions, part-of-speech tagging, and recognition of base constituents is used for finding the exact answer. Notable exceptions are [Katz and Lin, 2003], who processed a text collection (of about 20,000 articles) exhaustively using Minipar, and extracted specific relation tuples from the resulting dependency tuples, and [Litkowski, 2004], who describes a QA system based on a fully parsed corpus, stored in XML. A second issue which has been addressed by several researchers is how to use dependency relations exactly. Several researchers require an exact match between dependency tuples derived from the question and the answer. This is true for the QA systems of both [Katz and Lin, 2003] and [Litkowski, 2004]. The latter uses XPath to generate queries which XML documents must match. Generic metrics, applicable to all question types, have been proposed as well. [Punyakanok *et al.*, 2004] compute the edit distance between the dependency trees of the question and answer, and select answers from sentences which minimize this distance. Comparing trees in a principled way is difficult, and most researchers therefore turn dependency trees (as produced by a parser) into sets of dependency tuples. The PiQASso system [Attardi *et al.*, 2002] and AnswerFinder [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.

Although dependency relations eliminate many sources of variation that systems based on surface strings have to deal with, it is also true that the same semantic relations can sometimes be expressed by several dependency relation patterns. [Lin and Pantel, 2001] show how dependency paths expressing the same semantic relation can be acquired from a corpus automatically. [Rinaldi *et al.*, 2003] argue that such equivalences, or paraphrases, can be especially useful for QA in technical domains.

In this paper, we present a QA system for Dutch, which makes extensive use of dependency relations. We show that an existing, wide-coverage, parser for Dutch can be used effectively for QA by incoporating Named Entity classification and by using a disambiguation model trained on a representative corpus fragment. The use of dependency relations in various parts of the QA system is supported by a module which allows dependency patterns to be (partially) matched against dependency relations. Furthermore, syntactic variation is accounted for by formulating equivalence relations over dependency patterns. We show that both the use of dependency matching in general, and the addition of equivalence rules, improves the performance of our QA system.

*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.

2 In-depth Dependency Analysis for QA

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 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.

For the QA task, the disambiguation model was retrained on a corpus which contained additional material consisting of the (manually corrected) dependency trees of 650 quiz questions.[1] The retrained model achieves an accuracy on 92.7% on the CLEF 2003 questions and of 88.3% on CLEF 2004 questions.

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.[2] 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.[3] The accuracy on unseen CONLL data of the resulting classifier (which combines dictionary look-up and a maximum entropy classifier) is 88.2%.

We have used the Alpino-system to parse the full text collection for the Dutch CLEF QA-task. 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[4] to complete the process in about three weeks. The dependency trees are stored as (25 Gb of) XML. Fortunately, the analyzed material is not only useful for our QA system. It has been used to improve the coverage of the lexicon by error-mining [van Noord, 2004], and is a rich source of data for corpus linguistics. It has also been used by other research groups.[5]

3 Equivalences over Dependency Relations

Several components of our QA system make use of dependency relations. All of these components need to check whether a given sentence satisfies a certain syntactic pattern. We have developed a seperate module for dependency pattern matching, which also accounts for syntactic variation.

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

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

(2) $\left\{ \begin{array}{l} \langle \texttt{krijg/K, obj1, asiel/A} \rangle, \\ \langle \texttt{krijg/K, su, Su/S} \rangle \end{array} \right\}$

The following Prolog program defines when a pattern matches a set of dependency relations:

```
match([],_).
match(Pat,Rels) :-
  equivalent(LHS,RHS),
  partition(Pat,LHS,PatRest),
  partition(Rels,RHS,RelsRest),
  match(PatRest,RelsRest).

equivalent(Pat,Pat).
equivalent(LHS,RHS) :- eq(LHS,RHS).
equivalent(LHS,RHS) :- eq(RHS,LHS).
```

The relation `partition(Union,Set1,Set2)` holds if `Set1` ∪ `Set2` = `Union`, and `Set1` and `Set2` are disjoint. The pattern in (2) matches with the set in (1-b) and would, among others, instantiate the variable `Su` as `mengistu`.

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

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

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

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

[5] The Amsterdam system for CLEF 2004 uses information extracted from the parsed corpus. The University of Nijmegen has used it to obtain realistic data for psycholinguistic experiments.

Equivalences can be defined to account for the fact that in some cases we want a pattern to match a set 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 rule accounts for this:

$$\texttt{eq}(\{\langle \texttt{Vb/V,su,Su/S}\rangle\}, \left\{\begin{array}{l}\langle \texttt{word/W,vc,Vb/V}\rangle,\\ \langle \texttt{Vb/V,mod,door/D}\rangle,\\ \langle \texttt{door/D,obj1,Su/S}\rangle\end{array}\right\})$$

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

We have implemented 13 additional equivalence rules, to account for, among others, coordination, relative clauses, possessive relations expressed by the verb *hebben* (*to have*) as well as the following phenomena:

(4) a. de bondscoach van Noorwegen, Egil Olsen ⇔ Egil Olsen, de bondscoach van Noorwegen (*the coach of Norway, Egil Olsen ⇔ Egil Olsen, the coach of Norway*)
b. Australië's staatshoofd ⇔ staatshoofd van Australië (*Australia's head of state ⇔ head of state of Australia*)
c. president van Rusland, Jeltsin ⇔ Jeltsin is president van Rusland (*president of Russia, Jeltsin ⇔ Jeltsin is president of Russia*).
d. Moskou heeft 9 miljoen inwoners ⇔ de 9 miljoen inwoners van Moskou (*Moskow has 9 million inhabitants ⇔ the 9 million inhabitants of Moskow*).
e. Swissair en Austrian Airlines hebben vluchten naar Kroatië ⇔ Swissair heeft vluchten naar Kroatië (*Swissair and AA have flights into Croatia ⇔ Swissair has flights into Croatia*)
f. Ulbricht liet de Berlijnse Muur bouwen ⇔ Ulbricht, die de Berlijnse Muur liet bouwen. (*Ullbricht had the Berlin Wall be built ⇔ Ullbricht, who had the Berlin Wall be built*)

The equivalence rules we have implemented so far express linguistic equivalences, and thus are both general adn domain independent.

Note that both the `partition` and `equivalent` relation in general will have multiple solutions, and thus, backtracking may be required in order to determine whether a pattern matches a set of dependency relations. Also note that, at least for the moment, for efficiency reasons we do not allow recursive application of equivalence rules. That is, given a solution for `equivalent(LHS,RHS)`, we require that `RHS` is a subset of the set of dependency relations `Rels`, and we do not try to find an equivalent $\texttt{RHS}'$ which is part of `Rels`.

4 Applications in QA

Question analysis. Question analysis is the task of assigning a specific class (`person, location, date, ...`) to a question. Our QA system often assigns an additional argument to the class, i.e. `location(della_alpi_stadion)` asks for the location of the *Della Alpi stadium*. Question analysis requires a combination of syntactic analysis and ontological resources. 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 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 ...* as for a organization named entity.

The advantage of using dependency relations for this task has been widely recognized. The incorporation of equivalences over dependency relations means that patterns will automatically cover a certain amount of syntactic variation. The pattern in (5-a), for instance, ensures that the questions in (5-b) and (5-d) are both classified as `cause(Effect)`, with `Effect` instantiated as `rsi`, in spite of the fact that their word order and syntactic structure differs.

(5) a. $\left\{\begin{array}{l}\langle \texttt{ontsta/O, mod, waardoor/W}\rangle,\\ \langle \texttt{ontsta/O, su, Effect/E}\rangle\end{array}\right\}$
b. Waardoor ontstaat RSI? (*What causes RSI?*)
c. $\left\{\begin{array}{l}\langle \texttt{waardoor/1, wh, ontsta/2}\rangle,\\ \langle \texttt{ontsta/2, mod, waardoor/1}\rangle,\\ \langle \texttt{ontsta/2, su, rsi/3}\rangle\end{array}\right\}$
d. Waardoor kan RSI ontstaan? (*What can cause RSI?*)
e. $\left\{\begin{array}{l}\langle \texttt{waardoor/1, wh, kan/2}\rangle,\\ \langle \texttt{kan/2, su, rsi/3}\rangle,\\ \langle \texttt{ontsta/4, mod, waardoor/1}\rangle,\\ \langle \texttt{ontsta/4, su, rsi/3}\rangle\end{array}\right\}$

The modifier *waardoor* is analyzed as a modifier of the verb *ontstaan* in both cases, and the dependency analysis also makes explicit the fact that the subject of the modal verb *kan* also functions as subject of the verbal complement. A regular expression pattern would at least have to deal with the fact that the subject follows the verb *ontstaan* in (5-b), whereas it precedes the verb in (5-d).

Note also that the use of variables for dependents in a pattern allows identification of specific syntactic arguments in the question. The pattern in (5-a) picks up the subject as the `Effect` for which a cause is asked. This information is used in the answer extraction process.

Off-line answer extraction. Off-line methods have proven to be very effective in QA ([Fleischman *et al.*, 2003], [Jijkoun *et al.*, 2004]). 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 semi-structured table for quick and easy access.

One of the advantages of having the corpus analyzed in full is the fact that it opens up the possibility of off-line answer extraction based on dependency relations. [Jijkoun *et al.*, 2004] have shown for English that using dependency relations for this task can lead to significant improvements in recall over systems based on regular expression pattern matching. Dependency relations often allow patterns to be stated which are hard to capture using regular expressions.

The sentences in (6), for instance, all contain information about organizations and their founders.

(6)
a. **Minderop** richtte **de Tros** op toen .... (*Minderop founded the Tros when...*)
b. Op last van generaal De Gaulle in Londen richtte verzetsheld **Jean Moulin** in mei 1943 **de Conseil National de la Résistance (CNR)** op. (*Following orders of general De Gaulle, resistance hero Jean Moulin founded in May 1943 the Conseil National de la Résistance (CNR)*)
c. **Het Algemeen Ouderen Verbond** is op 1 december opgericht door de nu 75-jarige **Martin Batenburg**. (*The General Pensioners Union was founded on Dec, 1, by, now 75-year old, Martin Batenburg.*)
d. **Kasparov** heeft **een nieuwe Russische Schaakbond** opgericht en... (*Kasparov has founded a new Russian Chess Union and...*)
e. ... 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*).

The verb expressed to relation (*oprichten*, to found) can take on a wide variety of forms (active, with the particle *op* split from the root, participle, and infintival, 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. Such variation is almost impossible to capture accurately using regular expressions, whereas dependency relations can exploit the fact that in 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 (7) suffices to extract this relation from all of the examples above.[6]

(7) $\left\{ \begin{array}{l} \langle \texttt{richt\_op/R, su, Founder/S} \rangle, \\ \langle \texttt{richt\_op/V, obj1, Founded/O} \rangle \end{array} \right\}$

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 (8-a), the equivalence rules illustrated in (4-a), (4-b), and (4-c) can be used to match this single pattern against the alternative formulations in (8-b)- (8-d) as well.

[6]The fact that the by-phrase in passives acts as the logical subject of the participle is accounted for by means of an equivalence rule.

(8)
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*)

The table below illustrates the effect of using equivalence rules when applying the extraction patterns to the full CLEF corpus:

Table	-Equiv		+Equiv		Incr (%)	
	tuples	uniq	tuples	uniq	tpls	uniq
Abbreviation	21.170	8.405	21.497	8.543	1	1
Age	15.981	13.716	22.143	18520	38	35
Born Date	1.545	1.356	2356	1.990	54	47
Born Loc	371	351	937	879	152	150
Capital	1.940	406	2.146	515	10	27
Currency	3.111	124	6.619	222	113	80
Died Age	522	379	1.127	834	116	120
Died Date	374	349	583	544	56	55
Died Loc	364	332	664	583	82	76
Founded	604	559	1.021	953	69	70
Function	54.016	28.543	77.028	46.589	43	63
Inhabitants	529	473	708	633	34	34
Nobel Prize	75	67	169	141	125	110

For all relations, except abbreviations, which are found in a single syntactic environment, the overall number of extracted tuples, as well as the number of unique tuples increases considerably. Of course, for each relation,the number of extracted relations could have been increased by a similar amount by expanding the number of patterns for that relation. The interesting point here is that in this case this was achieved by adding a single, generic component.

The development and maintenance of extraction patterns is further facilitated by the fact that multiple dependency relations may be combined into paths (i.e. $\langle$`City, mod, tellend, me, Inhabitants`$\rangle$ refers to the pair $\langle$`City, mod, tellend`$\rangle$ and $\langle$`tellend, me, Inhabitants`$\rangle$), by providing developers with tools which support visualization of dependency relations as syntactic trees.

Answer reranking and extraction. For those questions that are not answered by the off-line method (i.e. either because no table exists for the given question type, or because no matching table entry was found), the QA system passes a set of keywords extracted from the question to the IR engine. IR returns a set of relevant paragraphs. Within this set, we try to identify the sentence which most likely contains the answer, and we try to extract the answer from the sentence.

For each sentence A, we compute its `a-score` relative to the question Q as follows:

$$\begin{array}{rcl} \texttt{a-score}(Q,A) = & & \alpha.\texttt{d-score}(Q,A) \\ & + & \beta.\texttt{type-score}(Q,A) \\ & + & \gamma.\texttt{IR}(Q,A) \end{array}$$

Here, `d-score` expresses to what extent the dependency relations of Q and A match, `type-score` expresses

whether a constituent matching the question type of Q could be found (in the right syntactic context) in A, and `IR` combines the score assigned by the IR engine and a score which expresses to what extent the proper names, nouns, and adjectives in Q and A plus the sentence immediately preceding A overlap. α, β and γ are (manually set) weights for these scores.

The `d-score` computes to what extent the dependency structure of question and answer match. To this end, the set of dependency relations of the question is turned into a pattern Q, by removing the dependency relations for the question word, and then substituting all string positions by variables. The `d-score` is the cardinality of the largest subset Q' of Q divided by $|Q|$, such that `match`(Q', A) holds (where A is the set of dependency relations of the answer):

$$\texttt{d-score}(Q, A) = \frac{|\underset{Q'}{\arg\max}\ \{Q' | Q' \subset Q \wedge \texttt{match}(Q', A)\}|}{|Q|}$$

Note that dependency matching is considerably more subtle than keyword matching. A case in point are Q/A-pairs such as the following:

(9) a. Wie is voorzitter van het Europese Parlement? (*Who is chair of the European Parliament?*
b. Karin Junkers (SPD), lid van het Europese Parlement en voorzitter van de vereniging van sociaal-democratische vrouwen in Europa...(*Karin Junkers (SPD), member of the European Parliament and chair of the society of social-democrat women in Europe...*)

Here, (9-b) does not contain the correct answer in spite of the fact that it contains all keywords from the question. In fact, even most of the dependency relations of the question are present in the answer, but crucially, there is no substitution for `W` that would satisfy:

$$\texttt{match}(\left\{ \begin{array}{l} \langle \texttt{voorzitter/V,mod,van/W} \rangle, \\ \langle \texttt{van/W,obj1,parlement/X} \rangle \end{array} \right\}, Q)$$

The type-score indicates whether a suitable answer type was found in the sentence (i.e. a date question requires an answer containing a dependent with Part-of-Speech tag `noun(date)`). Higher scores are assigned to potential answers where the phrase that matches the question type is also in some syntactic dependency relation to the topic of the question. I.e. given the question in (10-a), classified as `date(hereniging)`, the date phrase *in oktober 1990* in (10-b) receives a higher score than the date phrase *in 1962* in (10-c), as the first is a dependent of *hereniging*, whereas the second is not.

(10) a. Wanneer vond de Duitse hereniging plaats? (*When did the German unification take place?*)
b. Sinds de Duitse hereniging in oktober 1990... (*Since the German unification in October 1990...*)
c. Al in 1962 voorspelde hij de Duitse hereniging. (*As early as 1962 he predicted the German unification.*)

Merging off-line and IR-based techniques. One of the advantages of our approach is that the technology used for off-line and IR-based answer extraction becomes virtually identical. The only real difference is that the latter relies on IR to make a first selection of relevant paragraphs, whereas the off-line method performs an exhaustive search. Consequently, the metric used for selecting the best answer from a list of results provided by IR can be used for reranking the results of table look-up as well. Cases where this is useful, are questions like (11-a) and (11-b):

(11) a. Wie is de Duitse minister van Economische Zaken? (*Who is the German minister of Economy?*)
b. Wie was president van de VS in de Tweede Wereldoorlog? (*Who was president of the US during the second World War?*)

These are hard to account for by off-line methods. Question (11-a) and (11-b) would be classified as `function(minister,duits)` and `function(president,vs)`, respectively, by question analysis, and thus, in principle can be answered by consulting the functions-table. However, this ignores the modifiers *van Economische Zaken* and *in de Tweede Wereldoorlog*, which, in this case, are crucial for finding the correct answer.

Applying the same scoring technique to facts extracted from the table, as to on-line extracted facts, can help to overcome this problem. In particular, the `d-score` between the question and the sentence on which a table entry is based, can be used as an additional factor (in conjunction with frequency) in determining whether a table answer is correct. For instance, for question (11-a) classified as `function(minister,duits)`, there are several candidate answers, some of which are:

Answer	Freq	Answer	Freq
Klaus Kinkel	54	Volker Rühe	15
Theo Waigel	36	Günter Rexrodt	11

In this case, using frequency only to determine the correct answer, would give the wrong result, whereas a score that combines frequency and `d-score` (based on (12), on which one of the table entries was based) returns the correct answer.

(12) De Duitse minister van economische zaken, Günter Rexrodt, verwelkomde het rapport. (*The German minister of economy, Günter Rexrodt, welcomed the report.*)

5 Evaluation

We evaluated on 572 questions that were used in the Dutch monolingual task of the CLEF 2003 and 2004 QA track. We give the mean reciprocal rank (mrr) over the first 5 answers found by the system. The *off-line* column contains the results for questions answered by table look-up. The *on-line* column contains the score for the remaining questions, and the third column reports the overall score.

CLEF 03	off-line		on-line		total	
	# q	mrr	# q	mrr	# q	mrr
baseline	85	0.71	287	0.36	372	0.44
+d-score	85	0.71	287	0.40	372	0.47
+equiv	89	0.77	283	0.40	372	0.49

CLEF 04	off-line		on-line		total	
	# q	mrr	# q	mrr	# q	mrr
baseline	61	0.56	139	0.42	200	0.46
+d-score	61	0.57	139	0.45	200	0.48
+equiv	71	0.73	129	0.49	200	0.58

The baseline is a system which does not use `d-score` to rerank answers, and which does not use equivalences over dependency relations. It should be noted, however, that the baseline still makes use of dependency relations for question analysis and answer extraction. Adding `d-score` as a factor in reranking answers improves performance. Adding equivalences has a positive effect on the performance of the off-line method in particular. Since more tuples are extracted, the number of questions that can be answered by the off-line method increases. In addition, the accuracy of the overall system improves. The difference between the baseline system and the system using both `d-score` and equivalences is statistically significant (with 95% confidence) in both evaluations according to the paired t-test.

6 Conclusions and Future Work

We have shown that in-depth syntactic analysis in combination with a generic method for matching patterns against dependency relations and dealing with syntactic variation can be used to improve the performance of various components of a QA system. An advantage of this approach is that it opens up the possibility of using similar techniques for off-line and IR-based question answering. In future work, we would like to explore the possibility of integrating equivalence rules based on lexical equivalence (i.e. synonyms, term variants, and paraphrases acquired using the technique described in [Lin and Pantel, 2001]) and coreference.

References

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

[Briscoe and Carroll, 2002] Ted Briscoe and John Carroll. Robust accurate statistical annotation of general text. In *Proceedings the third international conference on Language Resources and Evaluation (LREC 2002)*, pages 1499–1504, Gran Canaria, 2002.

[Clark and Curran, 2004] Stephen Clark and James R. Curran. Parsing the wsj using ccg and log-linear models. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL '04)*, Barcelona, Spain, 2004.

[Collins, 1999] Michael Collins. *Head-driven Statistical Models for Natural Language Processing*. PhD thesis, University of Pennsylvania, 1999.

[Fleischman *et al.*, 2003] Michael Fleischman, Eduard Hovy, and Abdessamad Echihabi. 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, 2003.

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

[Katz and Lin, 2003] Boris Katz and Jimmy Lin. 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, 2003. EACL.

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

[Lin, 1994] Dekan Lin. Principar—an efficient, broad-coverage, principle-based parser. in proceedings of coling-94. In *Proceedings of COLING-94*, pages pp.42–48, Kyoto, Japan., 1994.

[Litkowski, 2004] Kenneth C. Litkowski. 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, 2004.

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

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

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

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

[Rinaldi *et al.*, 2003] Fabio Rinaldi, James Dowdall, Kaarel Kaljurand, Micahel Hess, and Diego Mollá. Exploiting paraphrases in a question answering system. In *The Second International Workshop on Paraphrasing: Paraphrase Acquisition and Applications*, pages 25–32, Sapporo, Japan, 2003.

[van Noord, 2004] Gertjan van Noord. Error mining for wide-coverage grammar engineering. In *Proceedings of the ACL 2004*, Barcelona, 2004.