A Practically Unsupervised Learning Method to Identify Single-Snippet Answers to Definition Questions on the Web

Ion Androutsopoulos and **Dimitrios Galanis**
Department of Informatics
Athens University of Economics and Business
Patission 76, GR-104 34, Athens, Greece

Abstract

We present a practically unsupervised learning method to produce single-snippet answers to definition questions in question answering systems that supplement Web search engines. The method exploits on-line encyclopedias and dictionaries to generate automatically an arbitrarily large number of positive and negative definition examples, which are then used to train an SVM to separate the two classes. We show experimentally that the proposed method is viable, that it outperforms the alternative of training the system on questions and news articles from TREC, and that it helps the search engine handle definition questions significantly better.

1 Introduction

Question answering (QA) systems for document collections typically aim to identify in the collections text snippets (e.g., 50 or 250 characters long) or exact answers (e.g., names, dates) that answer natural language questions submitted by their users. Although they are commonly evaluated on newspaper archives, as in the TREC QA track, QA systems can also supplement Web search engines, to help them return snippets, as opposed to Web pages, that provide more directly the information users require.

Most current QA systems first classify the input question into one of several categories (e.g., questions asking for locations, persons, etc.), producing expectations for types of named entities that must be present in the answer (locations, person names, etc.). Using the question's terms as a query, an information retrieval (IR) system identifies relevant documents. Snippets of these documents are then selected and ranked, using criteria such as whether or not they contain the expected types of named entities, the percentage of the question's terms they contain, etc. The system then outputs the most highly-ranked snippets, or named entities therein.

The approach highlighted above performs poorly with definition questions (e.g., "What is gasohol?", "Who was Duke Ellington?"), because definition questions do not generate expectations for particular types of named entities, and they typically contain only a single term. Definition questions are particularly common; in the QA track of TREC-2001, where the distribution of question types reflected that of real user logs, 27% of the questions were requests for definitions. Of course, answers to many definition questions can be found in on-line encyclopedias and dictionaries.[1] There are always, however, new or less widely used terms that are not included in such resources, and this is also true for many names of persons and products. Hence, techniques to discover definitions in ordinary Web pages and other document collections are valuable. Definitions of this kind are often 'hidden' in oblique contexts (e.g., "He said that *gasohol, a mixture of gasoline and ethanol*, has been great for his business.").

In recent work, Miliaraki and Androutsopoulos (2004), hereafter M&A, proposed a method we call

[1] See, for example, Wikipedia (http://www.wikipedia.org/). WordNet's glosses are another on-line source of definitions.

Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP), pages 323–330, Vancouver, October 2005.

DEFQA, which handles definition questions. The method assumes that a question preprocessor separates definition from other types of questions, and that in definition questions this module also identifies the term to be defined, called the *target term*.[2] The input to DEFQA is a (possibly multi-word) target term, along with the r most highly ranked documents that an IR system returned for that term. The output is a list of k 250-character snippets from the r documents, at least one of which must contain an acceptable short definition of the target term, much as in the QA track of TREC-2000 and TREC-2001.[3]

We note that since 2003, TREC requires definition questions to be answered by lists of *complementary* snippets, jointly providing a range of information nuggets about the target term (Voorhees, 2003). In contrast, here we focus on locating single-snippet definitions. We believe this task is still interesting and of practical use. For example, a list of single-snippet definitions accompanied by their source URLs is a good starting point for users of search engines wishing to obtain definitions. Single-snippet definitions can also be useful in information extraction, where the templates to be filled in often require short entity descriptions. We also note that the post-2003 TREC task has encountered evaluation problems, because it is difficult to agree on which nuggets should be included in the multi-snippet definitions (Hildebrandt et al., 2004). In contrast, our experimental results of Section 4 indicate strong inter-assessor agreement for single-snippet answers, suggesting that it is easier to agree upon what constitutes an acceptable single-snippet definition.

DEFQA relies on an SVM, which is trained to classify 250-character snippets that have the target term at their centre, hereafter called *windows*, as acceptable definitions or non-definitions.[4] To train the SVM, a collection of q training target terms is used; M&A used the target terms of definition questions from TREC-2000 and TREC-2001. The terms are submitted to an IR system, which returns the r most highly ranked documents per target term. The windows of the $q \cdot r$ resulting documents are tagged as acceptable definitions or non-definitions, and become the training instances of the SVM. At run time, when a definition question is submitted, the r top-ranked documents are obtained, their windows are collected, and for each window the SVM returns a score indicating how confident it is that the window is a definition. The k windows with the highest confidence scores are then reported to the user.

The SVM actually operates on vector representations of the windows, that comprise the verdicts or attributes of previous methods by Joho and Sanderson (2000) and Prager et al. (2002), as well as attributes corresponding to automatically acquired lexical patterns. On TREC-2000 and TREC-2001 data, M&A found that DEFQA clearly outperformed the original methods of Joho and Sanderson and Prager et al. Their best configuration answered correctly 73% of 160 definition questions in a cross-validation experiment with $k = 5$, $r = 50$, $q = 160$.

A limitation of DEFQA is that it cannot be trained easily on new document collections, because it requires the training windows to be tagged as definitions or non-definitions. In the experiments of M&A, there were 18,473 training windows. Tagging them was easy, because the windows were obtained from TREC questions and documents, and the TREC organizers provide Perl patterns that can be used to judge whether a snippet from TREC's documents is among the acceptable answers of a TREC question.[5] For non-TREC questions and document collections, however, where such patterns are unavailable, separating thousands of training windows into the two categories by hand is a laborious task.

In this paper, we consider the case where DEFQA is used as an add-on to a Web search engine. There are three training alternatives in this setting: (i) train DEFQA on TREC questions and documents; (ii) train DEFQA on a large collection of manually tagged training windows obtained from Web pages that the search engine returned for training target terms; or (iii) devise techniques to tag automatically the training windows of (ii). We have developed a technique along alternative (iii), which exploits on-

[2] Alternatively, the user can be asked to specify explicitly the question type and target term via a form-based interface.

[3] Definition questions were not considered in TREC-2002.

[4] See, for example, Scholkopf and Smola (2002) for information on SVMs. Following M&A, we use a linear SVM, as implemented by Weka's SMO class (http://www.cs.waikato.ac.nz/ml/weka/). The windows may be shorter than 250 characters, when the surrounding text is limited.

[5] The patterns' judgements are not always perfect, which introduces some noise in the training examples.

line encyclopedias and dictionaries. This allows us to generate and tag automatically an arbitrarily large number of training windows, in effect converting DEFQA to an unsupervised method. We show experimentally that the new unsupervised method is viable, that it outperforms alternative (i), and that it helps the search engine handle definition questions significantly better than on its own.

2 Attributes of DEFQA

DEFQA represents each window as a vector comprising the values of the following attributes:[6]

SN: The ordinal number of the window in the document, in our case Web page, it originates from. The intuition is that windows that mention the target term first in a document are more likely to define it.

WC: What percentage of the 20 words that are most frequent across all the windows of the target term are present in the particular window represented by the vector. A stop-list and a stemmer are applied first when computing WC.[7] In effect, the 20 most frequent words form a simplistic centroid of all the candidate answers, and WC measures how close the vector's window is to this centroid.

RK: The ranking of the Web page the window originates from, as returned by the search engine.

Manual patterns: 13 binary attributes, each signalling whether or not the window matches a different manually constructed lexical pattern (e.g., "*target*, a/an/the", as in "Tony Blair, the British prime minister"). The patterns are those used by Joho and Sanderson, and four more added by M&A. They are intended to perform well across text genres.

Automatic patterns: A collection of m binary attributes, each showing if the window matches a different automatically acquired lexical pattern. The patterns are sequences of n tokens ($n \in \{1, 2, 3\}$) that must occur either directly before or directly after the target term (e.g., "*target*, which is"). These patterns are acquired as follows. First, all the n-grams that occur directly before or after the target terms in the training windows are collected. The n-grams that have been encountered at least 10 times are considered candidate patterns. From those, the m patterns with the highest precision scores are retained, where *precision* is the number of training definition windows the pattern matches divided by the total number of training windows it matches. We set m to 200, the value that led to the best results in the experiments of M&A. The automatically acquired patterns allow the system to detect definition contexts that are not captured by the manual patterns, including genre-specific contexts.

M&A also explored an additional attribute, which carried the verdict of Prager et al.'s WordNet-based method (2002). However, they found the additional attribute to lead to no significant improvements, and, hence, we do not use it. We have made no attempt to extend the attribute set of M&A; for example, with attributes showing if the window contains the target term in italics, if the window is part of a list that looks like a glossary, or if the window derives from an authority Web page. We leave such extensions for future work. Our contribution is the automatic generation of training examples.

3 Generating training examples

When training DEFQA on windows from Web pages, a mechanism to tag the training windows as definitions or non-definitions is required. Rather than tagging them manually, we use a measure of how similar the wording of each training window is to the wording of definitions of the same target term obtained from on-line encyclopedias and dictionaries. This is possible because we pick training target terms for which there are several definitions in different on-line encyclopedias and dictionaries; hereafter we call these *encyclopedia definitions*.[8] Training windows whose wording is very similar to that of the corresponding encyclopedia definitions are tagged as definition windows (*positive examples*), while windows whose wording differs significantly from the encyclopedia definitions are tagged as non-definitions (*negative examples*). Training windows for which the similarity score does not indicate great similarity or dissimilarity to the wording of the encyclopedia definitions are excluded from DEFQA's

[6] SN and WC originate from Joho and Sanderson (2000).

[7] We use the 100 most frequent words of the British National Corpus as the stop-list, and Porter's stemmer.

[8] We use randomly selected entries from the index of http://www.encyclopedia.com/ as training terms, and Google's 'define:' feature, that returns definitions from on-line encyclopedias and dictionaries, to obtain the encyclopedia definitions.

training, as they cannot be tagged as positive or negative examples with sufficiently high confidence.

Note that encyclopedia definitions are used only to tag training windows. Once the system has been trained, it can be used to discover on ordinary Web pages definitions of terms for which there are no encyclopedia definitions, and indeed this is the main purpose of the system. Note also that we train DEFQA on windows obtained from Web pages returned by the search engine for training terms. This allows it to learn characteristics of the particular search engine being used; for example, what weight to assign to RK, depending on how much the search engine succeeds in ranking pages containing definitions higher. More importantly, it allows DEFQA to select lexical patterns that are indicative of definitions in Web pages, as opposed to patterns that are indicative of definitions in electronic encyclopedias and dictionaries. The latter explains why we do not train DEFQA directly on encyclopedia definitions; another reason is that DEFQA requires both positive and negative examples, while encyclopedia definitions provide only positive ones.

We now explain how we compute the similarity between a training window and the collection C of encyclopedia definitions for the window's target term. We first remove stop-words, punctuation, other non-alphanumeric characters and the target term from the training window, and apply a stemmer, leading to a new form W of the training window. We then compute the similarity of W to C as:

$$sim(W, C) = 1/|W| \cdot \Sigma_{i=1}^{|W|} sim(w_i, C)$$

where $|W|$ is the number of distinct words in W, and $sim(w_i, C)$ is the similarity of the i-th distinct word of W to C, defined as follows:

$$sim(w_i, C) = \mathit{fdef}(w_i, C) \cdot \mathit{idf}(w_i)$$

$\mathit{fdef}(w_i, C)$ is the percentage of definitions in C that contain w_i, and $\mathit{idf}(w_i)$ is the inverse document frequency of w_i in the British National Corpus (BNC):

$$\mathit{idf}(w_i) = 1 + \mathit{log}\frac{N}{\mathit{df}(w_i)}$$

N is the number of documents in BNC, and $\mathit{df}(w_i)$ the number of BNC documents where w_i occurs; if w_i does not occur in BNC, we use the lowest df score of BNC. $sim(w_i, C)$ is highest for words that occur in all the encyclopedia definitions and are used rarely in English. A training window with a large proportion of such words most probably defines the target term. More formally, given two thresholds t_+ and t_- with $t_- \leq t_+$, we tag W as a positive example if $sim(W, C) \geq t_+$, as a negative example if $sim(W, C) \leq t_-$, and we exclude it from the training of DEFQA if $t_- < sim(W, C) < t_+$. Hereafter, we refer to this method of generating training examples as the *similarity method*.

To select reasonable values for t_+ and t_-, we conducted a preliminary experiment for $t_- = t_+ = t$; i.e., both thresholds were set to the same value t and no training windows were excluded. We used $q = 130$ training target terms from TREC definition questions, for which we had multiple encyclopedia definitions. For each term, we collected the $r = 10$ most highly ranked Web pages.[9] To alleviate the class imbalance problem, whereby the positive examples (definitions) are much fewer than the negative ones (non-definitions), we kept only the first 5 windows from each Web page ($SN \leq 5$), based on the observation that windows with great SN scores are almost certainly non-definitions; we do the same in the training stage of all the experiments of this paper, and at run-time, when looking for windows to report, we ignore windows with $SN > 5$. From the resulting collection of training windows, we selected randomly 400 windows, and tagged them both manually and via the similarity method, with t ranging from 0 to 1. Figures 1 and 2 show the precision and recall of the similarity method on positive and negative training windows, respectively, for varying t. Here, *positive precision* is the percentage of training windows the similarity method tagged as positive examples (definitions) that were indeed positive; the true classes of the training windows were taken to be those assigned by the human annotators. *Positive recall* is the percentage of truly positive examples that the similarity method tagged as positive. *Negative precision* and *negative recall* are defined similarly.

Figures 1 and 2 indicate that there is no single threshold t that achieves both high positive precision and high negative precision. To be confident

[9] In all our experiments, we used the Altavista search engine.

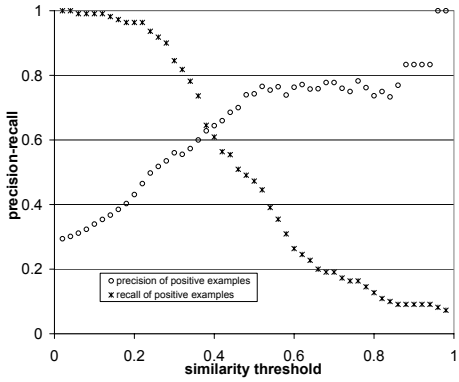

Figure 1: Positive precision and recall

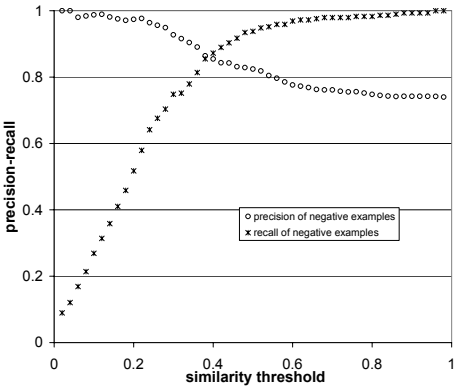

Figure 2: Negative precision and recall

that the training windows the similarity method will tag as positive examples are indeed positive (high positive precision), one has to set t close to 1; and to be confident that the training windows the similarity method will tag as negative examples are indeed negative (high negative precision), t has to be set close to 0. This is why we use two separate thresholds and discard the training windows whose similarity score is between t_- and t_+. Figures 1 and 2 also indicate that in both positive and negative examples the similarity method achieves perfect precision only at the cost of very low recall; i.e., if we insist that all the resulting training examples must have been tagged correctly (perfect positive and negative precision), the resulting examples will be very few (low positive and negative recall). There is also another consideration when selecting t_- and t_+: the ratio of positive to negative examples that the similarity method generates must be approximately the same as the true ratio before discarding any training windows, in order to avoid introducing an artificial bias in the training of DEFQA's SVM; the true ratio among the 400 training windows before discarding any windows was approximately $0.37 : 1$.

Based on the considerations above, in the remaining experiments of this paper we set t_+ to 0.5. In Figure 1, this leads to a positive precision of 0.72 (and positive recall 0.49), which does not improve much by adopting a larger t_+, unless one is willing to set t_+ at almost 1 at the price of very low positive recall. In the case of t_-, setting it to any value less than 0.34 leads to a negative precision above 0.9, though negative recall drops sharply as t_- approaches 0 (Figure 2). For example, setting t_- to 0.32, leads to 0.92 negative precision, 0.75 negative recall, and approximately the same positive to negative ratio ($0.31 : 1$) as the true observed ratio. In the experiments of Section 4, we keep t_+ fixed to 0.5, and set t_- to the value in the range $(0, 0.34)$ that leads to the positive to negative ratio that is closest to the true ratio we observed in the 400 windows.

The high negative precision we achieve (> 0.9) suggests that the resulting negative examples are almost always truly negative. In contrast, the lower positive precision (0.72) indicates that almost one in every four resulting positive examples is in reality a non-definition. This is a point where our similarity method needs to be improved; we return to this point in Section 6. Our experiments, however, show that despite this noise, the similarity method already outperforms the alternative of training DEFQA on TREC data. Note also that once the thresholds have been selected, we can generate automatically an arbitrarily large set of training examples, by starting with a sufficiently large number q of training terms to compensate for discarded training examples.

4 Evaluation

We tested two different forms of DEFQA. The first one, dubbed DEFQA^t, was trained on the $q = 160$ definition questions of TREC-2000 and TREC-2001 and the corresponding TREC documents, resulting in 3,800 training windows.[10] The second form of DE-

[10] For each question, the TREC organizers provide the 50 most highly ranked documents that an IR engine returned from the TREC document collection. We keep the top $r = 10$ of these documents, while M&A kept all 50. Furthermore, as discussed in Section 3, we retain up to the first 5 windows from each doc-

FQA, dubbed DEFQA^s, was trained via the similarity method, with $q = 480$ training target terms, leading to 7,200 training windows; as discussed in Section 3, one of the advantages of the similarity method is that one can generate an arbitrarily large set of training windows. As in the preliminary experiment of Section 3, r (Web pages per target term) was set to 10 in both systems. To simplify the evaluation and test DEFQA in a more demanding scenario, we set k to 1, i.e., the systems were allowed to return only one snippet per question, as opposed to the more lenient $k = 5$ in the experiments of M&A.

We also wanted a measure of how well DEFQA^t and DEFQA^s perform compared to a search engine on its own. For this purpose, we compared the performance of the two systems to that of a baseline, dubbed BASE^1, which always returns the first window of the Web page the search engine ranked first. In a search engine that highlights question terms in the returned documents, the snippet returned by BASE^1 is presumably the first snippet a user would read hoping to find an acceptable definition. To study how much DEFQA^t and DEFQA^s improve upon random behaviour, we also compared them to a second baseline, BASE^r, which returns a randomly selected window among the first five windows of all r Web pages returned by the search engine.

All four systems were evaluated on 81 unseen target terms. Their responses were judged independently by two human assessors, who had to mark each response as containing an acceptable short definition or not. As already pointed out, DEFQA^t and DEFQA^s consult encyclopedia definitions only during training, and at run time the systems are intended to be used with terms for which no encyclopedia definitions are available. During this evaluation, however, we deliberately chose the 81 test terms from the index of an on-line encyclopedia. This allowed us to give the encyclopedia's definitions to the assessors, to help them judge the acceptability of the single-snippet definitions the systems located on Web pages; many terms where related to, for example, medicine or biology, and without the encyclopedia's definitions the assessors would not be aware of their meanings. The following is a snippet returned correctly by DEFQA^s for 'genome':

discipline comparative genomics functional genomics bioinformatics the emergence of genomics as a discipline in 1920 , the term genome was proposed to denote the totality of all genes on all chromosomes in the nucleus of a cell . biology has...

while what follows is a non-definition snippet returned wrongly by BASE^1:

what is a genome national center for biotechnology information about ncbi ncbi at a glance a science primer databases...

The examples illustrate the nature of the snippets that the systems and assessors had to consider. The snippets often contain phrases that acted as links in the original pages, or even pieces of programming scripts that our rudimental preprocessing failed to remove. (We remove only HTML tags, and apply a simplistic tokenizer.) Nevertheless, in most cases the assessors had no trouble agreeing whether or not the resulting snippets contained acceptable short definitions. K_{Co} was 0.80, 0.81, 0.90, 0.89, and 0.86 in the assessment of the responses of DEFQA^s, DEFQA^t, BASE^r, BASE^1, and all responses, respectively, indicating strong inter-assessor agreement.[11] The agreement was slightly lower in DEFQA^s and DEFQA^t, because there were a few marginally acceptable or truncated definitions the assessors were uncertain about. There were also 4 DEFQA^s answers and 3 BASE^1 answers that defined secondary meanings of the target terms; e.g., apart from a kind of lizard, 'gecko' is also the name of a graphics engine, and 'Exodus' is also a programme for ex-offenders. Such answers were counted as wrong, though this may be too strict. With a larger k, there would be space to return both the main and secondary meanings, and the evaluation could require this.

Table 1 shows that DEFQA^s answered correctly approximately 6 out of 10 definition questions. This is lower than the score reported by M&A (73%), but remarkably high given that in our evaluation the systems were allowed to return only one snippet per question; i.e., the task was much harder than in M&A's experiments. DEFQA^s answered correctly more than twice as many questions as DEFQA^t, despite the fact that its training data contained a lot of noise. (Single-tailed difference-of-proportions tests show that all the differences of Table 1 are statisti-

ument. This is why we have fewer training windows than M&A.

[11] We follow the notation of Di Eugenio and Glass (2004). The $K_{S\&C}$ figures were identical. The $2 \cdot P(A) - 1$ figures were 0.80, 0.85, 0.95, 0.95, and 0.89 respectively.

	assessor 1	assessor 2	average
BASE^r	14.81 (12)	14.81 (12)	14.81 (12)
BASE^1	14.81 (12)	12.35 (10)	13.58 (11)
DEFQA^t	25.93 (21)	25.93 (21)	25.93 (21)
DEFQA^s	55.56 (45)	60.49 (49)	58.02 (47)

Table 1: Percentage of questions answered correctly

cally significant at $\alpha = 0.001$.) The superiority of DEFQA^s appears to be mostly due to its automatically acquired patterns. DEFQA^t too was able to acquire several good patterns (e.g., 'by *target*', 'known as *target*', '*target*, which is', '*target* is used in'), but its pattern set also comprises a large number of irrelevant n-grams; this had also been observed by M&A. In contrast, the acquired pattern set of DEFQA^s is much cleaner, with much fewer irrelevant n-grams, which is probably due to the largest, almost double, number of training windows. Furthermore, the pattern set of DEFQA^s contains many n-grams that are indicative of definitions on the Web. For example, many Web pages that define terms contain text of the form "What is a *target*? A *target* is...", and DEFQA^s has discovered patterns of the form 'what is a/an/the *target*', '? A/an/the *target*', etc. It has also discovered patterns like 'FAQ *target*', 'home page *target*', '*target* page' etc., that seem to be good indications of Web windows containing definitions.

Overall, DEFQA's process of acquiring lexical patterns worked better in DEFQA^s than in DEFQA^t, and we believe that the performance of DEFQA^s could be improved further by acquiring more than 200 patterns; we hope to investigate this in future work, along with an investigation of how the performance of DEFQA^s relates to q, the number of training target terms. Finally, note that the scores of both baselines are very poor, indicating that DEFQA^s performs significantly better than picking the first, or a random snippet among those returned by the search engine.

5 Related work

Definition questions have recently attracted several QA researchers. Many of the proposed approaches, however, rely on manually crafted patterns or heuristics to identify definitions, and do not employ learning algorithms (Liu et al., 2003; Fujii and Ishikawa, 2004; Hildebrandt et al., 2004; Xu et al., 2004).

Ng et al. (2001) use machine learning (C5 with boosting) to classify and rank candidate answers in a general QA system, but they do not treat definition questions in any special way; consequently, their worst results are for "What...?" questions, that presumably include definition questions. Ittycheriah and Roukos (2002) employ a maximum entropy model to rank candidate answers in a general-purpose QA system. Their maximum entropy model uses a very rich set of attributes, that includes 8,500 n-gram patterns. Unlike our work, their n-grams are five or more words long, they are coupled to two-word question prefixes, and, in the case of definition questions, they do not need to be anchored at the target term. The authors, however, do not provide separate performance figures for definition questions.

Blair-Goldensohn et al. (2003) focus on definition questions, but aim at producing coherent multi-snippet definitions, rather than single-snippet definitions. The heart of their approach is a component that uses machine learning (Ripper) to identify sentences that can be included in the multi-sentence definition. This component plays a role similar to that of our SVM, but it is intended to admit a larger range of sentences, and appears to employ only attributes conveying the ordinal number of the sentence in its document and the frequency of the target term in the sentence's context.

Since TREC-2003, several researchers have proposed ways to generate multi-snippet definitions (Cui et al., 2004; Fujii and Ishikawa, 2004; Hildebrandt et al., 2004; Xu et al., 2004). The typical approach is to locate definition snippets, much as in our work, and then report definition snippets that are sufficiently different; most of the proposals use some form of clustering to avoid reporting redundant snippets. Such methods could also be applied to DEFQA, to extend it to the post-2003 TREC task.

On-line encyclopedias and dictionaries have been used to handle definition questions in the past, but not as in our work. Hildebrandt et al. (2004) look up target terms in encyclopedias and dictionaries, and then, knowing the answers, try to find supporting evidence for them in the TREC document collection. Xu et al. (2004) collect from on-line encyclopedias and dictionaries words that co-occur with the target term; these words and their frequencies are then used as a centroid of the target term, and candidate

answers are ranked by computing their similarity to the centroid. This is similar to our WC attribute.

Cui et al. (2004) also employ a form of centroid, comprising words that co-occur with the target term. The similarity to the centroid is taken into consideration when ranking candidate answers, but it is also used to generate training examples for a learning component that produces soft patterns, in the same way that we use the similarity method to produce training examples for the SVM. As in our work, the training examples that the centroid generates may be noisy, but the component that produces soft patterns manages to generalize over the noise. To the best of our knowledge, this is the only other unsupervised learning approach for definition questions that has been proposed. We hope to compare the two approaches experimentally in future work. For the moment, we can only point out that Cui et al.'s centroid approach generates only positive examples, while our similarity method generates both positive and negative ones; this allows us to use a principled SVM learner, as opposed to Cui et al.'s more ad hoc soft patterns that incorporate only positive examples.

6 Conclusions and future work

We presented an unsupervised method to learn to locate single-snippet answers to definition questions in QA systems that supplement Web search engines. The method exploits on-line encyclopedias and dictionaries to generate automatically an arbitrarily large number of positive and negative definition examples, which are then used to train an SVM to separate the two classes. We have shown experimentally that the proposed method is viable, that it outperforms training the QA system on TREC data, and that it helps the search engine handle definition questions significantly better than on its own.

We have already pointed out the need to improve the positive precision of the training examples. One way may be to combine our similarity method with Cui et al.'s centroids. We also plan to study the effect of including more automatically acquired patterns and using more training target terms. Finally, our method can be improved by including attributes for the layout and authority of Web pages.

References

S. Blair-Goldensohn, K.R. McKeown, and A.H. Schlaikjer. 2003. A hybrid approach for answering definitional questions. Technical Report CUCS-006-03, Columbia University.

H. Cui, M.-Y. Kan, and T.-S. Chua. 2004. Unsupervised learning of soft patterns for generating definitions from online news. In *Proceedings of WWW-2004*, pages 90–99, New York, NY.

B. Di Eugenio and M. Glass. 2004. The kappa statistic: A second look. *Comput. Linguistics*, 30(1):95–101.

A. Fujii and T. Ishikawa. 2004. Summarizing encyclopedic term descriptions on the Web. In *Proceedings of COLING-2004*, pages 645–651, Geneva, Switzerland.

W. Hildebrandt, B. Katz, and J. Lin. 2004. Answering definition questions using multiple knowledge sources. In *Proceedings of HLT-NAACL 2004*, pages 49–56, Boston, MA.

A. Ittycheriah and S. Roukos. 2002. IBM's statistical question answering system – TREC-11. In *Proceedings of TREC-2002*.

H. Joho and M. Sanderson. 2000. Retrieving descriptive phrases from large amounts of free text. In *Proc. of the 9th ACM Conference on Information and Knowledge Management*, pages 180–186, McLean, VA.

B. Liu, C.W. Chin, and H.T. Ng. 2003. Mining topic-specific concepts and definitions on the Web. In *Proceedings of WWW-2003*, Budapest, Hungary.

S. Miliaraki and I. Androutsopoulos. 2004. Learning to identify single-snippet answers to definition questions. In *Proceedings of COLING-2004*, pages 1360–1366, Geneva, Switzerland.

H.T. Ng, J.L.P. Kwan, and Y. Xia. 2001. Question answering using a large text database: A machine learning approach. In *Proceedings of EMNLP-2001*, pages 67–73, Pittsburgh, PA.

J. Prager, J. Chu-Carroll, and K. Czuba. 2002. Use of WordNet hypernyms for answering what-is questions. In *Proceedings of TREC-2001*.

B. Scholkopf and A. Smola. 2002. *Learning with kernels*. MIT Press.

E.M. Voorhees. 2003. Evaluating answers to definition questions. In *Proceedings of HLT-NAACL 2003*, pages 109–111, Edmonton, Canada.

J. Xu, R. Weischedel, and A. Licuanan. 2004. Evaluation of an extraction-based approach to answering definitional questions. In *Proceedings of SIGIR-2004*, pages 418–424, Sheffield, U.K.