Natural Language Engineering **17** (4): 425–454. © Cambridge University Press 2010
doi:10.1017/S1351324910000276

Machine learning for query formulation in question answering

CHRISTOF MONZ
Informatics Institute, University of Amsterdam, Science Park 107, 1098 XG Amsterdam, The Netherlands
e-mail: c.monz@uva.nl

(*Received 18 December 2009; revised 9 August 2010; accepted 25 October 2010; first published online 5 January 2011*)

Abstract

Research on question answering dates back to the 1960s but has more recently been revisited as part of TREC's evaluation campaigns, where question answering is addressed as a subarea of information retrieval that focuses on specific answers to a user's information need. Whereas document retrieval systems aim to return the documents that are most relevant to a user's query, question answering systems aim to return actual answers to a users question. Despite this difference, question answering systems rely on information retrieval components to identify documents that contain an answer to a user's question. The computationally more expensive answer extraction methods are then applied only to this subset of documents that are likely to contain an answer. As information retrieval methods are used to filter the documents in the collection, the performance of this component is critical as documents that are not retrieved are not analyzed by the answer extraction component. The formulation of queries that are used for retrieving those documents has a strong impact on the effectiveness of the retrieval component. In this paper, we focus on predicting the importance of terms from the original question. We use model tree machine learning techniques in order to assign weights to query terms according to their usefulness for identifying documents that contain an answer. Term weights are learned by inspecting a large number of query formulation variations and their respective accuracy in identifying documents containing an answer. Several linguistic features are used for building the models, including part-of-speech tags, degree of connectivity in the dependency parse tree of the question, and ontological information. All of these features are extracted automatically by using several natural language processing tools. Incorporating the learned weights into a state-of-the-art retrieval system results in statistically significant improvements in identifying answer-bearing documents.

1 Introduction

Current question answering systems rely on document retrieval as a means of identifying documents which are likely to contain an answer to a user's question. The documents returned by the retrieval engine are then further analyzed by computationally more expensive techniques to identify an answer to a given question.

The effectiveness of the retrieval component is critical for the performance of a question answering system: If the retrieval system fails to find any relevant documents for a question, further processing steps to find an answer will inevitably fail as well. Ittycheriah *et al.* (2001) have carried out a component-based error analysis and have found that after answer selection most errors of their question answering system are attributable to the information retrieval component.

The role of retrieval for question answering has recently attracted more attention, and in 2005, the TREC evaluation conference devoted a task to evaluate the effectiveness of different retrieval approaches in the context of question answering.

In particular, the way queries are formulated has a strong impact on retrieval effectiveness. While most current, high-performing retrieval approaches, such as the vector space model, or language model are less strict with respect to presence or absence of query terms in documents, they are still sensitive to query formulation. Using all content words from the question for the query can steer the retrieval process in a wrong direction. For example, consider question (1).

(1) What is the abbreviation for the London stock exchange?

It seems natural to include the word *abbreviation* in a query like (2).

(2) `abbreviation London stock exchange`

However, most documents that contain an answer to question (1), express it in the form of '...London Stock Exchange (LSE)...,' not using the term *abbreviation* or one of its morphological variants at all.

Even in vector-space retrieval, a query such as (2) will prefer documents containing the term *abbreviation* over those that do not contain it, although, as discussed above, most documents providing an answer to question (1) do not contain it. The reason is that the term *abbreviation* receives a high term weight during document ranking, because it is much less frequent in the collection than the other terms in (2), and is therefore considered a well-discriminating term for this query. The discriminatory power of a term t is often measured by its inverse document frequency (idf):

$$idf(t) = \log \frac{N}{df(t)}$$

where N is the number of documents in the collection and $df(t)$ is the number of documents in which t occurs. Note that the discussion above is not specific to questions asking for abbreviations but also applies to other questions where the answer type is explicitly mentioned in the question, but unlikely to occur explicitly in a document containing a potential answer, such as 'which person', 'what month', etc.

Our approach on the other hand does not measure the discriminatory power of a term in general, but instead tries to predict the importance of a term *for a given question* by applying machine learning techniques. The learned importance weights are then used to improve the retrieval process.

Learning query term weights is appealing in the context of question answering because a user's information need is expressed as a well-formed sentence as opposed to sets of keywords, used in regular information retrieval. This allows one analyze the question by using natural language processing tools, such as parsers and POS taggers (Hermjakob 2001), to extract features from the question that help predict query term weights.

The remainder of this paper is organized as follows: The next section discusses related approaches. Section 3 discusses the impact of optimal query formulation, and Section 4 investigates how query term weights can be computed. In Section 5, we discuss how words can be represented by features. The model tree learning approach, which we use to learn term weights is introduced in Section 6, and Section 7 presents experimental results. Section 8 provides some conclusions and discusses open issues.

2 Related work

Previous work on query formulation for question answering has mainly been done for Web question answering. Brill, Dumais and Banko (2002) focus on formulating query strings that approximate the way an answer is likely to be expressed. This involves automatically mapping the syntax of an interrogative to the syntax of a declarative sentence. For instance, the AskMSR system (Brill *et al.* 2002) generates for question (3.a) the retrieval queries in (3.b).

(3) a. Where is the Louvre Museum located?

b.
```
''the Louvre Museum is located''
''the Louvre Museum is in''
''the Louvre Museum is near''
''the Louvre Museum is''
Louvre AND Museum AND near
```

Documents are required to either match one of the strings or the boolean query. This approach can be considered rather brute force, trying a number of query variants and the issue of term importance is not addressed.

Paşca (2001) does address the issue of term selection and term relevance. His work is closely related to the work presented in this paper. For query formulation, he distinguishes between three types of terms: high-relevance, medium-relevance, and low-relevance query terms. Deciding which class a given term belongs to is based on a number of rules, some of which are also integrated in our approach. One significant difference to his approach is that we predict the actual importance of a term in a more fine-grained way using numerical values. Note, our predicted term weights can also be negative, indicating that including such a term into the query will harm retrieval performance. Paşca's low relevance category on the other hand just indicates that including the term is unlikely to increase retrieval performance.

Harabagiu *et al.* (2001) use a tight integration of answer extraction and retrieval. Their approach allows for several iterations where the formulation of the retrieval query depends on the success or failure of the answer extraction component during the previous iteration. For each retrieval formulation step the decision which terms to include in a query is based on a number of sophisticated heuristics. Similar to (Brill *et al.* 2002), their approach does not include weights for query terms.

The work by Ittycheriah *et al.* (2001) uses machine learning techniques to rank sentences taken from the list of documents returned by a retrieval engine. Their approach is based on Maximum Entropy classification to score each snippet. Most of the six features used in their method are either count- or distance-based while the approach described here also uses syntactic and ontological features.

Although machine learning techniques have been used before to find answer extraction patterns, see, e.g., (Ravichandran and Hovy 2002; Lita and Carbonell 2004; Yousefi and

Kosseim 2006), it has rarely been applied before to query formulation in the context of question answering. Radev *et al.* (2001) use an expectation maximization approach to learn paraphrases of question for query formulation, where query reformulations are learned iteratively from retrieved documents.

On the other hand, machine learning has been applied to query formulation in the context of *ad hoc* retrieval. Cooper, Chen and Gey (1993) use logistic regression to assign weights to matching clues, such as the number of times a term occurs in the query, the number of times a query term occurs in a document, the idf score of a matching term, and the number of distinct terms common to both query and document. In addition, they assigned weights to query terms in case some relevance information is available, as is the case in document routing or feedback retrieval. Chen *et al.* (1998) did apply machine learning techniques for selecting query terms, but it was done in the context of relevance feedback retrieval. More recently, Bendersky and Croft (2008) developed an approach to identify the most important concepts in verbose *ad hoc* retrieval queries. Their approach used purely statistical features in addition to a simple capitalization indicator.

The work by Lee *et al.* (2009), is to some extent related to the work described here as it uses regression to rank terms in queries according to their importance. On the other hand, their approach uses this information to expand queries, while our approach is concerned with term selection.

The approaches by Mayfield and McNamee (2005) and Agichtein, Lawrence and Gravano (2004) are to some extent complementary to our work as they expand retrieval queries with terms or phrases that are likely to be found in the context of phrases which are of the expected answer type. While the approach in Mayfield and McNamee is based on simple co-occurrence statistics, Agichtein *et al.* also incorporate the BM25 retrieval weighting scheme to assign weights to phrases used for expanding the original query.

The work by Bilotti *et al.* (2007) on query formulation focuses on structured queries that reflect the argument structure of the question and the corresponding argument structure in the answer-bearing sentences.

3 Optimal query term selection

The first step is to estimate the effect query formulation, in the form of term selection, can have on retrieval performance. To this end we use the TREC-9, TREC-10, and TREC-11 data sets consisting of 500 different questions each and the AQUAINT document collection. At TREC-9 and TREC-10, participants were required to return up to five answer-document-id pairs for each question, where the answer can be any text string containing maximally 50 characters, and the document-id refers to the document from which the answer was extracted. At TREC-11, participants were required to return one answer-document-id pair for each question.

For evaluating the retrieval effectiveness, we use NIST's judgment files. A judgment file indicates for each submitted answer-document-id pair, whether the answer is correct. Questions with no correct answer were disregarded, as we are unable to tell whether no correct answer for this question exists in the collection or whether all of the participants failed to return a correct answer.

The set of documents that are known to contain a correct answer form the gold standard for our evaluation, which contains 480, 433 and 455 questions for TREC-9, TREC-10, and TREC-11, respectively.[1] Using the TREC judgment files resulted in the following average numbers of relevant documents: 10.7 (TREC-9, standard deviation: 12.4), 9.2 (TREC-10, standard deviation: 10.1), and 4.4 (TREC-11, standard deviation: 4.0).

Although our gold standard contains only a portion of all the documents containing an answer, Keenan, Smeaton and Keogh (2001) conclude that effective systems are likely to distinguish themselves using only a portion of all relevant documents for evaluation.

In order to compute the optimal term selection for each question, we compare all possible ways of selecting terms from a question. That is, given a question q in which the set of terms T occurs, we consider all possible subsets of T, and evaluate the respective performances. More formally, the set of term selection variants (*tsv*) is defined as:

(4) $$tsv(q) = POW(T) - \{\emptyset\}$$

where $POW(T)$ is the power set, the set of all subsets, of the set T. Obviously, the empty subset is disregarded. Consider question (5.a), which contains the morphologically stemmed terms in (5.b).

(5) a. What is the chemical formula for sulphur dioxide?
  b. $T = \{\texttt{chemic}, \texttt{dioxid}, \texttt{formula}, \texttt{sulphur}\}$

Since $|T| = 4$, there are $2^4 - 1 = 15$ term selection variants. For each query variant a retrieval process is carried out to compute the average precision. Retrieving documents for all variants of long questions can become computationally very expensive. On the other hand this did not affect our experiments as the questions in the TREC data have a limited number of content words: 3.46 (TREC-9, standard deviation: 1.51), 2.85 (TREC-10, standard deviation: 1.47), and 3.72 (TREC-11, standard deviation: 1.39). Since the TREC data sets are used to estimate the parameters offline, this does not affect the applicability of the resulting model to longer questions.

In the actual retrieval queries, all terms are required to be present in a document. For instance, the retrieval query corresponding to (5.b) is

```
chemic AND dioxid AND formula AND sulphur
```

Note that like all retrieval approaches that do not use query expansion or some form of concept-based matching this approach will not find documents that do not contain the actual words from the query but some of their synonyms. Here, we restrict ourselves to surface matches as expanding query words by their synonyms will further increase the number of selection variants and previous studies, see, e.g., (Voorhees 1994), have shown that query expansion with synonyms can lead to decreases in performance.

Table 1 lists all possible selection variants for question (5.a) sorted by their respective average precision. The query variants can be evaluated with respect to a number of evaluation measures. Here, we used average precision, because it is widely used and combines precision and recall. Given a query q, its set of relevant documents REL_q, and a ranking of

[1] The original TREC-9 data set contains 243 additional questions that are reformulations of the questions in the main TREC-9 set. For our experiments, these variants were disregarded.

Table 1. *Performances of term selection variants*

Rank	Avg. precision	Query variant
1	0.0285	`dioxid, sulphur`
2	0.0196	`chemic, dioxid, sulphur`
3	0.0180	`sulphur`
4	0.0086	`chemic, dioxid`
5	0.0078	`dioxid`
6	0.0032	`chemic, sulphur`
7	0	`chemic, formula`
8	0	`chemic, formula, sulphur`
9	0	`chemic, dioxid, formula, sulphur`
10	0	`dioxid, formula`
11	0	`formula`
12	0	`formula, sulphur`
13	0	`chemic`
14	0	`dioxid, formula, sulphur`
15	0	`chemic, dioxid, formula`

Table 2. *Comparison of the f@n scores of optimal retrieval queries to baseline runs. As these are error rates, lower values are better*

	TREC-9			TREC-10			TREC-11		
f@n	Base	Opt	Red.	Base	Opt	Red.	Base	Opt	Red.
5	0.300	0.177	(−41.0%)$^▼$	0.351	0.251	(−28.5%)$^▼$	0.477	0.310	(−35.0%)$^▼$
10	0.215	0.110	(−48.8%)$^▼$	0.266	0.185	(−30.5%)$^▼$	0.374	0.233	(−37.7%)$^▼$
20	0.155	0.079	(−49.0%)$^▼$	0.199	0.113	(−43.2%)$^▼$	0.295	0.176	(−40.3%)$^▼$
50	0.086	0.044	(−48.8%)$^▼$	0.125	0.076	(−39.2%)$^▼$	0.205	0.119	(−42.0%)$^▼$

documents ($rank_q : D \rightarrow \mathbb{N}$) resulting from the retrieval process, the average precision of an individual query is defined as:

$$\text{avg\_prec}(q) = \frac{\sum_{d \in \text{REL}_q} p@rank_q(d)}{|\text{REL}_q|} \tag{6}$$

where $p@rank_q(d)$ returns the ratio of relevant documents that are ranked at least as high as document d in the ranking returned by the retrieval system. If d is not included in the list of returned documents, $p@rank_q(d)$ returns 0.

The total number of query variants for all queries of the three data sets are 7,587 (TREC-9), 6,975 (TREC-10), and 11,957 (TREC-11). In order to determine the maximum reductions in failure rate one could theoretically gain from query formulation, we determined the query variant with the highest average precision for each question. Table 2 shows the gains that can be achieved by using an oracle to pick the optimal query variant.

In *ad hoc* retrieval, a number of metrics can be used to evaluate the retrieval effectiveness, where the choice depends on the application. For instance, in web retrieval it is common to measure precision at 10, as most search engine users only look at the first page of results. For evaluating retrieval systems without a specific scenario in mind it is common to use mean average precision, which is defined as the average of avg_prec(q), as defined in (6), over a set of queries. In question answering it is common to compute the mean reciprocal rank (MRR) which is defined as $1/r_h$, where r_h is rank of the highest-ranking correct answer for a given question. MRR severely penalizes differences in high-ranking positions, while being less discriminative for lower ranking answers. As the *exact* position in the ranking is less important in order to determine the effectiveness of different document retrieval approaches in the context of question answering we use failure-at-n (f@n) which is an error rate measuring the percentage of questions for which the system failed to return at least one document containing an answer in the top-n ranked documents. Focusing on the the number of questions for which the retrieval system failed (or succeeded) to find relevant documents up to a certain cut-off is commonly used; see, e.g., (Thompson *et al.* 1996; Roberts and Gaizauskas 2004). For a specific question q, f@$n(q)$ is a binary-valued function.

(7) $$\text{f@}n(q) = \begin{cases} 0 & \text{if } |\{d \in \text{REL}_q : rank_q(d) \leq n\}| \geq 1 \\ 1 & \text{otherwise} \end{cases}$$

The results shown in Table 2 were achieved by comparing the optimal query variant (Opt), to a commonly used vector-space retrieval approach (Lnu.ltc); see (Singhal *et al.* 1996), which is discussed in more detail in Section 7.[2]

As one could expect, query formulation has a significant impact on the overall performance of a retrieval system, even if query formulation is just based on term selection without expanding the queries with semantically related terms. This comparison shows that much can be gained from better query formulation, but, of course, the problem of identifying an optimal query without having any relevance assessments remains open. In the remainder we explore ways to solve this issue.

To determine whether the observed differences between two retrieval approaches are statistically significant and not just caused by chance, we used the bootstrap method, a powerful nonparametric inference test (Efron 1979). Bootstrapping is more appropriate than the t-test since it does not assume a normal distribution; a condition that is frequently not met in information retrieval (Wilbur 1994). Two retrieval methods a and b are compared by one-tailed significance testing. Relative reductions in $f@n$ that are statistically significant at a confidence level of 95 per cent are marked with '$^\triangledown$' and at a confidence level of 99 pre cent with '$^\blacktriangledown$'. No mark-up indicates no significant decreases in error rate.

4 Computing query term weights

Our approach is to use the different query variants of a question to distinguish between terms that help retrieve relevant documents, and terms that harm the retrieval effectiveness for that particular question. In the previous section, we considered only one single best-performing query variant for determining the potential gains in retrieval performance, but often there are several almost equally well-performing query variants. Looking at the ranked query variants reveals that some terms occur more frequently in higher-ranked

[2] We have also experimented with the probabilistic Okapi BM25 measure (Robertson, Walker and Beaulieu 1998), but the results were almost identical to the ones using Lnu.ltc.

Table 3. *Example term weights*

t	$w_+(t)$	$w_-(t)$	$Gain(t)$
Sulphur	0.808	0.192	0.616
Dioxid	0.752	0.248	0.506
Chemic	0.367	0.633	−0.266
Formula	0.000	1.000	−1.000

variants than other terms. Consider for instance Table 1, where the terms *dioxid* and *sulphur* occur more often in high-ranked variants than *chemic* and *formula* which occur only in variants that did not retrieve any relevant documents.

An analysis of the distribution of query terms over the ranked query variants allows one to assign a weight to each query term: If a term occurs mainly in query variants that have a high average precision it should receive a high weight, whereas a term that occurs mainly in query variants that have a low average precision should receive a low weight. Thus, the weight of a query term depends on two factors: The average precisions of the query variants in which the term occurs (its presence weight: $w_+(t)$), and the average precisions of the query variants in which the term does not occur (its absence weight: $w_-(t)$). Presence and absence weights are normalized by the sum of the average precisions of all query variants, so the weights will range between 0 and 1.

Given a question q and all its query variants $tsv(q)$, the *presence weight* of term t ($w_+(t)$) is computed as:

$$w_+(t) = \frac{\sum\limits_{q' \in tsv(q) \wedge t \in q'} \text{avg\_prec}(q')}{\sum\limits_{q' \in tsv(q)} \text{avg\_prec}(q')} \tag{8}$$

Analogously, the *absence weight* of term t ($w_-(t)$) is computed as:

$$w_-(t) = \frac{\sum\limits_{q' \in tsv(q) \wedge t \notin q'} \text{avg\_prec}(q')}{\sum\limits_{q' \in tsv(q)} \text{avg\_prec}(q')} \tag{9}$$

The presence and absence weights of a term t, can be combined into a single weight by subtracting the absence weight from the presence weight, which we call the *gain* of term t: $gain(t) = w_+(t) - w_-(t)$. If a query term has a positive gain it should be included in the query. If the gain is negative is should be excluded.

Let us return to question (5.a) and its query variants in Table 1. The presence and absence weights, as well as the gain of each term, are shown in Table 3. The gains of the terms are in line with the earlier observation that *sulphur* and *dioxid* are better query terms than *chemic* and *formula*.

This approach of computing term weights assumes that terms occur independently of each other. This assumption does not hold in practice, but it is commonly used in information retrieval and allows us to simplify the computation of term weights.

Table 4. *List of features for question words*

	Feature	Values
1.	`part-of-speech`	Fixed list of POS tags from the Penn Treebank tag set
2.	`location`	Boolean (is the word is part of a location name?)
3.	`question focus`	Boolean (is the word part of the question focus?)
4.	`abbreviation`	Boolean (is the word an abbreviation?)
5.	`superlative`	Boolean (does the question contain a superlative adjective?)
6.	`upper case`	Boolean (does the word start with an uppercase letter?)
7.	`question class`	A fixed list of question classes
8.	`classif. word`	Boolean (was the word used to classify the question?)
9.	`multpl. occurr.`	Boolean (does the word occur more than once in the question?)
10.	`person name`	A fixed set of values indicating what part of a person's name the word is, if applicable
11.	`quoted`	Boolean (does the word occur between quotation marks?)
12.	`honorific`	Boolean (is the word a honorific term (e.g., Dr.)?)
13.	`modified noun`	Boolean (is the word a noun that is preceded (modified) by another noun?)
14.	`no. edges`	A natural number indicating the number of edges pointing to a word in the dependency parse graph of the question
15.	`term ratio`	$1/m$, where m is the number of unique terms in the question
16.	`hypernym`	Boolean (is the word a WordNet hypernym of another word in the question?)
17.	`no. leaves`	The number n ($n \geq 0$) of hyponyms of the word in the WordNet hierarchy that do not have any further hyponyms themselves
18.	`relative idf`	A real value indicating the relative frequency of the word in the document collection compared to the frequencies of the other words in the question

5 Representing terms by feature sets

In the previous section, the computation of the term weights was based on the distribution of the terms themselves over the query variants. This is problematic for two reasons. Firstly, the same term can have a high gain in one query, and a low gain in another. Secondly, if the learning algorithm is based on the surface terms themselves, it cannot assign weights to terms that did not occur in the training data. The first problem is a direct consequence of the term independence assumption. It could be solved by conditioning the weight of a term on a number of terms that also occur in the question, but then data sparseness becomes a serious issue.

One way to address both problems is to represent terms and their contexts in a more abstract manner. Here, we use a set of features that represent certain characteristics of a term and its role in a question. The list of features contains information about the term's part-of-speech, whether it semantically includes other terms in the question, the type of question it occurs in, etc. As mentioned above, some of the features capture aspects inherent to a term, such as part-of-speech, while others capture contextual aspects, such as semantic inclusion. Table 4 lists all 18 features used in our experiments.

We will now discuss all the features used here in more detail. Some of these features can also be found elsewhere in the literature, see, e.g., (Paşca 2001). In particular, the specification of the features *question focus*, *superlative*, *quoted*, *number of leaves*, *modified noun*, and *person name* is based on (Paşca 2001).

Part-of-Speech. The part-of-speech feature can take values such as `NNP` (proper name, singular), `JJS` (superlative adjective), `VBZ` (verb in present tense, third person singular), etc. These are the standard part-of-speech texts from the Penn Treebank (Santorini 1990).

Part-of-speech tagging is accomplished by using TreeTagger (Schmid 1994), a decision-tree-based tagger. The general parameter setting of TreeTagger, which is based on a newspaper training set, turned out to be inappropriate for tagging questions. This is due to the difference in word order between interrogative and declarative sentences. In order to improve the performance of TreeTagger for questions, we trained it on 700 questions, 328 of which were taken from the Penn Treebank corpus,[3] and the remaining 482 were questions from the TREC-9 data set. Whereas the Penn Treebank questions were already manually part-of-speech tagged, we had to tag the questions from the TREC-9 data set ourselves. Although we did not evaluate the increase in performance of the tagger, inspecting randomly chosen questions indicated a clear improvement in tagging accuracy.

The actual values of the `part-of-speech` feature are a slight simplification of the Penn Treebank tags. For example, we do not make a distinction between singular and plural (proper) nouns, i.e., `NNP` and `NNPS` are mapped onto `NNP`, and `NN` and `NNS` are mapped onto `NN`. Also, the different inflections of a verb are disregarded, and all verb forms are represented by the single tag `V`.

Question Focus. The focus of a question is a phrase describing the ontological superclass (i.e. hypernym) of the answer.[4] For example, in question (10), the focus is *country*, in (11), it is *peninsula*, and in (12), it is *college*.

(10) In what country did the game of croquet originate?
(11) What is a peninsula in the Philippines?
(12) What college did Magic Johnson attend?

The answer to question (10), which is *France*, is an instance of *country*, i.e., *France* is a *country*; analogously for the other two examples.

Whether a word is part of the question focus has consequences for formulating the query, as many documents containing an answer to the question do not make this instance relation explicit. For instance, the fact that France is a country is considered to be common knowledge and therefore seldomly stated explicitly in a document. Hence requiring a document to contain words from the question focus can harm retrieval.

The question focus feature can take three values: 0, 0.5 and 1. If a word is not part of the question focus, the feature is set to 0. If a word is part of the question focus and the semantic head of a noun phrase, it is set to 1. For words that are part of the question focus, but are not the semantic head of a noun phrase, the feature is set to 0.5. For instance in question (13), the focus feature is set to 1 for the noun *explorer*, which is the semantic head of the

[3] Distributed by the Linguistic Data Consortium: `http://www.ldc.upenn.edu/`.
[4] Note that the term *question focus* in the way it has been used in the literature on question answering does not necessarily coincide with its definition in the linguistic literature. In question answering, the question focus is sometimes also referred to as *answer type term*.

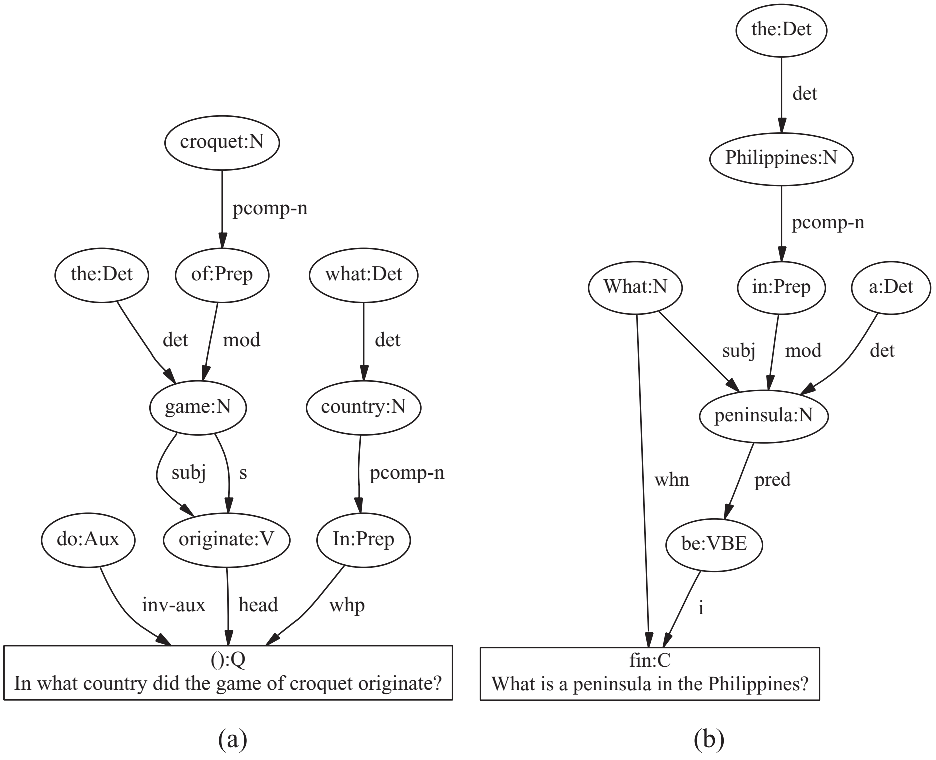

Fig. 1. Examples of MINIPAR dependency graphs for questions.

noun phrase *Spanish explorer*, and the focus feature is set to 0.5 for the modifying adjective *Spanish*.

(13) What Spanish explorer discovered the Mississippi River?
(14) What mythical Scottish town appears for one day every 100 years?

For question (14), the focus feature is set to 0.5 for both modifying adjectives *mythical* and *Scottish*. We do not make a distinction whether a modifying word immediately precedes the head or whether there are more words between them. The semantic head is simply identified as the rightmost word of the noun phrase in the question focus, cf. (Williams 1981).

In order to determine the focus of a question we used MINIPAR (Lin 1998), which is a robust, full dependency parser and able to cover about 87 per cent of the dependency relationships in the SUSANNE evaluation corpus (Sampson 1995) with about 89 per cent precision, cf. (Lin and Pantel 2001). A directed arc in a dependency graph going from node *x* to node *y* means that node *x* modifies node *y*. The arcs in the dependency graph carry labels that indicate the type of modification. To determine the question focus, we use only a small portion of the dependency graph of a question. In particular, we focus on the outgoing arcs of nodes representing the wh-words *what* and *which*, because they modify the question focus. In Figure 1, we show the dependency graphs that are generated by MINIPAR for question (10) and (11).[5] The dependency graph shown in Figure 1(a), illustrates a trivial situation, where the wh-word *what* modifies the noun *country* as a determiner. In question (11), focus determination is slightly more complicated. In the corresponding dependency graph, see Figure 1(b), the wh-word *what* modifies the noun *peninsula* as a subject. Note that in the MINIPAR representations, the `subj` (subject) arcs refer to the logical subject, and not to the syntactic subject. Two kind of modifier relations are used to identify the question focus: `det` (determiner), and `subj` (subject).

Superlative. Despite our earlier comments, under certain circumstances, words from the question focus can be relevant for query formulation. In particular, if the question contains a superlative. In question (15), the focus *island*, and in question (16), the focus *lake* are modified by a superlative adjective.

(15) What is the world's second largest island?
(16) What is the deepest lake in America?

In these cases, the instance relation, viz. that *New Guinea* is an island and that *Crater* is a lake, is likely to be expressed explicitly in the document containing an answer, e.g., ... *Crater Lake is America's deepest lake*. In order to be able to make this distinction the feature `superlative` indicates whether the question contains a superlative adjective.

Detecting whether a question contains a superlative adjective relies on the output of the part-of-speech tagger. If there is at least one word which is tagged as `JJS`, which is the Penn Treebank tag for superlative adjective, the `superlative` is set to 1, and 0 otherwise.

Question Class. The decision to select a term for query formulation is to some extent also based on the type of question. Question classes, or question types, specify the kind of answer the question is asking for. Question types include categories such as `date` (when did something happen?), *location* (where is something?), *agent* (who did something?), etc. For example, consider questions (17) and (18).

(17) Who started the Protestant reformation?
(18) When did the Black Panther party start in California?

Both questions contain the term *start*, but question (17) is of type `agent`, and question (18) is of type `date`. It turns out that including the term *start* is more important in formulating the retrieval query for question (17), where the gain for *start* is 1.0, than for question (18), where the gain for *start* is 0.094. This might be due to the fact that starting dates can be expressed in several ways.

To identify the question class, often also referred to as the *question target*, pattern matching is applied to assign to a question one of 33 classes. In total, a set of 102 patterns is used to accomplish this. Some of the patterns used are shown in Table 5.

The patterns are applied in an ordered manner so that more specific patterns match first. These patterns were generated manually by inspecting a sample of the TREC questions. Of course, one could imagine several types of questions that are not necessarily well-captured by these types, but since our approach is applied in the context of question answering as

[5] The graphs were constructed using AT&T's Graphviz graph displaying tool: `http://www.research.att.com/sw/tools/graphviz/`.

Table 5. *Example patterns for question classification*

Question target	Example patterns
`name`	`/[Ww]hat( wa| i|\')s the name/`
`pers-def`	`/[Ww]ho( wa| i|\')s [A-Z][a-z]+/`
`thing-def`	`/[Ww]hat( wa| i|\')s an? /`
`pers-ident`	`/[Ww]ho( wa| i|\')s the/`
`thing-ident`	`/[Ww](hat|hich)( wa| i|\')s the /`
`number`	`/[Hh]ow (much|many) /`
`expand-abbr`	`/stand(s)? for( what)?\s*?/`
`find-abbr`	`/[Ww]hat( i|\')s (the|an) (acronym|abbreviation) for`
`agent`	`/[Ww]ho /, / by whom[\.\?]/`
`object`	`/[Ww]hat (did|do|does) /`
`known-for`	`/[Ww]hy .+ famous/ /[Ww]hat made .+ famous/`
`aka`	`/[Ww]hat( i|\')s (another|different) name /`
`name-instance`	`/Name (a|one|some|an) /`
`location`	`/[Ww]here(\'s)? /, / is near what /`
`date`	`/[Ww]hen /, /[Ww](hat|hich) year /`
`reason`	`/[Ww]hy /`
`what-np`	-
`unknown`	-

defined in the TREC question answering campaign, we focus on those questions that are typically present in the TREC data sets. Note that this pattern matching approach is rather simple, but nevertheless quite effective for our purposes here, as the question class is just used as one of the features for query formulation, and the goal of our approach is not answer extraction, where accuracy of the question classifier is more important. For more sophisticated approaches to question classification see (Harabagiu, Paşca and Maiorano 2000; Ittycheriah *et al.* 2001; Li and Roth 2006).

Multiple Occurrences. As mentioned above, if a word occurs in the question focus, including it in the query may harm retrieval performance. For instance, the query for question (19) should not contain the term *state*.

(19) Which U.S. state is the leading corn producer?
(20) What state is the geographic center of the lower 48 states?

On the other hand, if a word occurs in the question focus and also outside of it, excluding that term from the query may harm the results. In question (20), the term *state* occurs twice. Note, that although the second occurrence is the plural form of *state*, after morphological normalization, such as stemming, both occurrences are mapped to the same term. Whether a word occurs more than once is captured by the boolean feature `multiple occurrences`.

Quoted. Words that occur between quotation marks require special consideration. Quoted phrases often refer to titles of movies or theater plays, nicknames, etc. Many words that do not bear much content, and therefore are not very helpful for retrieval, are critical for retrieval if they occur in a quotation. For instance, in question (21), the words *gone*, *with*, and *the*, are highly frequent terms.

(21) What is the name of the heroine in ‘Gone with the Wind’?

However, not selecting them for query formulation would only leave the word *wind* as a description of the movie title, which is certainly insufficient. In order to distinguish between words that occur in a quotation and those that do not, the boolean feature `quoted` is set to 1 if a word is quoted and 0 otherwise.

Number of Leaves. As discussed above, including words that appear in the question focus often harms retrieval. But the extent to which including question focus words harms retrieval, also depends on the generality of the word. For instance, in question (22), the question focus is *person*, which is a very general term, and including it in the query is likely to harm retrieval.

(22) What person developed COBOL?
(23) What Spanish explorer discovered the Mississippi River?

In question (23), the term *explorer* is rather specific and it is likely that the answer document makes the fact explicit that *Hernando de Soto* (the correct answer to the question), is an explorer. Whereas it is rather unlikely that a document containing the answer to question (22) explicitly states that *Grace Hopper* (the correct answer) is a person.

There are several ways to measure the generality of a term. Here, we use WordNet to count the number of concepts that are hyponyms of the question focus and that do not have any hyponyms themselves. A concept x is a hyponym of concept y, if x is a y. If the question focus is ambiguous, i.e., it belongs to several concepts, we take the sum of all hyponyms of all concepts the word belongs to. The feature `no. leaves` provides the number of hyponyms. For instance, the term *person* occurs in three concepts in WordNet, which in total have 5,765 leaves, whereas the term *explorer* occurs in one concept, which has 3 leaves. One can conclude that the term *person* is much more general than the term *explorer*, and hence less likely to occur explicitly in an answer document.

Term Ratio. A more general aspect of query formulation is the length of the original question. If a question contains many words, leaving one out in formulating the query has less of an impact on the effectiveness of the retrieval process than for questions that contain only two or three terms. The feature `term ratio` expresses the length of the original question (after removing general stop words), as its reciprocal: $1/m$, where m is the number of words in the question.

Classifying Word. Certain words are good indicators for classifying a question. For instance, in question (24), the word *abbreviation* in combination with the word *mean* indicates that the question is of type `expand-abbreviation`.

(24) What does the abbreviation WASP mean?
(25) What is the height of the tallest redwood?
(26) What province is Calgary located in?

Similarly, the word *height* in question (25) indicates that the question is of type `height`, and the word *located* that question (26) is of type `location`. However, words that are good indicators for question classification, are less frequently used in answers occurring in the documents. For example, it is rather unlikely that the word *located* is used in a declarative sentence that answers question (26).

Whether a word is a classifying word depends also on the question category of the question at hand. If the question category is `expand-abbr`, the words *stand*, *abbreviation*, and *mean* are classifying words, but if the question category is `known-for`, the words *famous* and *made* are classifying words, see Table 5. The classifying words are extracted from the patterns that are used for question classification.

Location. The location feature indicates whether a word is part of a location name. For many questions, it is essential to include words that are part of a location name in order to find an answer. For instance, in question (27), the location words *San*, *Antonio*, and *TX* are relevant terms as the question refers to the temperature of that particular location.

(27) What is the highest recorded temperature in San Antonio, TX?
(28) When was the Buckingham Palace built in London, England?

On the other hand, in question (28), the location words *London* and *England* seem to be superfluous as it is relatively well-known that Buckingham Palace is in London, and it is common knowledge that London is a city in England.

To recognize locations we use the CLR gazetteer[6], which is a large list of locations, including cities, counties, harbors, countries, etc., containing 162,853 entries in total.

Abbreviation. If the word is an abbreviation, the value of this feature is set to 1, and 0 otherwise. If a question asks for the definition of an abbreviation, such as question (29), the abbreviated term obviously has to be included in the query.

(29) What does HTML stand for?
(30) When is Fashion week in NYC?
(31) What TV series did Pierce Brosnan play in?

This is to a lesser extent the case for questions that do not ask for the full form of an abbreviation. In question (30), the word *NYC*, and in question (31) the word *TV* are abbreviations. Documents containing an answer do not necessarily have to contain the abbreviated word—in contrast to answer documents for question (29)—but they might as well contain the full form, i.e., *New York City*, and *television*, respectively.

Abbreviations are recognized by applying simple pattern matching. If a word consists of a series of capitalized letters, where each might be followed by a period, or a series of letters, where each is followed by a period, or occurs in a list of known abbreviations, the word is classified as an abbreviation. Maintaining a list of common abbreviations is necessary to recognize words such as *mph* as an abbreviation.

Upper Case. Words starting with a capital letter are normally part of a proper name, even when the word itself is not a noun.

(32) Who was Woodrow Wilson's First Lady?
(33) What group sang the song "Happy Together"?

In question (32), the adjective *First*, is part of the proper name *First Lady*, and in question (33), the adjective *Happy* and the adverb *Together* are part of the proper name *Happy Together*. Proper names are particularly important query terms and for recognizing them as such, it is not sufficient to rely on part-of-speech tags. Whether the fact that a part-of-speech tagger (TreeTagger in our case) tags the word *First* as adjective and not as proper

[6] Available from `ftp://crl.nmsu.edu/CLR/lexica/gazetteer`. Last accessed on February 4, 2008.

name has to be considered a mistake or not is difficult to say. From a syntactic point of view, *First* is clearly an adjective, and it is by convention that it is used a a proper noun in this specific context. Using the upper case feature in addition to part-of-speech tags allows us to recognize these cases.

Modified Noun. The information content of modified nouns tends to be higher than the content of single nouns, because the modifier imposes additional restrictions. In question (34), the noun *performer* is modified by the noun *child*, and in question (35) the noun range is modified by the nouns *blood* and *sugar*.

(34) Who holds the record as the highest paid child performer?
(35) What is the normal blood sugar range for people?

A noun is marked as modified by inspecting the part-of-speech tagged question. If a noun is preceded by an adjective, noun, or possessive, the `modified noun` feature is set to `yes`, and otherwise it is set to `no`. If the word is not tagged as a noun, the feature is set to `na` (not applicable). (36.b) is the part-of-speech tagged output of TreeTagger when applied to question (36.a). Here, *blood*, *sugar*, and *range* are all preceded by either an adjective or a noun. Hence, the `modified noun` feature is set to `yes` for the three words. *people* is preceded by a preposition, and therefore the `modified noun` feature is set to `no`. For all other words in the question, the feature is set to `na`.

(36) a. What is the normal blood sugar range for people?
b.

What	is	the	normal	blood	sugar	range	for	people	?
WRB	VBZ	DT	JJ	NN	NN	NN	IN	NN	SENT

Note that our definition of modification is rather simple and does not take any internal phrase structure into account. We pursue a simple linear approach to modification, which seems sufficient as we are not interested in the exact phrase that modifies a noun, but simply have to decide whether a noun is modified or not. This also allows us to circumvent the problem of phrase structure ambiguity.

Person Name. Words that are part of a person name are a special instance of words that are part of a proper name. Person names deserve special attention, because they can be further subdivided into first, middle, and last names.

(37) What is Francis Scott Key best known for?
(38) When did George W. Bush get elected as the governor of Texas?

In question (37), *Francis* is the first name, *Scott*, the middle name, and *Key*, the last name. Often, the middle name is abbreviated by using the first letter only, as in question (38), where the *W.* stands for *Walker*. The distinction between the different parts is important, because in many documents, the full name of a person is only used the first time the name occurs, and then later on that person is referred to by using the last name only. Hence, last names are more important for finding an answer.

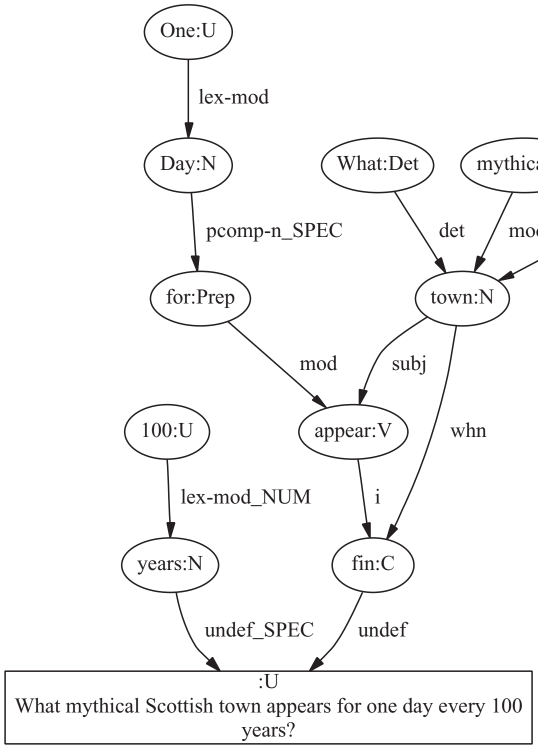

Fig. 2. MINIPAR dependency graphs for example question.

To identify person names, we use part of the U.S. Census resource,[7] which contains a list of first and last names. The list of first names contains 4,275 female and 1,219 male first names, and 101,865 last names.

Honorific. Honorific expressions include words such as *Mr.*, *Mrs.*, *Dr.*, etc. These terms do not bear much information and are therefore not essential for formulating a query.

(39) Where did Dr. King give his speech in Washington?

In question (39), insisting on the presence of the honorific word *Dr.* in a potential answer document is too restrictive, as many documents refer to Martin Luther King without using an honorific expression.

Number of incoming edges. The number of incoming edges refers to the dependency parse graph of the question which is generated by MINIPAR. If a word has a larger number of incoming edges, several words in the question are in a modifier or argument relationship with this word, and therefore it is more likely to play a central role in the question. Figure 2 shows the dependency graph for question (40).

(40) What mythical Scottish town appears for one day every 100 years?

[7] Available from `http://www.census.gov/genealogy/names/`.

The verb *appear* has two incoming edges, and the noun *town* has three. Nodes in the graph which are not associated with a word in the question, such as the *fin* node, are not considered.

Hypernym. Sometimes questions contain words that explicitly give the type of other words in the question. For instance, in question (41), *croquet* is classified by the word *game*.

(41) In what country did the game of croquet originate?
(42) What is the name of the volcano that destroyed the ancient city of Pompeii?

Similarly, in question (42), the word *city* makes explicit that *Pompeii* is a city. The word *game* is a hypernym of *croquet*, and the word *city* is a hypernym of *Pompeii*. Often, this kind of information is common knowledge and not explicitly mentioned in documents containing an answer. Hence, including these words in the queries might harm retrieval effectiveness. This is similar to the situation of words that appear in the question focus, as discussed above.

We use WordNet to find words or phrases that are hypernyms of other words or phrases in the question. Although WordNet does not contain entries for all question words—in particular most named entities are missing—the overall coverage is reasonable and stable across the test sets, covering on average 62.1 per cent (TREC-9), 67.7 per cent (TREC-10), and 61.6 per cent (TREC-11) of all nonstop words in the questions.

Relative idf score. Another indicator of the importance of a term is its frequency in the document collection. It is common to measure importance as the inverted document frequency (idf). Here, we are more interested in the relative importance of a term with respect to the other terms in the question. The relative idf score of term t in question q is defined as

$$\mathrm{ridf}(t, q) = \frac{\log_2(N/df_t)}{\sum_{t' \in q} \log_2(N/df'_t)}$$

where N is the number of documents in the collection, and df_t is the number of documents in which term t occurs.

At this point, after having discussed the individual features in some detail, it might be helpful to consider some example questions. Table 6 provides the complete feature instantiations for a number of questions.[8]

When comparing the feature representations of the words, a number of things can be noticed. For instance, three of the questions have words in the question focus: *country* in question 1, *year* in question 3, and *first* and *satellite* in question 4. Although all four terms occur in the focus of the respective question, there is a clear difference with respect to their generality. The word *country* has 300 leaves as hyponyms, the word year has 21 leaves as hyponyms, *first* has 7 and *satellite* has 20 leaves as hyponyms. Therefore, *country* is less likely to be helpful for retrieving answer documents than, for instance, *satellite*, as motivated above. On the other hand, *year* has only 7 leaves as hyponyms,

[8] In Table 6, some of the question classes had to be shortened to make the table fit on the page. The class `location` was shortened to `locat.`, `thing-ident` to `th-id`, and `date-of-death` to `dod`.

Table 6. *Example questions and their feature instantiations*

word	part-of-speech tag	question focus	superlative	question class	multiple occurrences	quoted	number leaves	term ratio	classifying word	location	abbreviation	upper case	modified noun	person name	honorific	no. incoming edges	hypernym	rel. idf
	In what country did the game of croquet originate?																	(question 1)
countri	NN	1	0	locat.	0	0	300	0.25	1	0	0	0	no	no	0	1	0	0.07
game	NN	0	0	locat.	0	0	195	0.25	0	0	0	0	no	no	0	2	1	0.13
croquet	NN	0	0	locat.	0	0	0	0.25	0	0	0	0	no	no	0	0	0	0.58
origin	VB	0	0	locat.	0	0	0	0.25	0	0	0	0	na	na	0	1	0	0.20
	When did president Herbert Hoover die?																	(question 2)
presid	NN	0	0	dod	0	0	2	0.25	0	0	0	0	no	no	0	1	0	0.08
herbert	NNP	0	0	dod	0	0	0	0.25	0	0	0	1	no	first	0	0	0	0.34
hoover	NNP	0	0	dod	0	0	0	0.25	0	0	0	1	no	last	0	1	0	0.42
die	VB	0	0	dod	0	0	4	0.25	1	0	0	0	na	na	0	1	0	0.16
	What year was the movie ‘Ole Yeller made?																	(question 3)
year	NN	1	0	date	0	0	21	0.20	1	0	0	0	no	no	0	1	0	0.03
movi	NN	0	0	date	0	0	15	0.20	0	0	0	0	no	no	0	0	0	0.16
ol	NNP	0	0	date	0	1	0	0.20	0	0	0	1	no	no	0	0	0	0.29
yeller	NNP	0	0	date	0	1	0	0.20	0	0	0	1	no	no	0	4	0	0.48
make	V	0	0	date	0	0	2	0.20	0	0	0	0	na	na	0	0	0	0.04
	What was the first satellite in space?																	(question 4)
first	JJS	1	1	th-id	0	0	7	0.33	0	0	0	0	na	na	0	0	0	0.12
satellit	NN	1	1	th-id	0	0	20	0.33	0	0	0	0	yes	no	0	4	0	0.49
space	NN	0	1	th-id	0	0	103	0.33	0	0	0	0	no	no	0	0	0	0.40

indicating that it is a rather specific term, which is certainly not the case. This is due to the fact that WordNet does not list all possible years as hyponyms of the concept *year*. One way to distinguish between truly specific focus words such as *satellite* and words such as *year* is the feature `classifying word`, which is set to 1 for *year*, and 0 for *satellite*.

In question 2, *Herbert Hoover* was recognized as a name and for *Herbert* the `person name` feature was set to `first` and for *Hoover* it was set to last. If a noun was not recognized as part of a name the feature is set to `no`, and for part-of-speech tags other than nouns, the feature is set to `na` (not applicable).

6 Learning term weights

The previous section discussed the individual features that are used to represent words in a question. In this section we discuss how we use these features to predict the importance of a term for query formulation. Instead of using some heuristics for predicting the query term importance we apply machine learning techniques to assign a weight to each term in the question, where the actual query that is used for retrieval will include these weights. The input for the learning algorithm is the set of feature vectors as described in the previous section, and the classes are the terms' gains as described in Section 4.

For the purpose of learning term weights, the machine learning algorithm should learn to predict the degree of the query term's usefulness for query formulation. Decision trees, naive Bayes, and linear regression, all allow for interval classification and generate transparent classification rules. Naive Bayes classification is known to be well-performing for nominal classification, see (Domingos and Pazzani 1997), but performs badly for interval classification (Frank *et al.* 2000).

In general, decision trees partition the training data into a hierarchical structure. The hierarchical structure is based on the information gains of each attribute or feature, where attributes with higher information gains are closer to the root of the tree. Decision trees are commonly used for classification and they are general enough to be applied to numerous natural language processing tasks, such as part-of-speech tagging (Schmid 1994), parsing (Haruno, Shirai and Ooyama 1998), and word-sense disambiguation (Brown *et al.* 1991). The best-known algorithm for decision tree learning is Quinlan's C4.5 (Quinlan 1993), but C4.5 cannot deal with cases where classes are not labels, but real numbers. M5 (Quinlan 1992), on the other hand, which is an extension of C4.5, does allow for this type of continuous classification, also referred to as regression.

The M5 algorithm builds model trees combining conventional decision tree learning with the possibility of linear regression models at the leaves of the tree. The resulting representation is transparent because the decision structure is clear and the regression models are normally easily interpretable. The idea of model trees is largely based on the concept of regression trees. The advantage of M5 is that model trees are generally much smaller than regression trees and have proved to be more accurate in a number of tasks; see (Quinlan 1992). The learning algorithm used here, is M5′ (Wang and Witten 1997), which is a reconstruction of Quinlan's M5 algorithm. M5′ is part of the Weka ML package (Witten and Frank 2005). Our specific choice of M5′ is mainly motivated by the comparison of Frank *et al.* (2000) who have shown experimentally that M5′ outperforms Naive Bayes, linear regression, and locally weighted regression, on a large number of regression tasks.

Figure 3 shows parts of a model tree that has been generated by M5′. The highest branching of the tree in Figure 3, checks whether the word at hand is a first name, last name or other noun, i.e., `personname = first,no,last <= 0.5`. If not, further analysis descends down the other branch. At each leaf node is a linear regression model, e.g., `LM1` in Figure 3. `LM1` confirms some of the intuitions for query term selection as discussed above. If the word occurs in the question focus, this has a negative impact on the term weight (`-0.00816focus`). Also, if the question does not contain a superlative adjective, the query term weight is lowered (`-0.00135superlative=0`). Words that are not used to

```
persname=first,no,last <= 0.5 :
|   relidf <= 0.332 :
|   |   noleaves <= 2.5 :
|   |   |   tag=JJ,LS,CD,JJS,NN,PRP,
|   |   |       F,NNP,NN,NNP <= 0.5 : LM1
...
LM1: class = -1.12 - 0.00816focus
             - 0.00135superlative=0
             ...
             + 0.0084usedtoclassify=0
             + 1.03abbreviation=0
             + 0.03uppercase=1
             + 0.0181persname=na,first,no,last
             + 0.00178personname=no,last
             + 0.00131personname=last
             + 0.00218hypernym=0
             + 0.0112relidf
             ...
```

Fig. 3. Excerpt of a learned model tree.

classify the question receive a higher term weight (`+0.0084usedtoclassify=0`), and the same applies to question words that are not abbreviations (`+1.03abbreviation=0`).

7 Results

In the previous sections we discussed which features are extracted to represent terms and how these features are used to compute a term's importance in the context of the question in which the term occurs. In this section we discuss the experimental results when integrating the term importance information into a standard retrieval approach.

7.1 Experimental results

To integrate the learned term weights, as described above, the computation of the retrieval status value (*RSV*) has to be adapted appropriately. We use the learned query term weights in combination with the original retrieval status value that resulted from computing the similarity between a query q and a document d according to the Lnu.ltc weighting scheme. For the details of the Lnu.ltc scheme, the reader is referred to (Buckley, Singhal and Mitra 1995; Singhal *et al.* 1996).

$$RSV(q,d) = \sum_{i \in q \cap d} \frac{\dfrac{1+\log(\mathrm{freq}_{i,d})}{1+\log(\mathrm{avg}_{j\in d}\mathrm{freq}_{j,d})} \cdot \dfrac{\mathrm{freq}_{i,q}}{\max\limits_{j\in q}\mathrm{freq}_{j,q}} \cdot \log\left(\dfrac{N}{n_i}\right)}{((1-sl)\cdot pv + sl \cdot \mathrm{uw}_d) \cdot \sqrt{\sum\limits_{i\in q}\left(\dfrac{\mathrm{freq}_{i,q}}{\max\limits_{j\in q}\mathrm{freq}_{j,q}} \cdot \log\left(\dfrac{N}{n_i}\right)\right)^2}}$$

For our experiments we carried out a simple grid search to determine the best value for the slope resulting in a value of 0.2, which is in line with commonly reported values for the slope. *pv* (pivot) was set to the average number of unique words occurring in the collection.

Table 7. *Comparison of the f@n scores of learned-weights retrieval (LWR) runs to the baseline runs. As these are error rates, lower values are better*

	TREC-9			TREC-10			TREC-11		
f@*n*	Base	LWR	Red.	Base	LWR	Red.	Base	LWR	Red.
5	0.300	0.273	(−9.0%)$^\triangledown$	0.351	0.346	(−1.4%)	0.477	0.453	(−5.0%)$^\triangledown$
10	0.215	0.194	(−9.8%)$^\triangledown$	0.266	0.270	(+1.5%)	0.374	0.363	(−2.9%)
20	0.155	0.137	(−11.6%)	0.199	0.196	(−1.5%)	0.295	0.268	(−9.2%)$^\triangledown$
50	0.086	0.092	(+7.3%)	0.125	0.141	(+12.8%)	0.205	0.185	(−9.6%)

uw_d refers to the number of unique words in document d. The formula for computing the *RSV* above is a standard approach in document retrieval. The slope and pivot parameters are used to counterbalance the tendency of most cosine-based retrieval measures to prefer shorter documents. The *RSV* measure above is used as our baseline.

In order to integrate the term importance weights, as defined in Section 5 and 6, we changed the Lnu.ltc similarity measure to:

$$RSV_L(q,d) = \sum_{i \in q \cap d} \frac{w(rep(t,q)) \cdot \dfrac{1+\log(\mathrm{freq}_{i,d})}{1+\log(\mathrm{avg}_{j \in d}\mathrm{freq}_{j,d})} \cdot \dfrac{\mathrm{freq}_{i,q}}{\max_{j \in q} \mathrm{freq}_{j,q}} \cdot \log\left(\dfrac{N}{n_i}\right)}{((1-sl)\cdot pv + sl \cdot \mathrm{uw}_d) \cdot \sqrt{\sum_{i \in q}\left(w(rep(t,q)) \cdot \dfrac{\mathrm{freq}_{i,q}}{\max_{j \in q}\mathrm{freq}_{j,q}} \cdot \log\left(\dfrac{N}{n_i}\right)\right)^2}}$$

Here, $rep(t,q)$ is the feature representation of term t in query q, as described in Section 5, and $w(rep(t,q))$ is the learned weight, which results from applying the M5′ model tree to t's feature representation, as described in Section 6.

As the term weights are estimated by using parts of the TREC collection, we used cross-validation to clearly distinguish between training data and test data. For the evaluation three different model trees were generated, one for each of the TREC data sets. The model tree for assigning weights to terms in the TREC-9 test set was trained on feature representations of words from TREC-10 and TREC-11 (2,854 instances), the model tree for the TREC-10 test set was trained on feature representations from TREC-9 and TREC-11 (3,167 instances), and the model tree for the TREC-11 test set used feature representations from TREC-9 and TREC-10 (2,769 instances).

First, we consider the performance with respect to the failure-at-*n* (f@*n*) measure. Table 7 shows the results of using learned query terms weights in contrast to the Lnu.ltc baseline. The improvements vary between the different test sets. For nine of the twelve runs LWR yields a reduction in f@*n* compared to the baseline. In particular for lower and medium cut-offs ($n \leq 20$) the improvements are consistent, with the exception of TREC-10 for f@10. At higher cut-off values ($n = 50$) only the run for TREC-11 led to an improvement.

It is notable that the relative reductions in error rate for the TREC-10 test set are substantially lower. One reason for this is the large number of definition questions it TREC-10. While the TREC-9 data sets contains 9 per cent definition questions and TREC-11 contain

10 per cent, TREC-10 contains 25 per cent of definition questions. As definition questions contain more technical terms, one could assume that this could pose a problem for some of the features, such as the hypernym feature and the number of leaves feature which both rely on WordNet, as they often contain technical terms that are not found in WordNet. When looking at the WordNet coverage for definition questions, it turns out that WordNet coverage is actually higher (75.0 per cent on average) for definition questions than for the entire TREC-10 data set (67.7 per cent on average).

One other significant difference between the TREC-10 on the one hand and the TREC-9 and TREC-11 sets on the other hand, is the average length of the questions. Even after discarding definition questions, which are typically rather short, TREC-9 questions contain on average 3.2 content words, while this is 3.6 for TREC-10 and 3.8 for TREC-11.

One would expect term weights to have more impact for longer questions than for shorter questions. To quantify this we computed the correlation between the improvements and the question length using Kendall's tau, which resulted in a modest positive correlation of 0.41.

While our term weighting approach leads to improvements, one might be able to get similar improvements with other retrieval schemes that have been applied in the context of question answering. Passage-based retrieval is commonly used within question answering as it emphasizes the tendency of question words to occur within some proximity, see, e.g., (Roberts and Gaizauskas 2004). To this end we compare our results to two passage retrieval approaches. The first approach splits all documents in the collection into passages of fixed length where passages within a document have an overlap ratio of 50 per cent. The second passage-based retrieval approach models Approach 3 of Roberts and Gaizauskas (2004), where in the first stage document retrieval is carried out and in a second stage the top 5,000 documents are split into passages and passage-based retrieval is carried out on the subcollection of top documents. For both approaches the passage length is set to 150 words. While we have experimented with several passage lengths, setting the length to 150 words yielded the most stable results.

Table 8 shows the results for both sets of passage retrieval runs. Our approach clearly outperforms both types of passage-based retrieval. Indeed, the improvements are substantially larger than compared to the baseline. Although it may seem surprising that both passage-based approaches perform worse than the baseline, this is consistent with (Roberts and Gaizauskas 2004).

The approaches discussed so far only used words from the original question to formulate queries. It has been shown for *ad hoc* retrieval that it can be helpful to extend the query with words occurring in documents that can be assumed to be relevant. This approach is known as Blind-Relevance Feedback, where terms from the top-n ranked (n typically being 5 or 10) documents from an initial retrieval pass are added to the query to perform a second retrieval pass, see (Mitra, Singhal and Buckley 1998) for more details. Table 9 shows the results comparing the learned-weights approach to blind relevance feedback in the context of question answering. LWR clearly outperforms blind relevance feedback. This is also in line with earlier findings, see, e.g. (Monz 2003). The main reason for this is that while the terms that are used to expand the query are topical they are not specific enough to the exact question.

Finally, we compare the performance between LWR and the baseline with respect to mean average precision and the results are displayed in Table 10. Compared to the state-of-the-art Lnu.ltc baseline, the improvements are clearly statistically significant. This shows

Table 8. *Comparison of the f@n scores for learned-weight retrieval (LWR) to passage-based retrieval (Pass) and document-passage retrieval (Pass-3)*

	TREC-9			TREC-10			TREC-11		
f@n	Pass	LWR	Red.	Pass	LWR	Red.	Pass	LWR	Red.
5	0.325	0.273	(−16.0%)▼	0.390	0.346	(−11.3%)▼	0.527	0.453	(−14.0%)▼
10	0.235	0.194	(−17.4%)▼	0.327	0.270	(−17.4%)▼	0.410	0.363	(−11.5%)▼
20	0.170	0.137	(−19.5%)▼	0.219	0.196	(−10.5%)▼	0.304	0.268	(−11.8%)▼
50	0.088	0.092	(+4.5%)	0.135	0.141	(+4.4%)	0.219	0.185	(−15.5%)▼
f@n	Pass-3	LWR	Red.	Pass-3	LWR	Red.	Pass-3	LWR	Red.
5	0.329	0.273	(−17.0%)▼	0.393	0.346	(−11.9%)▽	0.532	0.453	(−14.8%)▼
10	0.227	0.194	(−14.5%)▼	0.285	0.270	(−5.3%)	0.404	0.363	(−10.1%)▽
20	0.166	0.137	(−17.5%)▼	0.205	0.196	(−4.4%)	0.312	0.268	(−14.1%)▼
50	0.087	0.092	(+5.7%)	0.127	0.125	(−1.6%)	0.212	0.185	(−12.7%)▼

Table 9. *Comparison of the f@n scores for learned weight retrieval (LWR) to blind relevance feedback (BRF)*

	TREC-9			TREC-10			TREC-11		
f@n	BRF	LWR	Red.	BRF	LWR	Red.	BRF	LWR	Red.
5	0.388	0.273	(−29.6%)▼	0.472	0.346	(−26.7%)▼	0.600	0.453	(−24.5%)▼
10	0.288	0.194	(−32.6%)▼	0.398	0.270	(−32.2%)▼	0.508	0.363	(−28.5%)▼
20	0.217	0.137	(−36.9%)▼	0.294	0.196	(−33.3%)▼	0.418	0.268	(−35.9%)▼
50	0.140	0.092	(−34.3%)▼	0.181	0.125	(−30.9%)▼	0.293	0.185	(−36.9%)▼

Table 10. *Comparison of mean average precisions (MAP) of learned-weights retrieval (LWR) to the baseline*

TREC-9			TREC-10			TREC-11		
Base	LWR	Impr	Base	LWR	Impr	Base	LWR	Impr
0.280	0.328	(+17.1%)▲	0.279	0.296	(+6.1%)▲	0.214	0.242	(+13.1%)▲

that using query term weights tends to rank relevant documents higher than the baseline and that it tends to find more documents containing an answer, which is not measured by f@n.

7.2 Importance of individual features

Having evaluated the effectiveness of using the model tree to predict term weights for retrieval purposes, we discuss the accuracy of the learned model tree itself. Table 11

Table 11. *Accuracy of the model tree learning algorithm*

Evaluation measure	Score
Correlation coefficient	0.5018
Mean absolute error	0.3783
Relative absolute error	82.1%

provides some of the figures that are generated by the WEKA machine learning package for the model tree learning algorithm. Evaluation has been done on the training data using ten-fold cross validation. In n-fold cross validation, the training data is arbitrarily split into n partitions. The model tree learning algorithm is applied n times to $n - 1$ partitions, where each time a different partition is held out for evaluating. Overall evaluation scores are obtained by averaging over the n individual evaluation scores. The correlation coefficient indicates the degree to which the predicted value and the original values, as provided by the test data, correlate. A value of 1 (-1) indicates perfect (inverse) correlation, and a value of 0 indicates no correlation at all. Here, a correlation coefficient of 0.5 means that the predicted and original values are weakly correlated. The mean absolute error is the mean absolute difference between the predicted value (term weight) and the original value. The relative absolute error is the mean relative difference between the predicted value and the original value expressed in percents. A relative absolute error of 100 per cent corresponds to the error that would have been obtained by always taking the mean value of all training instances for prediction. In our experiments, the relative absolute error is 82.1 per cent, which is rather high, but still substantially better than choosing the mean training value for prediction.

In addition to evaluating the accuracy of the whole model tree, it is also interesting to estimate the importance of a single feature or attribute for learning the query term weight. This can be done by computing the attribute's information gain, cf. (Breimann *et al.* 1984). Information gain measures the reduction in uncertainty, where the degree of uncertainty is measured as the entropy. The *information gain* of attribute A with respect to class C is defined as:

$$(43)\quad \begin{aligned} \mathit{InfoGain}(A, C) &= H(C) - H(C|A) \\ &= -\sum_{c \in C} p(c)\log_2(p(c)) \\ &\quad -\left(-\sum_{c \in C}\sum_{a \in A} p(c, a)\log_2(p(c|a))\right) \end{aligned}$$

Note that the information gain computes the importance of an attribute independently of other attributes.

The problem with using information gain in the current context is that a number of attributes and the learned class, the query term weight, have to be discretized. Discretization is a nontrivial process in itself, and the way discretization is carried out has an impact on the estimation of the information gain. Hence, we used *regression relief*, which is a measure for estimating the importance of an attribute for learning the query term weight, and which can easily deal with continuous attributes and classes.

Table 12. *RReliefF estimates of features*

Rank	Feature	RReliefF value
1	`abbreviation`	0.006088
2	`qtype`	0.004000
3	`noleaves`	0.003909
4	`relidf`	0.003655
5	`tag`	0.003272
6	`focus`	0.003058
7	`wordratio`	0.002637
8	`hypernym`	0.001847
9	`superlative`	0.001492
10	`twice`	0.000966
11	`quotes`	0.000502
12	`honorific`	0.000229
13	`usedtoclassify`	0.000163
14	`uppercase`	0.000109
15	`noincomingedges`	0.000006
16	`modifiednoun`	−0.000030
17	`location`	−0.000454
18	`personname`	−0.001028

(Robnik-Šikonja and Kononenko 1997; Robnik-Šikonja and Kononenko 2003) introduce the regression relief algorithm (RReliefF) to estimate the weight of an attribute. The key idea of the RReliefF algorithm is to estimate the quality of an attribute according to how well it discriminates between instances (feature vectors of query terms) that are near to each other. For this purpose, an instance R is randomly selected. Then, the k nearest instances, with respect to the class value, are selected, and the difference between the value of an attribute A of R and the value of the same attribute for one of the k instances is compared with respect to the difference of their class values. This process is repeated for a number of instances, potentially all, which finally leads to a weight for each attribute. The weight can range between -1 and 1. The full details of the RReliefF algorithm can be found in (Robnik-Šikonja and Kononenko 1997; Robnik-Šikonja and Kononenko 2003).

Table 12 shows the RReliefF estimates for the 18 attributes or features that were used to learn query term weights. The classes (the term weights themselves) were determined by applying the model tree that was generated from the TREC-9, TREC-10 and TREC-11 datasets. For computing the RReliefF estimates, we used the WEKA system, which provides and implementation of the RReliefF algorithm.

The ranking of the features reveals a number of interesting aspects. First, the `personname` feature is ranked lowest according to the RReliefF estimation, but it is the highest branching feature in the model tree in Figure 3. One explanation for this discrepancy is the fact that `personname` is apparently too general a feature to predict query term weights by itself. The `abbreviation` feature receives the highest RReliefF estimate, although it does not appear in the model tree in Figure 3. The high rank of the `abbreviation` feature is probably due to the fact that it occurs in all of the linear models `LM1`–`LM17` with a relatively high regression coefficient, at the leaves of the model tree. The same holds for the `qtype` feature. The `relidf` and `noleaves` features, which are also ranked high by RReliefF, also occur high in the model tree, but are apparently more helpful for predicting the term weights than the `personname` feature, because they also occur in all linear models with coefficients that are higher than the coefficients of the `personname` feature.

Unfortunately, it is hard to distillate an explanation for each of the features' RReliefF estimate from the model tree. Nevertheless the RReliefF estimate does provide some insights into the importance of a feature independent of other features.

8 Conclusions

In this paper we investigated the importance of term weighting for information retrieval in the context of question answering. We have seen that optimal query weighting can have a substantial impact on retrieval quality, reducing the failure rates of the retrieval component significantly.

In order to learn query term weights, we considered all possible ways of selecting terms from the original question for query formulation and used the performance results of each possible formulation in order to determine individual query term weights.

To overcome data sparseness issues, query terms are represented as sets of features on which the M5$'$ model tree learning algorithm is trained. The resulting model trees confirm some of the heuristics and intuitions for keyword selection than can be found in other approaches; see, e.g., (Paşca 2001).

Integrating the learned query term weights into a standard retrieval approach reduces the failure rate substantially in most cases. In addition, our proposed term weight learning approach yields significantly better results than passage-based retrieval approaches commonly used for document retrieval in the context of question answering. Our approach also outperforms the baseline with respect to mean average precision. Hence question answering systems that are more sensitive to the rank position of a retrieved document can benefit from using our approach.

In some cases the issue of whether a term is helpful for retrieving answer documents simply depends on idiosyncrasies of the documents that contain an answer, but our training sets were fairly large and varied in order to generalize properly.

References

Agichtein, E., Lawrence, S., and Gravano, L. 2004. Learning to find answers to questions on the web. *ACM Transactions on Internet Technology* **4**(2): 129–162.

Bendersky, M., and Croft, B. 2008. Discovering key concepts in verbose queries. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Singapore, pp. 491–498.

Bilotti, M., Ogilvie, P., Callan, J., and Nyberg, E. 2007. Structured retrieval for question answering. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Amsterdam, The Netherlands, pp. 351–358.

Breimann, L., Friedman, J., Ohlsen, R., and Stone, C. 1984. *Classification and Regression Trees*. Wadsworth and Brooks.

Brill, E., Dumais, S., and Banko, M. 2002. An analysis of the AskMSR question-answering system. In *Proceedings of Empirical Methods in Natural Language Processing (EMNLP 2002)*, Morristown, N.J., USA, pp. 257–264.

Brown, P. F., Della Pietra, S., Della Pietra, V., and Mercer, R. L. 1991. Word sense disambiguation using statistical methods. In *Proceedings of the 29th Annual Meeting of the Association for Computational Linguistics (ACL)*, Morristown, N.J., USA, pp. 264–270.

Buckley, C., Singhal, A., and Mitra, M. 1995. New retrieval approaches using SMART: TREC 4. In *Proceedings of the Fourth Text REtrieval Conference (TREC-4)*, Gaithersburg, M.D., USA, pp. 25–48.

Chen, H., Shankaranarayanan, G., She, L., and Iyer, A. 1998. A machine learning approach to inductive query by examples: an experiment using relevance feedback, ID3, genetic algorithms, and simulated annealing. *Journal of the American Society for Information Science* **49**(8): 693–705.

Cooper, W., Chen, A., and Gey, F. 1993. Full text retrieval based on probabilistic equations with coefficients fitted by logistic regression. In *Proceedings of the 2nd Text REtrieval Conference*, Gaithersburg, M.D., USA, pp. 57–66.

Domingos P., and Pazzani, M. (1997) On the optimality of the simple bayesian classifier under zero-one loss. *Machine Learning* **29**(2–3): 103–130.

Efron, B. 1979. Bootstrap methods: another look at the jackknife. *Annals of Statistics* **7**(1): 1–26.

Frank, E., Trigg, L., Holmes, G., and Witten, I. H. 2000. Naive bayes for regression. *Machine Learning* **41**(1): 5–25.

Harabagiu, S., Paşca, M., and Maiorano, S. 2000. Experiments with open-domain textual question answering. In *Proceedings of the 18th International Conference on Computational Linguistics (COLING 2000)*, Saarbrücken, Germany, pp. 292–298.

Harabagiu, S., Moldovan, D., Paşca, M., Mihalcea, R., Surdeanu, M., Bunescu, R., Girju, R., Vasile, R., and Morarescu, P. 2001. The role of lexico-semantic feedback in open-domain textual question-answering. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics (ACL-2001)*, Morristown, N.J., USA, pp. 274–281.

Haruno, M., Shirai, S., and Ooyama, Y. 1998. Using decision trees to construct a practical parser. In *Proceedings of the joint 17th International Conference on Computational Linguistics and 36th Annual Meeting of the Association for Computational Linguistics (COLING-ACL)*, Montreal, Canada, pp. 1136–1142.

Hermjakob, U. 2001. Parsing and question classification for question answering. In *Proceedings of the ACL 2001 Workshop on Question Answering*, Morristown, N.J., USA, pp. 17–22.

Ittycheriah, A., Franz, M., Zhu, W., Ratnaparkhi, A., and Mammone, R. 2001. Question answering using maximum entropy components. In *Proceedings of the 2nd Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL-2001)*, Morristown, N.J., USA, pp. 33–39.

Keenan, S., Smeaton, A. F., and Keogh, G. 2001. The effect of pool depth on system evaluation in TREC. *Journal of the American Society for Information Science and Technology* **52**(7): 570–574.

Lee, C., Chen, R., Kao, S., and Cheng, P. 2009. A term dependency-based approach for query terms ranking. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management (CIKM)*, Hong Kong, China, pp. 1267–1276.

Li, X., and Roth, D. 2006. Learning question classifiers: the role of semantic information. *Natural Language Engineering* **12**(3): 229–249.

Lin, D. 1998. Dependency-based evaluation of Minipar. In *Proceedings of the Workshop on the Evaluation of Parsing Systems*, Granada, Spain, pp. 317–330.

Lin, D., and Pantel, P. 2001. Discovery of inference rules for question-answering. *Natural Language Engineering* **7**(4): 343–360.

Lita, L. V., and Carbonell, J. 2004. Unsupervised question answering data acquisition from local corpora. In *Proceedings of the Thirteenth Conference on Information and Knowledge Management (CIKM 2004)*, Washington, D.C., USA, pp. 607–614.

Mayfield, J., and McNamee, P. 2005. JHU/APL at TREC 2005: QA retrieval and robust tracks. In E. M. Voorhees, and L. P. Buckland (eds.), *Proceedings of the Fourteenth Text*

REtrieval Conference (TREC 2005). NIST Special Publication: SP 500-266, Gaithersburg, M.D., USA.

Mitra, M., Singhal, A., and Buckley, C. 1998. Improving automatic query expansion. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Melbourne, Australia, pp. 206–214.

Monz, C. 2003. Document retrieval in the context of question answering. In F. Sebastiani (ed.), *Proceedings of the 25th European Conference on Information Retrieval Research (ECIR-03)*, LNCS 2633, pp. 571–579. Springer.

Paşca, M. 2001. *High-Performance Open-Domain Question Answering from Large Text Collections*. PhD thesis, Southern Methodist University.

Quinlan, J. R. 1992. Learning with continuous classes. In *Proceedings of the 5th Australian Joint Conference on Artificial Intelligence*, Singapore, pp. 343–348.

Quinlan, J. R. 1993. *C4.5: Programs for Machine Learning*. Morgan Kaufmann.

Radev, D., Qi, H., Zheng, Z., Blair-Goldensohn, S., Zhang, Z., Fan, W., and Prager, J. 2001. Mining the web for answers to natural language questions. In *Proceedings of the tenth International Conference on Information and Knowledge Management (CIKM)*, Atlanta, Georgia, USA, pp. 143–150.

Ravichandran, D., and Hovy, E. 2002. Learning surface text patterns for a question answering system. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, Morristown, N.J., USA, pp. 41–47.

Robertson, S., Walker, S., and Beaulieu, M. 1998. Okapi at TREC-7: automatic ad hoc, filtering, VLC and interactive. In *The 7th Text REtrieval Conference (TREC 7)*, Gaithersburg, M.D., USA: NIST Special Publication 500-242, pp. 253–264.

Robnik-Šikonja, M., and Kononenko, I. 1997. An adaptation of relief for attribute estimation on regression. In D. Fisher (ed.), *Proceedings of 14th International Conference on Machine Learning ICML'97*, pp. 296–304, San Francisco, C.A.

Robnik-Šikonja, M., and Kononenko, I. 2003. Theoretical and empirical analysis of ReliefF and RReliefF. *Machine Learning* **54**(1–2): 23–69.

Roberts, I., and Gaizauskas, R. 2004. Evaluating passage retrieval approaches for question answering. In *Proceedings of the 26th European Conference on Information Retrieval (ECIR'04)*, Sunderland, UK, pp. 72–84.

Sampson, G. 1995. *English for the Computer—The SUSANNE Corpus and Analytic Scheme*. Clarendon Press.

Santorini, B. 1990. *Part-of-speech tagging guidelines for the Penn Treebank*, 3rd rev., 2nd ed.. Department of Computer Science, University of Pennsylvania.

Schmid, H. 1994. Probabilistic part-of-speech tagging using decision trees. In *Proceedings of International Conference on New Methods in Language Processing*, Manchester, UK.

Singhal, A., Salton, G., Mitra, M., and Buckley, C. 1996. Document length normalization. *Information Processing & Management* **32**(5): 619–633.

Collins-Thompson, K., Callan, J., Terra, E., and Clarke, C. 2004. The effect of document retrieval quality on factoid question answering performance. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Sheffield, UK, pp. 574–574.

Voorhees, E. 1994. Query expansion using lexical-semantic relations. In *Proceedings of the 37th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Dublin, Ireland, pp. 61–69.

Wang, Y. and Witten, I. H. 1997. Induction of model trees for predicting continuous classes. In *Proceedings of the Poster Papers of the European Conference on Machine Learning (ECML)*, Prague, Czech Republic, pp. 128–137.

Wilbur, W. J. 1994. Nonparametric significance tests of retrieval performance comparisons. *Journal of Information Science* **20**(4): 270–284.

Williams, E. 1981. On the notions 'lexically related' and 'head of a word'. *Linguistic Inquiry* **12**: 245–274.

Witten, I. H., and Frank, E. 2005. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*, 2nd ed. Morgan Kaufmann.

Yousefi, J., and Kosseim, L. 2006. Automatic acquisition of semantic-based question reformulations for question answering. In *Proceedings of the Seventh International Conference on Intelligent Text Processing and Computational Linguistics (CICLing)*, Mexico City, Mexico, pp. 441–452.