Answer Sentence Retrieval by Matching Dependency Paths Acquired from Question/Answer Sentence Pairs

Michael Kaisser
AGT Group (R&D) GmbH
Jägerstr. 41, 10117 Berlin, Germany
mkaisser@agtgermany.com

Abstract

In Information Retrieval (IR) in general and Question Answering (QA) in particular, queries and relevant textual content often significantly differ in their properties and are therefore difficult to relate with traditional IR methods, e.g. key-word matching. In this paper we describe an algorithm that addresses this problem, but rather than looking at it on a term matching/term reformulation level, we focus on the syntactic differences between questions and relevant text passages. To this end we propose a novel algorithm that analyzes dependency structures of queries and known relevant text passages and acquires transformational patterns that can be used to retrieve relevant textual content. We evaluate our algorithm in a QA setting, and show that it outperforms a baseline that uses only dependency information contained in the questions by 300% and that it also improves performance of a state of the art QA system significantly.

1 Introduction

It is a well known problem in Information Retrieval (IR) and Question Answering (QA) that queries and relevant textual content often significantly differ in their properties, and are therefore difficult to match with traditional IR methods. A common example is a user entering words to describe their information need that do not match the words used in the most relevant indexed documents. This work addresses this problem, but shifts focus from words to syntactic structures of questions and relevant pieces of text. To this end, we present a novel algorithm that analyses the dependency structures of known valid answer sentence and from these acquires patterns that can be used to more precisely retrieve relevant text passages from the underlying document collection. To achieve this, the position of key phrases in the answer sentence relative to the answer itself is analyzed and linked to a certain syntactic question type. Unlike most previous work that uses dependency paths for QA (see Section 2), our approach does not require a candidate sentence to be similar to the question in any respect. We learn valid dependency structures from the known answer sentences alone, and therefore are able to link a much wider spectrum of answer sentences to the question.

The work in this paper is presented and evaluated in a classical factoid Question Answering (QA) setting. The main reason for this is that in QA suitable training and test data is available in the public domain, e.g. via the Text REtrieval Conference (TREC), see for example (Voorhees, 1999). The methods described in this paper however can also be applied to other IR scenarios, e.g. web search. The necessary condition for our approach to work is that the user query is somewhat grammatically well formed; this kind of queries are commonly referred to as Natural Language Queries or NLQs.

Table 1 provides evidence that users indeed search the web with NLQs. The data is based on two query sets sampled from three months of user logs from a popular search engine, using two different sampling techniques. The "head" set samples queries taking query frequency into account, so that more common queries have a proportionally higher chance of being selected. The "tail" query set samples only queries that have been is-

Set	Head	Tail
Query #	15,665	12,500
how	1.33%	2.42%
what	0.77%	1.89%
define	0.34%	0.18%
is/are	0.25%	0.42%
where	0.18%	0.45%
do/does	0.14%	0.30%
can	0.14%	0.25%
why	0.13%	0.30%
who	0.12%	0.38%
when	0.09%	0.21%
which	0.03%	0.08%
Total	3.55%	6.86%

Table 1: Percentages of Natural Language queries in head and tail search engine query logs. See text for details.

sued less that 500 times during a three months period and it disregards query frequency. As a result, rare and frequent queries have the same chance of being selected. Doubles are excluded from both sets. Table 1 lists the percentage of queries in the query sets that start with the specified word. In most contexts this indicates that the query is a question, which in turn means that we are dealing with an NLQ. Of course there are many NLQs that start with words other than the ones listed, so we can expect their real percentage to be even higher.

2 Related Work

In IR the problem that queries and relevant textual content often do not exhibit the same terms is commonly encountered. Latent Semantic Indexing (Deerwester et al., 1900) was an early, highly influential approach to solve this problem. More recently, a significant amount of research is dedicated to query alteration approaches. (Cui et al., 2002), for example, assume that if queries containing one term often result in the selection of documents containing another term, then a strong relationship between the two terms exist. In their approach, query terms and document terms are linked via sessions in which users click on documents that are presented as results for the query. (Riezler and Liu, 2010) apply a Statistical Machine Translation model to parallel data consisting of user queries and snippets from clicked web documents and in such a way extract contextual expansion terms from the query rewrites.

We see our work as addressing the same fundamental problem, but shifting focus from query term/document term mismatch to mismatches observed between the grammatical structure of Natural Language Queries and relevant text pieces. In order to achieve this we analyze the queries' and the relevant contents' syntactic structure by using dependency paths.

Especially in QA there is a strong tradition of using dependency structures: (Lin and Pantel, 2001) present an unsupervised algorithm to automatically discover inference rules (essentially paraphrases) from text. These inference rules are based on dependency paths, 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" and can be mapped to another path representing e.g. "X solves Y." As such the approach is suitable to detect paraphrases that describe the relation between two entities in documents. However, the paper does not describe how the mined paraphrases can be linked to questions, and which paraphrase is suitable to answer which question type.

(Attardi et al., 2001) describes a QA system that, after a set of candidate answer sentences have been identified, matches their dependency relations against the question. Questions and answer sentences are parsed with MiniPar (Lin, 1998) and the dependency output is analyzed in order to determine whether relations present in a question also appear in a candidate sentence. 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.

(Cui et al., 2005) describe a fuzzy dependency relation matching approach to passage retrieval in QA. 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. As in some of the papers mentioned earlier, a statistical translation model is used, but this time to learn relatedness between paths. (Cui et al., 2004) apply the same idea to 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 similar fashion as in (Cui et al., 2005). The candidate whose paths show the highest matching score is selected. (Shen and Klakow, 2006) also describe a method that is primarily based on similarity scores between dependency relation pairs. However, 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 adaptation of the Dynamic Time Warping algorithm (Rabiner et al., 1991).

(Molla, 2006) presents an approach for the acquisition of question answering rules by applying graph manipulation methods. Questions are represented as dependency graphs, which are extended with information from answer sentences. These combined graphs can then be used to identify answers. Finally, in (Wang et al., 2007), a quasi-synchronous grammar (Smith and Eisner, 2006) is used to model relations between questions and answer sentences.

In this paper we describe an algorithm that learns possible syntactic answer sentence formulations for syntactic question classes from a set of example question/answer sentence pairs. Unlike the related work described above, it acknowledges that a) a valid answer sentence's syntax might be very different for the question's syntax and b) several valid answer sentence structures, which might be completely independent from each other, can exist for one and the same question.

To illustrate this consider the question "When was Alaska purchased?" The following four sentences all answer the given question, but only the first sentence is a straightforward reformulation of the question:

1. The United States purchased Alaska in 1867 from Russia.
2. Alaska was bought from Russia in 1867.
3. In 1867, the Russian Empire sold the Alaska territory to the USA.
4. The acquisition of Alaska by the United States of America from Russia in 1867 is known as "Seward's Folly".

The remaining three sentences introduce various forms of syntactic and semantic transformations. In order to capture a wide range of possible ways on how answer sentences can be formulated, in our model a candidate sentence is not evaluated according to its similarity with the question. Instead, its similarity to known answer sentences (which were presented to the system during training) is evaluated. This allows to us to capture a much wider range of syntactic and semantic transformations.

3 Overview of the Algorithm

Our algorithm uses input data containing pairs of the following:

NLQs/Questions NLQs that describe the users' information need. For the experiments carried out in this paper we use questions from the TREC QA track 2002-2006.

Relevant textual content This is a piece of text that is relevant to the user query in that it contains the information the user is searching for. In this paper, we use sentences extracted from the AQUAINT corpus (Graff, 2002) that contain the answer to the given TREC question.

In total, the data available to us for our experiments consists of 8,830 question/answer sentence pairs. This data is publicly available, see (Kaisser and Lowe, 2008). The algorithm described in this paper has three main steps:

Phrase alignment Key phrases from the question are paired with phrases from the answer sentences.

Pattern creation The dependency structures of queries and answer sentences are analyzed and patterns are extracted.

Pattern evaluation The patterns discovered in the last step are evaluated and a confidence score is assigned to each.

The acquired patterns can then be used during retrieval, where a question is matched against the antecedents describing the syntax of the question.

Input: (a) Query: "When was Alaska purchased?"
(b) Answer sentence: "The acquisition of Alaska happened in 1867."

Step 1: Question is segmented into key phrases and stop words:
`When[1]+was[2]+NP[3]+VERB[4]`

Step 2: Key question phrases are aligned with key answer sentence phrases:

[3]Alaska	→	Alaska
[4]purchased	→	acquisition
ANSWER	→	1867

Step 3: A pre-computed parse tree of the answer sentence is loaded:

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

Step 4: Dependency paths from key question phrases to the answer are computed:
Alaska⇒1867: ⇑pobj⇑prep⇑nsubj⇓prep⇓pobj
acquisition⇒1867: ⇑nsubj⇓prep⇓pobj

Step 5: The resulting pattern is stored:

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

Figure 1: The pattern creation algorithm exemplified in five key steps for the query "When was Alaska purchased?" and the answer sentence "The acquisition of Alaska happened in 1867."

Note that one question can potentially match several patterns. The consequents contain descriptions of grammatical structures of potential answer sentences that can be used to identify and evaluate candidate sentences.

4 Phrase Alignment

The goal of this processing step is to align phrases from the question with corresponding phrases from the answer sentences in the training data. Consider the following example:

Query: "When was the Alaska territory purchased?"

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

The mapping that has to be achieved is:

Query phrase	Answer Sentence phrase
"Alaska territory"	"territory of Alaska"
"purchased"	"acquisition"
ANSWER	"1867"

In our approach, this is a two step process. First we align on a word level, then the output of the word alignment process is used to identify and align phrases. Word Alignment is important in many fields of NLP, e.g. Machine Translation (MT) where words in parallel, bilingual corpora need to be aligned, see (Och and Ney, 2003) for a comparison of various statistical alignment models. In our case however we are dealing with a monolingual alignment problem which enables us to exploit clues not available for bilingual alignment: First of all, we can expect many query words to be present in the answer sentence, either with the exact same surface appearance or in some morphological variant. Secondly, there are tools available that tell us how semantically related two words are, most notably WordNet (Miller et al., 1993). For these reasons we implemented a bespoke alignment strategy, tailored towards our problem description.

This method is described in detail in (Kaisser, 2009). The processing steps described in the next sections build on its output. For reasons of brevity, we skip a detailed explanations in this paper and focus only on its key part: the alignment of words with very different surface structures. For more details we would like to point the reader to the aforementioned work.

In the above example, the alignment of "purchased" and "acquisition" is the most problematic, because the surface structures of the two words clearly are very different. For such cases we experimented with a number of alignment strategies based on WordNet. These 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 words 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 these approaches make use of WordNet::Similarity, a Perl software package that measures 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). Additionally, we developed a custom-built method that assumes that two words are semantically related if any kind of pointer exists between any occurrence of the words root form in WordNet. For details of these experiments, please refer to (Kaisser, 2009). In our experiments the custom-built method performed best, and was therefore used for the experiments described in this paper. The main reasons for this are:

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 only a weak semantic relationship exists. For our purposes however, it is beneficial to only take strong semantic relations into account.

5 Pattern Creation

Figure 1 details our algorithm in its five key steps. In step 1 and 2 key phrases from the question are aligned to the corresponding phrases in the answer sentence, see Section 4 of this paper. Step 3 is concerned with retrieving the parse tree for the answer sentence. In our implementation all answer sentences in the training set have for performance reasons been parsed beforehand with the Stanford Parser (Klein and Manning, 2003b; Klein and Manning, 2003a), so at this point they are simply loaded from file. Step 4 is the key step in our algorithm. From the previous steps, we know where the key constituents from the question as well as the answer are located in the answer sentence. This enables us to compute the dependency paths in the answer sentences' parse tree that connect the answer with the key constituents. In our example the answer is "1867" and the key constituents are "acquisition" and "Alaska." Knowing the syntactic relationships (captured by their dependency paths) between the answer and the key phrases enables us to capture one syntactic possibility of how answer sentences to queries of the form `When+was+NP+VERB` can be formulated.

As can be seen in Step 5 a flat syntactic question representation is stored, together with numbers assigned to each constituent. The numbers for those constituents for which alignments in the answer sentence were sought and found are listed together with the resulting dependency paths. Path 3 for example denotes the path from constituent 3 (the NP "Alaska") to the answer. If no alignment could be found for a constituent, *null* is stored instead of a path. Should two or more alternative constituents be identified for one question constituent, additional patterns are created, so that each contains one of the possibilities. The described procedure is repeated for all question/answer sentence pairs in the training set and for each, one or more patterns are created.

It is worth to note that many TREC questions are fairly short and grammatically simple. In our training data we for example find 102 questions matching the pattern `When[1]+was[2]+NP[3]+VERB[4]`, which together list 382 answer sentences, and thus 382 potentially different answer sentence structures from which patterns can be gained. As a result, the amount of training examples we have available, is sufficient to achieve the performance described in Section 7. The algorithm described in this paper can of course also be used for more complicated NLQs, although in such a scenario a significantly larger amount of training data would have to be used.

6 Pattern Evaluation

For each created pattern, at least one matching example must exists: the sentence that was used to create it in the first place. However, we do not know how precise each pattern is. To this end, an additional processing step between pattern creation and application is needed: pattern evaluation. Similar approaches to ours have been described in the relevant literature, many of them concerned with bootstrapping, starting with (Ravichandran and Hovy, 2002). The general purpose of this step is to use the available data about questions and their correct answers to evaluate how often each created pattern returns a correct or an incorrect result. This data is stored with each pattern and the result of the equation, often called pattern precision, can be used during retrieval stage. Pattern precision in our case is defined as:

$$p = \frac{\#correct + 1}{\#correct + \#incorrect + 2} \tag{1}$$

We use Lucene to retrieve the top 100 paragraphs from the AQUAINT corpus by issuing a query that consists of the query's key words and all non-stop words in the answer. Then, all patterns are loaded whose antecedent matches the query that is currently being processed. After that, constituents from all sentences in the retrieved 100 paragraphs are aligned to the query's constituents in the same way as for the sentences during pattern creation, see Section 5. Now, the paths specified in these patterns are searched for in the paragraphs' parse trees. If they are all found, it is checked whether they all point to the same node and whether this node's surface structure is in some morphological form present in the answer strings associated with the question in our training data. If this is the case a variable in the pattern named *correct* is increased by 1, otherwise the variable *incorrect* is increased by 1. After the evaluation process is finished the final version of the pattern given as an example in Figure 1 now is:

Query:	When[1]+was[2]+NP[3]+VERB[4]
Path 3:	⇑pobj⇑prep⇑nsubj⇓prep⇓pobj
Path 4:	⇑nsubj⇓prep⇓pobj
Correct:	15
Incorrect:	4

The variables $correct$ and $incorrect$ are used during retrieval, where the score of an answer candidate ac is the sum of all scores of all matching patterns p:

$$score(ac) = \sum_{i=1}^{n} score(p_i) \tag{2}$$

where

$$score(p_i) = \begin{cases} \frac{correct_i + 1}{correct_i + incorrect_i + 2} & \text{if match} \\ 0 & \text{no match} \end{cases} \tag{3}$$

The highest scoring candidate is selected.

We would like to explicitly call out one property of our algorithm: It effectively returns two entities: a) a sentence that constitutes a valid response to the query, b) the head node of a phrase in that sentence that constitutes the answer. Therefore the algorithm can be used for sentence retrieval or for answer retrieval. It depends on the application which of the two behaviors is desired. In the next section, we evaluate its answer retrieval performance.

7 Experiments & Results

This section provides an evaluation of the algorithm described in this paper. The key questions we seek to answer are the following:

1. How does our method perform when compared to a baseline that extracts dependency paths from the question?
2. How much does the described algorithm improve performance of a state-of-the-art QA system?
3. What is the effect of training data size on performance? Can we expect that more training data would further improve the algorithm's performance?

7.1 Evaluation Setup

We use all factoid questions in TREC's QA test sets from 2002 to 2006 for evaluation for which a known answer exists in the AQUAINT corpus. Additionally, the data in (Lin and Katz, 2005) is used. In this paper the authors attempt to identify a much more complete set of relevant documents for a subset of TREC 2002 questions than TREC itself. We adopt a cross validation approach for our evaluation. Table 4 shows how the data is split into five folds.

In order to evaluate the algorithm's patterns we need a set of sentences to which they can be applied. In a traditional QA system architecture,

Test	Number of Correct Answer Sentences										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 2: Fraction of sentences that contain correct answers in Evaluation Set 1 (approximation).

Test	Number of Correct Answer Sentences										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 3: Fraction of sentences that contain correct answers in Evaluation Set 2 (approximation).

Fold	Training Data		Test Data	
	sets used	#	set	#
1	T03, T04, T05, T06	4565	T02	1159
2	T02, T04, T05, T06, Lin02	6174	T03	1352
3	T02, T03, T05, T06, Lin02	6700	T04	826
4	T02, T03, T04, T06, Lin02	6298	T05	1228
5	T02, T03, T04, T05, Lin02	6367	T06	1159

Table 4: Splits into training and tests sets of the data used for evaluation. T02 stands for TREC 2002 data etc. Lin02 is based on (Lin and Katz, 2005). The # rows show how many question/answer sentence pairs are used for training and for testing.

see e.g. (Prager, 2006; Voorhees, 2003), 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 the answer in the first place. This also holds true in our case: the patterns cannot be expected to identify a correct answer if none of the sentences used as input contains the correct answer. We therefore use two different evaluation sets to evaluate our algorithm:

1. The first set contains for each question all sentences in 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. For the second set, the query additionally list all known correct answers to the question as parts of one `OR` operator. This increases the chance that the evaluation set actually contains valid answer sentences significantly.

In order to provide a quantitative characterization of the two evaluation sets we estimated the number of correct answer sentences they contain. For each paragraph it was determined whether it contained one of the known answer strings and at least of one of the question key words. Tables 2 and 3 show for 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 is contained in the evaluation set, while column ">= 90" gives the fraction of questions with 90 or more valid answer sentences. The last two columns show mean and median values.

7.2 Comparison with Baseline

As pointed out in Section 2 there is a strong tradition of using dependency paths in QA. Many relevant papers describe algorithms that analyze a question's grammatical structure and expect to find a similar structure in valid answer sentences, e.g. (Attardi et al., 2001), (Cui et al., 2005) or (Bouma et al., 2005) to name just a few. As already pointed out, a major contribution of our work is that we do not assume this similarity. In our approach valid answer sentences are allowed to have grammatical structures that are very different from the question and also very different from each other. Thus it is natural to compare our approach against a baseline that compares candidate sentences not against patterns that were gained from question/answer sentence pairs, but from questions alone. In order to create these patterns, we use a small trick: During the Pattern Creation step, see Section 5 and Figure 1, we replace the answer sentences in the input file with the questions, and assume that the question word indicates the position where the answer should be located.

Test set	Q number	Qs with patterns	> 1 correct	Overall correct	Accuracy overall	Acc. if pattern
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: Performance based on evaluation set 1.

Test set	Q number	Qs with patterns	> 1 correct	Overall correct	Accuracy overall	Acc. if pattern
2002	429	321	239	133	0.310	0.414
2003	354	237	149	88	0.248	0.371
2004	204	142	119	65	0.319	0.458
2005	319	214	161	92	0.288	0.429
2006	352	208	139	84	0.238	0.403
Sum	1658	1122	807	462	0.278	0.411

Table 6: Performance based on evaluation set 2.

Tables 5 and 6 show how our algorithm performs on evaluation sets 1 and 2, respectively. Tables 7 and 8 show how the baseline performs on evaluation sets 1 and 2, respectively. The tables' columns list the year of the TREC test set used, the number of questions in the set (we only use questions for which we know that there is an answer in the corpus), the number of questions for which one or more patterns exist, how often at least one pattern returned the correct answer, how often we get an overall correct result by taking all patterns and their confidence values into account, accuracy@1 of the overall system, and accuracy@1 computed only for those questions for which we have at least one pattern available (for all other questions the system returns no result.) As can be seen, on evaluation set 1 our method outperforms the baseline by 300%, on evaluation set 2 by 311%, taking accuracy if a pattern exists as a basis.

Test set	Q number	Qs with patterns	Min one correct	Overall correct	Accuracy overall	Acc. if pattern
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 7: Baseline performance based on evaluation set 1.

Many of the papers cited earlier that use an approach similar to our baseline approach of course report much better results than Tables 7 and 8. This however is not too surprising as the approach

Test set	Q number	Qs with patterns	Min one correct	Overall correct	Accuracy overall	Acc. if pattern
2002	429	321	77	37	0.086	0.115
2003	354	237	39	26	0.073	0.120
2004	204	142	25	15	0.074	0.073
2005	319	214	38	18	0.056	0.084
2006	352	208	34	16	0.045	0.077
Sum	1658	1122	213	112	0.068	0.100

Table 8: Baseline performance based on evaluation set 2.

described in this paper and the baseline approach do not make use of many techniques commonly used to increase performance of a QA system, e.g. TF-IDF fallback strategies, fuzzy matching, manual reformulation patterns etc. It was a deliberate decision from our side not to use any of these approaches. After all, this would result in an experimental setup where the performance of our answer extraction strategy could not have been observed in isolation. The QA system used as a baseline in the next section makes use of many of these techniques and we will see that our method, as described here, is suitable to increase its performance significantly.

7.3 Impact on an existing QA System

Tables 9 and 10 show how our algorithm increases performance of our QuALiM system, see e.g. (Kaisser et al., 2006). Section 6 in this paper describes via formulas 2 and 3 how answer candidates are ranked. This ranking is combined with the existing QA system's candidate ranking by simply using it as an additional feature that boosts candidates proportionally to their confidence score. The difference between both tables is that the first uses all 1658 questions in our test sets for the evaluation, whereas the second considers only those 1122 questions for which our system was able to learn a pattern. Thus for Table 10 questions which the system had no chance of answering due to limited training data are omitted. As can be seen, accuracy@1 increases by 4.9% on the complete test set and by 11.5% on the partial set.

Note that the QA system used as a baseline is at an advantage in at least two respects: a) It has important web-based components and as such has access to a much larger body of textual information. b) The algorithm described in this paper is an answer extraction approach only. For paragraph retrieval we use the same approach as for evaluation set 1, see Section 7.1. However, 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. In such cases, the simple paragraph retrieval makes it close to impossible for our algorithm to return the correct answer.

Test Set	QuALiM	QASP	combined	increase
2002	0.503	0.117	0.524	4.2%
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 9: Top-1 accuracy of the QuALiM system on its own and when combined with the algorithm described in this paper. All increases are statistically significant using a sign test ($p < 0.05$).

Test Set	QuALiM	QASP	combined	increase
2002	0.530	0.156	0.595	12.3%
2003	0.380	0.093	0.430	13.3%
2004	0.465	0.183	0.514	10.6%
2005	0.388	0.214	0.421	8.4%
2006	0.385	0.149	0.428	11.3%
02-06	0.436	0.157	0.486	11.5%

Table 10: Top-1 accuracy of the QuALiM system on its own and when combined with the algorithm described in this paper, when only considering questions for which a pattern could be acquired from the training data. All increases are statistically significant using a sign test ($p < 0.05$).

7.4 Effect of Training Data Size

We now assess the effect of training data size on performance. Tables 5 and 6 presented earlier show that an average of 32.2% of the questions have no matching patterns. 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, this percentage would decrease and performance would increase. In order to test this assumption, we repeated the evaluation procedure detailed in this section several times, initially using data from only one TREC test set for training and then gradually adding more sets until all available training data had been used. The results for evaluation set 2 are presented in Figure 2. As can be seen, every time more data is added, performance increases. This strongly suggests that the point of diminishing returns, when adding additional training data no longer improves performance is not yet reached.

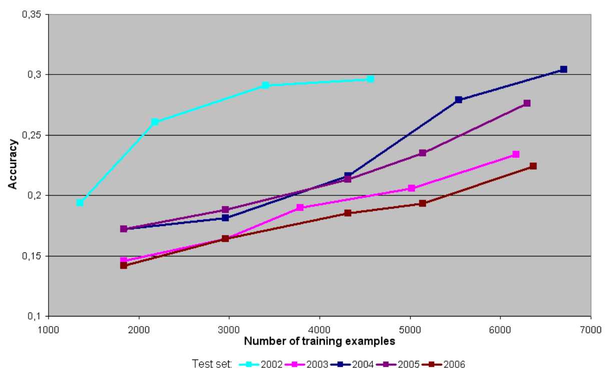

Figure 2: Effect of the amount of training data on system performance

8 Conclusions

In this paper we present an algorithm that acquires syntactic information about how relevant textual content to a question can be formulated from a collection of paired questions and answer sentences. Other than previous work employing dependency paths for QA, our approach does not assume that a valid answer sentence is similar to the question and it allows many potentially very different syntactic answer sentence structures. The algorithm is evaluated using TREC data, and it is shown that it outperforms an algorithm that merely uses the syntactic information contained in the question itself by 300%. It is also shown that the algorithm improves the performance of a state-of-the-art QA system significantly.

As always, there are many ways how we could imagine our algorithm to be improved. Combining it with fuzzy matching techniques as in (Cui et al., 2004) or (Cui et al., 2005) is an obvious direction for future work. We are also aware that in order to apply our algorithm on a larger scale and in a real world setting with real users, we would need a much larger set of training data. These could be acquired semi-manually, for example by using crowd-sourcing techniques. We are also thinking about fully automated approaches, or about using indirect human evidence, e.g. user clicks in search engine logs. Typically users only see the title and a short abstract of the document when clicking on a result, so it is possible to imagine a scenario where a subset of these abstracts, paired with user queries, could serve as training data.

References

Giuseppe Attardi, Antonio Cisternino, Francesco Formica, Maria Simi, and Alessandro Tommasi. 2001. PIQASso: Pisa Question Answering System. In *Proceedings of the 2001 Edition of the Text REtrieval Conference (TREC-01)*.

Gosse Bouma, 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-05)*.

Hang Cui, Ji-Rong Wen, Jian-Yun Nie, and Wei-Ying Ma. 2002. Probabilistic query expansion using query logs. In *11th International World Wide Web Conference (WWW-02)*.

Hang Cui, Keya Li, Renxu Sun, Tat-Seng Chua, and Min-Yen Kan. 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)*.

Hang Cui, Renxu Sun, Keya Li, Min-Yen Kan, and Tat-Seng Chua. 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)*.

Scott Deerwester, Susan Dumais, George Furnas, Thomas Landauer, and Richard Harshman. 1900. Indexing by Latent Semantic Analysis. *Journal of the American society for information science*, 41(6).

David Graff. 2002. The AQUAINT Corpus of English News Text.

Michael Kaisser and John Lowe. 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)*.

Michael Kaisser, Silke Scheible, and Bonnie Webber. 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)*.

Michael Kaisser. 2009. *Acquiring Syntactic and Semantic Transformations in Question Answering*. Ph.D. thesis, University of Edinburgh.

Dan Klein and Christopher D. Manning. 2003a. Accurate Unlexicalized Parsing. In *Proceedings of the 41st Meeting of the Association for Computational Linguistics (ACL-03)*.

Dan Klein and Christopher D. Manning. 2003b. Fast Exact Inference with a Factored Model for Natural Language Parsing. In *Advances in Neural Information Processing Systems 15*.

Jimmy Lin and Boris Katz. 2005. Building a Reusable Test Collection for Question Answering. *Journal of the American Society for Information Science and Technology (JASIST)*.

Dekang Lin and Patrick Pantel. 2001. Discovery of Inference Rules for Question-Answering. *Natural Language Engineering*, 7(4):343–360.

Dekang Lin. 1998. Dependency-based Evaluation of MINIPAR. In *Workshop on the Evaluation of Parsing Systems*.

George A. Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine Miller. 1993. Introduction to WordNet: An On-Line Lexical Database. *Journal of Lexicography*, 3(4):235–244.

Diego Molla. 2006. Learning of Graph-based Question Answering Rules. In *Proceedings of HLT/NAACL 2006 Workshop on Graph Algorithms for Natural Language Processing*.

Franz Josef Och and Hermann Ney. 2003. A Systematic Comparison of Various Statistical Alignment Models. *Computational Linguistics*, 29(1):19–52.

Ted Pedersen, Siddharth Patwardhan, and Jason Michelizzi. 2004. WordNet::Similarity - Measuring the Relatedness of Concepts. In *Proceedings of the Nineteenth National Conference on Artificial Intelligence (AAAI-04)*.

John Prager. 2006. Open-Domain Question-Answering. *Foundations and Trends in Information Retrieval*, 1(2).

L. R. Rabiner, A. E. Rosenberg, and S. E. Levinson. 1991. Considerations in Dynamic Time Warping Algorithms for Discrete Word Recognition. In *Proceedings of IEEE Transactions on Acoustics, Speech and Signal Processing*.

Deepak Ravichandran and Eduard Hovy. 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)*.

Stefan Riezler and Yi Liu. 2010. Query Rewriting using Monolingual Statistical Machine Translation. *Computational Linguistics*, 36(3).

Dan Shen and Dietrich Klakow. 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)*.

David A. Smith and Jason Eisner. 2006. Quasisynchronous grammars: Alignment by Soft Projection of Syntactic Dependencies. In *Proceedings of the HLTNAACL Workshop on Statistical Machine Translation*.

Ellen M. Voorhees. 1999. Overview of the Eighth Text REtrieval Conference (TREC-8). In *Proceedings of the Eighth Text REtrieval Conference (TREC-8)*.

Ellen M. Voorhees. 2003. Overview of the TREC 2003 Question Answering Track. In *Proceedings of the 2003 Edition of the Text REtrieval Conference (TREC-03)*.

Mengqiu Wang, Noah A. Smith, and Teruko Mitamura. 2007. What is the Jeopardy model? A Quasisynchronous Grammar for QA. In *Proceedings of EMNLP-CoNLL 2007*.