Proceedings of the Conference on Empirical Methods in Natural
Language Processing (EMNLP), Philadelphia, July 2002, pp. 265-272.
Association for Computational Linguistics.

A Machine Learning Approach to Introspection in a Question Answering System

Krzysztof Czuba
Carnegie-Mellon University
Pittsburgh, PA 15213
kczuba@cs.cmu.edu

John Prager
IBM T.J. Watson Research Center
Yorktown Heights, NY 10598
jprager@us.ibm.com

Jennifer Chu-Carroll
IBM T.J. Watson Research Center
Yorktown Heights, NY 10598
jencc@us.ibm.com

Abstract

The ability to evaluate intermediate results in a Question Answering (QA) system, which we call *introspection*, is necessary in architectures based on planning or on processing loops. In particular, it is needed to determine if an earlier phase must be retried, or if the response "No Answer" must be offered. We look at an introspection task of performing a cursory evaluation of the search engine output in a QA system. We define this task as a concept-learning problem and evaluate two classifiers that use features based on score progression in the ranked list returned by the search engine and candidate answer types. Our experiments showed promising results, achieving 25% relative improvement over a majority class baseline on unseen data.

1 Introduction

Recent experience in building Question Answering (QA) systems shows that the answer search strategy should depend on the question and/or answer type, and intermediate results produced during search. Practically all state-of-the-art QA systems identify the sought answer type and some vary their search strategy accordingly (Prager et al. (2000), Hovy et al. (2001), Harabagiu et al. (2000)). It is less common for QA systems to examine intermediate results during search, although the success of Falcon, which uses a number of feedback loops, suggests that it is a promising approach (Paşca and Harabagiu (2001)).

In order to implement such feedback loops, the system needs an *introspection mechanism* to be able to examine the quality of the intermediate results and make a decision to redo a step if the quality is not satisfactory. Similarly, in plan-driven architectures (Hiyakumoto (2001)), the planner requires information on the quality of an operator's results to decide which operator to apply next.

A related problem has been posed by the TREC-10 QA-Track (Voorhees (2001)): the test set contained questions with no answer in the document collection and the systems were supposed to detect them. A search strategy that always produces a set of answers can only be misleading in such a situation. Similarly, deployed QA systems need mechanisms to detect questions with no answers in the collection, or with no answers found reliably using the algorithms the system implements.

We examine a machine-learning approach to an introspection task in a QA system. We concentrate on the search engine output in our system (Prager et al. (2000)). We train a set of classifiers that use features based on the progression of the scores the search engine assigns to documents in a hit list, in order to try and predict whether the hit list contains a good answer candidate. We evaluate the classifiers on different question types and report a 25% relative gain in accuracy over a majority class baseline for the important class of definition questions.

The paper is organized as follows. Section 2 presents a short overview of our QA system.

Section 3 describes how we defined the introspection problem as a concept learning task and the feature sets we used. The experimental design and data sets are discussed in Section 4, followed by the experimental results in Section 5 and discussion in Section 6.

2 System Architecture

The flavor of QA considered here is as defined by the TREC-10 conference (Voorhees (2001)): for a given factual question in English, the system is supposed to find in a large, cross-domain document collection a 50-byte text segment that answers this question. For example, for the question *What is the state flower of Hawaii?* acceptable answers expected from the system are *hibiscus*, *yellow hibiscus*, etc. For the majority of the TREC QA-track questions the answer can be formulated in the form of a named entity (NE) of a certain type.

Our QA system implements the technique of Predictive Annotation (Prager et al. (2000)) that creates textual annotation in the collection documents in the form of QA-Tokens. QA-Tokens are semantic class labels associated with recognized text segments (mostly Named Entities). Examples of QA-Tokens include ones corresponding to the well-known MUC NE categories such as PERSON$, PLACE$ or DATE$. There are approximately 80 QA-Tokens, and thus answer types, and they are used to focus search throughout the system; they are indexed and can occur in search engine queries.

The system's architecture is a pipeline starting with question analysis, followed by the GuruQA search engine, and an answer selection component (Radev et al. (2000)). The question is first transformed into a query containing question keywords and a set of QA-Tokens corresponding to the recognized question/answer type. The search engine is designed to retrieve a ranked hit list of passages of a dynamically determined size (1–3 sentences; most passages contain only one sentence).

Passage ranking is based on a combination match (Prager et al. (2000)). A passage score consists of two components: the *term score* and the *density score*. The term score is computed by summing the weights of each query term found in the passage. The term weights are set in query analysis, roughly, 400 for QA-Tokens, 200 for proper names and 100 for common words. The density score, which effectively acts as a tie-breaker, is inversely proportional to the distance (in words) between the first and last passage words that matched a query keyword; all matched words being consecutive gives a density score of 99.

The order of the words is irrelevant. The search engine holds off matching QA-Tokens till last, and ensures that they do not match words that had already matched. For example, the question: *What is the capital of Sri Lanka?* is converted by the question analysis module to "PLACE$ capital Sri_Lanka" (weight and window operators not shown); it is important for the PLACE$ QA-Token not to match "Sri Lanka" in text.

Because of the high weight assigned to QA-Tokens, the returned passages are very likely to contain an answer candidate of the desired type. The hit list, the analyzed question and the query are given to the answer selection module.

Not all question types are processed using the above strategy. Two notable exceptions are definition-type (such as *What is thalassemia?*, or *Who was Jane Goodall?*) and abbreviation questions. For definition-type questions, the system employs the Virtual Annotation technique (Prager et al. (2001)) to find defining terms in WordNet. Abbreviation-finding is an auxiliary role of our named-entity recognizer Textract (Byrd and Ravin, (1999)). Similarly, for abbreviations possible expansions are found based on collection statistics.

Additionally, for definition questions, the defining expressions are located using syntactic patterns such as appositives or copular expressions.

3 Hit List Quality as Concept Learning

We cast the task of determining the quality of a hit list as a concept-learning problem. Our goal is to develop classifiers using features which can be extracted from the hit lists returned by the search engine to classify a hit list as "good", i.e., at least one passage in the hit list contains the answer to the question, or "bad", i.e., the answer is not present in the hit list passages.

Because of the use of QA-Tokens, hit lists returned by the search engine are highly focused on Named Entities of the right type — assuming the QA-Token identification in question analysis and the passage annotation were correct. We hypothesize that the score assigned to each passage by the search engine provides important information about the passage quality, and the patterns in which the scores change within the hit list may also indicate the hit list quality. For example, if the score for the top passage is far from the maximum achievable score given the query terms, it is an indication that many important keywords were not present in the passage, and as a result, we may safely conclude that the passage is not a good candidate for finding a correct answer. The correlation of score change patterns with hit list quality may also vary according to QA-Token types, in particular the frequency of QA-Tokens in the corpus. High passage scores indicating that a Name Entity of the right semantic type is present in the passage may be better evidence that the correct answer is present if the candidate QA-Token is relatively rare (e.g., BODYPART$) than if it is very frequent (e.g., PERSON$), since in the latter case, the QA-Token has an increased possibility of being present by chance.

We identified several classes of features which we believe may be correlated with hit list quality. Let N be the hit list length (typically 10), T_{max} and D_{max} be the maximum achievable term and density scores, respectively, and T_i and D_i be the term and density scores of the i-th document in the hit list, respectively. We define $T_0 = T_{max}$ and $D_0 = D_{max}$. The classes of features we chose that are based on search engine scores are defined as follows:

- **F**: N features calculated from the relative change in the term score: $f_i = (T_i - T_{i-1})/T_{i-1}$, $1 \leq i \leq N$.
- **S**: $N-1$ features calculated from the sign of the second derivative of the score function: $s_i = sign(f_{i+1} - f_i)$, $1 \leq i < N$;
- **D**: N features calculated from the relative change in the density score: $d_i = (D_i - D_{i-1})/D_{i-1}$, $1 \leq i \leq N$.

The features in **F** and **S** are based only on the term score, i.e., they reflect the change in the number and kind (QA-Token, proper name, other) of keyword matches. **D** is based solely on the density scores. Additionally, we introduced two alternative feature classes based on the QA-Tokens identified by the question analysis:

- **T**: 83 binary features, each of which corresponds to a QA-Token type in the system.
- **P**: 47 binary features, similar to **T**; only the frequently-occurring QA-Tokens are represented, the remaining QA-Tokens[1] are combined to form the type OTHER.

These sets make the question type information available to the classifier and are motivated by the intuition that some question types might be more amenable to our classification technique. **P** is introduced in an attempt to reduce the data sparseness problem for rare types.

Given the above feature classes, we proceed to design an experiment to investigate the correlation between these feature classes and hit list quality, which we describe in the next section.

4 Experimental Design

Our experiments seek to answer three questions. We are interested in knowing 1) which of the above feature classes are best predictors of hit list quality; 2) how well the best performing set of feature classes predicts hit list quality; 3) whether particular question types (those seeking answers of particular QA-Token types) are better candidates for our prediction mechanism. This section describes the data and learning algorithms used in our experiments, as well as the experimental settings.

[1]QA-Tokens instantiated for fewer than 10 questions in the combined TREC-9 and TREC-10 sets.

	TREC-9	**TREC-10**
Total number of examples	688	498
% positive examples	70.6%	53.4%

Table 1: Percentage of positive examples in the **FULL** sets

	TREC-9	**TREC-10**	**Question Types**
ABBREV	83 (66.2%)	136 (56.6%)	Abbreviation questions
DEFINE	84 (65.7%)	144 (52%)	Definition questions
ORG	70 (70.0%)	24 (66.6%)	ORG$
PERSON	143 (74.8%)	69 (60.8%)	PERSON$ PRESIDENT$ ROYALTY$ etc.
PLACE	114 (77.1%)	63 (57.1%)	PLACE$ CITY$ CAPITAL$ etc.
THING	120 (59.1%)	193 (50.2%)	THING$
TIME	76 (67.1%)	53 (77.3%)	YEAR$ DATE$ AGO$ TIMEOFYEAR$ etc.
FILTERED	491 (74.5%)	252 (58.3%)	Excludes: THING$, definition questions, and abbreviation questions

Table 2: Statistics for the data subsets: columns TREC-9 and TREC-10 show the number of questions in each set with the percentage of positive examples, i.e., examples with correct answers in the hit list

4.1 Experimental Data

We ran our system on the TREC-9 and TREC-10 question sets, containing a total of 1193 questions. This resulted in a set of QA-Tokens being associated with each question. Due to analysis errors, 7 questions were not assigned any QA-Tokens, and were unsuitable for further analysis. For the remaining questions, henceforth referred to as the **FULL** data sets, we matched the Named Entities in the retrieved passages against answer patterns[2] to identify ones corresponding to correct answers and classified each hit list according to whether it contained a correct answer. Table 1 shows the percentage of positive examples (hit lists containing a good candidate) in the **FULL** data sets.

In order to investigate the performance of our classifiers on subsets of question types, we split the **FULL** sets based on the frequency of candidate QA-Tokens and on QA-Token semantics. We selected those question types that occurred more than 100 times in the combined **FULL** data sets. The cardinalities of the resulting 7 data subsets[3] and the percentages of positive examples are shown in Table 2.

The last row in Table 2 shows the **FILTERED** data sets, which exclude from the **FULL** sets definition and abbreviation questions, along with questions assigned the QA-Token THING$. The former two classes are excluded in order to eliminate the effects of different processing strategies specific to these questions; THING$ questions were eliminated because their extension covers practically all noun phrases not assigned to other QA-Tokens. The **FULL** and **FILTERED** sets are the largest but also least specific. The motivation behind using these sets was to test if given a large number of examples, classifier features could be found that are good predictors across different question types.

4.2 Machine Learning Algorithms

To run our experiments, we used the Weka package (WEKA (2002), Witten et al. (1999)). We experimented with a number of different Boolean classifiers including OneR, Naive Bayes, decision trees and neural networks. We obtained the best results with the latter two and we describe them below. We used the decision tree (C4.5) and neural net classifier implementation from the Weka package. We used the default configuration for the decision tree algorithm,

[2]For TREC-9 we used the answer patterns provided by NIST, for TREC-10 we used our own patterns combined with patterns provided by Ken Litkowski of CL Research.

[3]We added the **ORG** set since it corresponds to only one well-defined QA-Token and it is an important question type.

with the exception of allowing for binary splits. For the neural network, we trained a plain feed-forward network with one hidden layer. If the feature set had N features, the number of hidden units was set to $\lceil N/2 \rceil$. We set the number of epochs for backpropagation to a fixed number of 500. In the rest of the paper, the decision tree results are labeled **DT**, the neural net results are labeled **NN**.

4.3 Experimental Settings

To find out which subsets of the feature classes identified in Section 3 best predict hit list quality, we conducted extensive experiments by varying the feature classes used by the classifiers. Since we consider the term scores the most important indicator of passage quality, all experiments included the feature class **F**. We then generated all possible combinations based on inclusion/exclusion of all other feature classes, which can be described by the following regular expression: **FS**?**D**?(**T**|**P**)?. For example, the feature set **FSD** contains all features from classes **F**, **S** and **D**. For each combination, we performed 10-fold stratified cross-validation on the data subsets shown in Tables 1 and 2. Our experiments focused on selecting the optimal feature classes rather than the optimal learners' parameters such as the number of epochs, which we leave for future research. A more fine-grained feature selection in which single features from the different sets are picked is also possible but not investigated here.

5 Results and Analysis

We conducted experiments as described in Section 4.3, and identified, for each data set, the best performing set of feature classes. Table 3 shows these best performing feature classes and the absolute success rates (the percentages of correctly classified examples in the test set) of the resulting classifiers. As our baseline we adopted the accuracy of a classifier that always predicts the majority class, which amounts to predicting that the hit list is always "good". This baseline corresponds to no feedback available to the QA system. The relative percentage gains on the baseline are also presented in Table 3. The results for the TREC-10 data subsets **ORG** and **TIME** are shown in italics because these sets are significantly smaller, and we consider their results to be less reliable.

For the vast majority of the data sets we observe gains in accuracy over the baseline, some quite substantial, achieving around or over 20% relative improvement. The gains are generally higher for TREC-10 subsets. This is contributed partly by lower baselines: the TREC-9 question set was used to fine-tune the system's query formulation strategy, including query expansion and rules for assigning QA-Tokens, and the TREC-10 data were new to the system. However, the lower baseline on the TREC-10 data sets indicates question analysis module deficiencies on unseen data, and until its performance plateaus, we argue that the performance of the classifiers on the TREC-10 data sets is more indicative of what could be expected on unseen data.

There is also considerable variation in the performance for different question types. The **DEFINE** class performed consistently well, with a 25.3% relative gain for the neural net trained on the TREC-10 subset, which is a valuable result given the high percentage of such questions in the TREC question sets.

A few of the types show mixed performance. The **FILTERED** set shows a much higher accuracy on TREC-10. For the **ABBREV**, **ORG** and **THING** (DT) subsets, the accuracy is better on TREC-9. The **ORG** type seems to be least suitable for the method since the performance is below the baseline for both sets.

For the **FILTERED** and **FULL** sets we observe that the optimal feature sets contain **T** or **P**, which suggests that the question type information is crucial for good performance.

The neural net performed consistently better on the important **DEFINE** class, and produced non-negative gains on all TREC-10 subsets. This small sample suggests that a neural network could be very effective for this task given a more appropriate training set-up (e.g., with a validation set and early stopping) than the fixed strategy we adopted.

It is interesting to note the effect of feature

	TREC-9						**TREC-10**					
	DT			NN			**DT**			NN		
FULL	**FDT**	72.8	+3.1	**FT**	71.5	+1.2	**FSDP**	63.4	+18.7	**FDP**	62.2	+16.4
ABBREV	**FSD**	73.3	+10.7	**FS**	69.8	+5.4	**FS**	55.2	-2.5	**FS**	57.1	+0.8
DEFINE	**FSDT**	71.1	+8.2	**FDT**	72.3	+10.0	**FP**	62.2	+19.6	**FDP**	65.2	+25.3
ORG	**F**	67.1	-4.1	**FD**	68.5	-2.1	***FSD***	*58.3*	*-12.5*	***FSD***	*66.6*	*0*
PERSON	**F**	75.4	+0.8	**F**	74.1	-0.9	**FSDT**	73.8	+21.4	**FP**	70.7	+16.2
PLACE	**FD**	79.0	+2.5	**FDP**	81.8	+6.0	**F**	69.0	+20.8	**F**	67.1	+17.5
THING	**FD**	68.3	+15.6	**FS**	59.1	0	**FS**	53.4	+6.4	**F**	59.5	+18.5
TIME	**FSD**	71.6	+6.7	**FD**	66.0	-1.6	***FDSP***	*78.3*	*+1.3*	***FSD***	*82.6*	*+6.8*
FILTERED	**FSDT**	72.2	-3.1	**FS**	73.5	-1.3	**FST**	68.7	+17.8	**FST**	67.4	+15.6

Table 3: Best performing feature sets, accuracy and relative percentage gains

selection on the classifier accuracy. For the **DEFINE** set, the accuracy of the neural network ranges from 52.0% on the **FT** subset to 65.2% on the **FDP** subset as reported in Table 3. Also, the richest feature set (**FSDT**) does not guarantee the best performance; in fact, the simplest set (**F**) is optimal in 6 different experiments.

6 Discussion

To better illustrate the performance of the classifiers, we present the precision figures for the "good" and "bad" classes in Table 4. The figures correspond to the percentage of correctly identified members of each class and suggest the performance that can be expected from the classifiers when used in a QA system. We also give the best performing feature set and the overall classifier accuracy in the **All** column. In order to make the estimates more reliable we chose the combined data sets for this discussion.

According to Table 4, the "good" examples can be identified quite reliably for most of the subsets. The least reliable is the **THING** subset, which is not surprising since the THING$ QA-Token is very general and its presence is not a good predictor of the presence of an answer. For the **FULL**, **ABBREV**, **DEFINE** and **PLACE** subsets, close to half of all "bad" hit lists are discovered. Another interesting result is the dominance of the feature set **F** in the table, which suggests that this might be the set of choice for larger training sets.

Given that there were 228 questions in the **DEFINE** subset, the DT results from Table 4 translate to the confusion matrix in Table 5. According to Table 5, in the worst-case scenario, in which all false negatives are lost after the correction step, the loss in the number of questions with "good" hit lists can be as high as 11.8%. In the ideal case, no false negative will result in discarding an answer and all "bad" hit lists will be corrected, which translates to a 20.6% gain in the number of "good" hit lists. The actual gain in the number of good hit lists and consequently the number of answers found needs to be evaluated empirically.

False negatives are examples in which there is a potential answer candidate in the hit list, but the classifier predicted that there was none. Depending on the correction procedure, false negatives are potentially harmful to the accuracy of the QA system. For example, if the correction procedure discards a hit list falsely classified as a negative, it will throw away a good answer candidate which may or may not return in the regenerated hit list. Given that this answer candidate might be the only one in the collection or the only one easily found, discarding the retrieval results might not be the optimal choice. When the original retrieval results are not discarded but are augmented with the second hit list, the subsequent modules in the QA system are faced with an additional set of documents to process. In the simplest case this can incur additional processing time. In systems that rely on a form of redundancy (Clarke et al. (2001)) — the frequency count among answer candidates would be the simplest example — compounding of errors can occur, but, by the same token, so can positive reinforcement. If and how

	DT				NN			
	Set	**All**	**Good**	**Bad**	**Set**	**All**	**Good**	**Bad**
ABBREV	**F**	63.9	78.5	41.9	**FS**	60.2	90.7	14.0
DEFINE	**FT**	65.7	79.2	47.6	**FSP**	64.4	84.6	37.6
ORG	**F**	65.7	85.4	21.6	**FD**	66.7	82.1	30.0
PERSON	**F**	72.1	89.2	30.9	**FDSP**	67.9	81.2	37.1
PLACE	**F**	76.2	94.4	33.3	**F**	74.5	89.6	40.6
THING	**FS**	57.1	60.9	39.0	**F**	59.4	45.1	75.6
TIME	**F**	67.5	83.3	16.6	**FT**	70.5	90.1	21.6
FULL	**FSDT**	65.0	76.0	46.0	**FSP**	66.4	82.8	37.9
FILTERED	**FSDP**	70.3	82.6	43.0	**F**	69.9	84.4	37.8

Table 4: Per-class precision

	Classified as good	**Classified as bad**
good	103	27
bad	51	47

Table 5: Confusion matrix for the **DEFINE** subset

often such harmful situations occur is system-dependent and it could be empirically estimated on a large question set, which we leave for future research.

The precision figures in Table 4 were given for the feature sets that resulted in the highest overall accuracy of the classifier. Given that false positives are potentially harmful, minimizing their number might be another criterion for feature selection. As an example we give the precision for the classifier trained and evaluated on the **FILTERED** subset using the **FS** feature set. In this case 91.7% "good" examples are classified as such, but only 14.7% "bad" examples receive the correct label, with the overall classifier accuracy of 67.9%. Although the overall performance of the classifier is worse than with the **FST** feature set, the cost of discarding an answer candidate might be too high and a feature set that results in a different split of misclassified examples might be a better choice.

We also examined the structure of the decision trees that were learned. For all feature sets that contained question type information, the features corresponding to question types tend to appear high in the tree. For type specific data sets, such as **PERSON** or **PLACE**, features corresponding to the change in the term match tend to be the most discriminative ones, with the features corresponding to the drop from the maximum achievable score to the score of the first passage (f_1) appearing at the top of almost all the trees (the **TIME** set was the exception). The structure of the TREC-10 decision trees was much less systematic.

7 Future directions

We have examined the misclassified examples and — not surprisingly – we found many cases that should have been classified differently on semantic grounds, which is not possible given our feature sets. Nevertheless, on many of the question types we looked at, our method can detect approximately half of the "bad" hit lists. The performance of the classifiers is especially high for the **DEFINE** question set whose members are relatively easily identified automatically. Definition type questions are a very important subset as they are usually quite numerous both in the TREC evaluations and in real-life question logs. The actual improvement in the number of questions the system can answer correctly needs to be evaluated in an end-to-end test given a specific correction procedure.

Another experiment would be to train the classifiers only on questions for which the QA-Token assignment was correct. This would require manual verification of the assignment. Such verification might be slightly difficult since many QA-Tokens could potentially retrieve a correct answer. It is possible that a correct answer is found even if the QA-Token is not optimally assigned, but there are still likely to be cases for which the current set of rules used to pick the QA-Token result in a wrong choice from which the subsequent modules cannot recover. Thus, only clearly wrong QA-Tokens, i.e., QA-Tokens that are unlikely to retrieve the correct answer would have to be discarded.

Another remaining issue is the difference in performance on the data sets drawn from the TREC-10 questions in comparison to the sets derived from TREC-9. Currently we do not have a good explanation for the difference. We will monitor the performance of the method as the overall system coverage improves to see if the difference was due to the incomplete coverage on the new TREC-10 question set.

8 Conclusions

We presented an approach to implementing an introspection mechanism in a QA system that evaluates the search engine output based solely on the scores assigned to the documents in the returned hit list. Despite a limited amount of information, we showed promising results on easily identified question types. The method is particularly effective for the important definition class questions and it results in a 25.3% relative improvement over a majority class baseline in the number of correctly identified "bad" hit lists for the neural network classifier with the optimal feature set.

9 Acknowledgments

We thank Ken Litkowski of CL Research for providing the answer patterns for the TREC-10 question set, and Jamie Callan and Jaime Carbonell for comments and helpful discussion. We also thank the anonymous EMNLP reviewers for their comments and suggestions.

References

R. Byrd and Y. Ravin. 1999. Identifying and Extracting Relations in Text. In *Proceedings of NLDB 99.* Klagenfurt, Austria.

C. Clarke, G. Cormack, and T. Lynam. 2001. Exploiting Redundancy in Question Answering. In *Proceedings of SIGIR 2001.* New Orleans, LA.

S. Harabagiu, D. Moldovan, M. Paşca, R. Mihalcea, M. Surdeanu, R. Bunescu, R. Gîrju, V. Rus, and P. Morărescu. 2000. FALCON: Boosting Knowledge for Answer Engines. In *Proceedings of the 9th Text Retrieval Conference (TREC-9)*

L. Hiyakumoto. 2001. Planning and Execution for Open-Domain Question Answering PhD Thesis proposal. Carnegie Mellon University.

H. Hovy, L. Gerber, U. Hermjakob, M. Junk, and C-Y. Lin. 2001 Question Answering in Webclopedia. In *Proceedings of the 9th Text REtrieval Conference (TREC9).*

M.A. Paşca and S.M. Harabagiu. 2001 High Performance Question/Answering. In *Proceedings of SIGIR 2001.* New Orleans, LA.

J.M. Prager, E.W. Brown, A.R. Coden, and D.R. Radev. 2000 Question-Answering by Predictive Annotation. In *Proceedings of SIGIR 2000.* pp. 184–191. Athens, Greece.

J.M. Prager, D. Radev, and K. Czuba. 2001 Answering What-Is Questions by Virtual Annotation. In *Proceedings of HLT'01.*

D.R. Radev, J.M. Prager and V. Samn. 2000 Ranking Suspected Answers to Natural Language Questions using Predictive Annotation. In *Proceedings of ANLP'00.* Seattle, WA.

E. Voorhees. 2001. QA-Track Overview. In *Proceedings of the 10th Text REtrieval Conference (TREC2001).*

Weka. On-line information and implementation available at http://www.cs.waikato.ac.nz/ml/weka/.

I.H. Witten and E. Frank. 1999. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations* Morgan Kaufmann. October 1999.