Learning Question Classifiers[*]

Xin Li   **Dan Roth**

Department of Computer Science
University of Illinois at Urbana-Champaign
{xli1,danr}@uiuc.edu

Abstract

In order to respond correctly to a free form factual question given a large collection of texts, one needs to understand the question to a level that allows determining some of the constraints the question imposes on a possible answer. These constraints may include a semantic classification of the sought after answer and may even suggest using different strategies when looking for and verifying a candidate answer.

This paper presents a machine learning approach to question classification. We learn a hierarchical classifier that is guided by a layered semantic hierarchy of answer types, and eventually classifies questions into fine-grained classes. We show accurate results on a large collection of free-form questions used in TREC 10.

1 Introduction

Open-domain question answering (Lehnert, 1986; Harabagiu et al., 2001; Light et al., 2001) and story comprehension (Hirschman et al., 1999) have become important directions in natural language processing. Question answering is a retrieval task more challenging than common search engine tasks because its purpose is to find an accurate and concise answer to a question rather than a relevant document. The difficulty is more acute in tasks such as story comprehension in which the target text is less likely to overlap with the text in the questions. For this reason, advanced natural language techniques rather than simple key term extraction are needed. One of the important stages in this process is analyzing the question to a degree that allows determining the "type" of the sought after answer. In the TREC competition (Voorhees, 2000), participants are requested to build a system which, given a set of English questions, can automatically extract answers (a short phrase) of no more than 50 bytes from a 5-gigabyte document library. Participants have realized that locating an answer accurately hinges on first filtering out a wide range of candidates (Hovy et al., 2001; Ittycheriah et al., 2001) based on some categorization of answer types.

This work develops a machine learning approach to question classification (QC) (Harabagiu et al., 2001; Hermjakob, 2001). Our goal is to categorize questions into different semantic classes that impose constraints on potential answers, so that they can be utilized in later stages of the question answering process. For example, when considering the question **Q:** *What Canadian city has the largest population?*, the hope is to classify this question as having answer type **city**, implying that only candidate answers that are cities need consideration.

Based on the SNoW learning architecture, we develop a hierarchical classifier that is guided by a layered semantic hierarchy of answer types and is able to classify questions into fine-grained classes. We suggest that it is useful to consider this classification task as a multi-label classification and find that it is possible to achieve good classification results (over 90%) despite the fact that the number of different labels used is fairly large, 50. We observe that local features are not sufficient to support this accuracy, and that inducing semantic features is crucial for good performance.

The paper is organized as follows: Sec. 2 presents the question classification problem; Sec. 3 discusses the learning issues involved in QC and presents our learning approach; Sec. 4 describes our experimental study.

2 Question Classification

We define Question Classification(QC) here to be the task that, given a question, maps it to one of k classes, which provide a semantic constraint on the sought-after answer[1]. The intension is that this

---

[1]We do not address questions like "Do you have a light?", which calls for an action, but rather only factual Wh-questions.

classification, potentially with other constraints on the answer, will be used by a downstream process which selects a correct answer from among several candidates.

A question classification module in a question answering system has two main requirements. First, it provides constraints on the answer types that allow further processing to precisely locate and verify the answer. Second, it provides information that downstream processes may use in determining answer selection strategies that may be answer type specific, rather than uniform. For example, given the question *"Who was the first woman killed in the Vietnam War?"* we do not want to test every noun phrase in a document to see whether it provides an answer. At the very least, we would like to know that the target of this question is a **person**, thereby reducing the space of possible answers significantly. The following examples, taken from the TREC 10 question collection, exhibit several aspects of this point.

Q: *What is a prism?* Identifying that the target of this question is a **definition**, strategies that are specific for *definitions* (e.g., using predefined templates) may be useful. Similarly, in:

Q: *Why is the sun yellow?* Identifying that this question asks for a **reason**, may lead to using a specific strategy for *reasons*.

The above examples indicate that, given that different answer types may be searched using different strategies, a good classification module may help the question answering task. Moreover, determining the specific semantic type of the answer could also be beneficial in locating the answer and verifying it. For example, in the next two questions, knowing that the targets are a **city** or **country** will be more useful than just knowing that they are **locations**.

Q: *What Canadian city has the largest population?*
Q: *Which country gave New York the Statue of Liberty?*

However, confined by the huge amount of manual work needed for constructing a classifier for a complicated taxonomy of questions, most question answering systems can only perform a coarse classification for no more than 20 classes. As a result, existing approaches, as in (Singhal et al., 2000), have adopted a small set of simple answer entity types, which consisted of the classes: *Person, Location, Organization, Date, Quantity, Duration, Linear Measure*. The rules used in the classification were of the following forms:

– If a query starts with *Who* or *Whom*: type **Person**.

– If a query starts with *Where*: type **Location**.
– If a query contains *Which* or *What*, the head noun phrase determines the class, as for What X questions.

While the rules used have large coverage and reasonable accuracy, they are not sufficient to support fine-grained classification. One difficulty in supporting fine-grained classification is the need to extract from the questions finer features that require syntactic and semantic analysis of questions, and possibly, many of them. The approach we adopted is a multi-level learning approach: some of our features rely on finer analysis of the questions that are outcomes of learned classifiers; the QC module then applies learning with these as input features.

2.1 Classification Standard

Earlier works have suggested various standards of classifying questions. Wendy Lehnert's conceptual taxonomy (Lehnert, 1986), for example, proposes about 13 conceptual classes including *causal antecedent, goal orientation, enablement, causal consequent, verification, disjunctive*, and so on. However, in the context of factual questions that are of interest to us here, conceptual categories do not seem to be helpful; instead, our goal is to *semantically* classify questions, as in earlier work on TREC (Singhal et al., 2000; Hovy et al., 2001; Harabagiu et al., 2001; Ittycheriah et al., 2001). The key difference, though, is that we attempt to do that with a significantly finer taxonomy of answer types; the hope is that with the semantic answer types as input, one can easily locate answer candidates, given a reasonably accurate named entity recognizer for documents.

2.2 Question Hierarchy

We define a two-layered taxonomy, which represents a natural semantic classification for typical answers in the TREC task. The hierarchy contains 6 coarse classes (ABBREVIATION, ENTITY, DESCRIPTION, HUMAN, LOCATION and NUMERIC VALUE) and 50 fine classes, Table 1 shows the distribution of these classes in the 500 questions of TREC 10. Each coarse class contains a non-overlapping set of fine classes. The motivation behind adding a level of coarse classes is that of compatibility with previous work's definitions, and comprehensibility. We also hoped that a hierarchical classifier would have a performance advantage over a multi-class classifier; this point, however is not fully supported by our experiments.

Class	#	Class	#
ABBREV.	9	description	7
abb	1	manner	2
exp	8	reason	6
ENTITY	94	**HUMAN**	65
animal	16	group	6
body	2	individual	55
color	10	title	1
creative	0	description	3
currency	6	**LOCATION**	81
dis.med.	2	city	18
event	2	country	3
food	4	mountain	3
instrument	1	other	50
lang	2	state	7
letter	0	**NUMERIC**	113
other	12	code	0
plant	5	count	9
product	4	date	47
religion	0	distance	16
sport	1	money	3
substance	15	order	0
symbol	0	other	12
technique	1	period	8
term	7	percent	3
vehicle	4	speed	6
word	0	temp	5
DESCRIPTION	138	size	0
definition	123	weight	4

Table 1: The distribution of 500 TREC 10 questions over the question hierarchy. Coarse classes (in bold) are followed by their fine class refinements.

2.3 The Ambiguity Problem

One difficulty in the question classification task is that there is no completely clear boundary between classes. Therefore, the classification of a specific question can be quite ambiguous. Consider

1. *What is bipolar disorder?*
2. *What do bats eat?*
3. *What is the PH scale?*

Question 1 could belong to **definition** or **disease_medicine**; Question 2 could belong to **food**, **plant** or **animal**; And Question 3 could be a **numeric value** or a **definition**. It is hard to categorize those questions into one single class and it is likely that mistakes will be introduced in the downstream process if we do so. To avoid this problem, we allow our classifiers to assign multiple class labels for a single question. This method is better than only allowing one label because we can apply all the classes in the later precessing steps without any loss.

3 Learning a Question Classifier

Using machine learning methods for question classification is advantageous over manual methods for several reasons. The construction of a manual classifier for questions is a tedious task that requires the analysis of a large number of questions. Moreover, mapping questions into fine classes requires the use of lexical items (specific words) and therefore an explicit representation of the mapping may

be very large. On the other hand, in our learning approach one can define only a small number of "types" of features, which are then expanded in a data-driven way to a potentially large number of features (Cumby and Roth, 2000), relying on the ability of the learning process to handle it. It is hard to imagine writing explicitly a classifier that depends on thousands or more features. Finally, a learned classifier is more flexible to reconstruct than a manual one because it can be trained on a new taxonomy in a very short time.

One way to exhibit the difficulty in manually constructing a classifier is to consider reformulations of a question:

What tourist attractions are there in Reims?
What are the names of the tourist attractions in Reims?
What do most tourists visit in Reims?
What attracts tourists to Reims?
What is worth seeing in Reims?

All these reformulations target the same answer type **Location**. However, different words and syntactic structures make it difficult for a manual classifier based on a small set of rules to generalize well and map all these to the same answer type. Good learning methods with appropriate features, on the other hand, may not suffer from the fact that the number of potential features (derived from words and syntactic structures) is so large and would generalize and classify these cases correctly.

3.1 A Hierarchical Classifier

Question classification is a multi-class classification. A question can be mapped to one of 50 possible classes (We call the set of all possible class labels for a given question a *confusion set* (Golding and Roth, 1999)). Our learned classifier is based on the SNoW learning architecture (Carlson et al., 1999; Roth, 1998)[2] where, in order to allow the classifier to output more than one class label, we map the classifier's output activation into a conditional probability of the class labels and threshold it.

The question classifier makes use of a sequence of two simple classifiers (Even-Zohar and Roth, 2001), each utilizing the Winnow algorithm within SNoW. The first classifies questions into coarse classes (*Coarse Classifier*) and the second into fine classes (*Fine Classifier*). A feature extractor automatically extracts the same features for each classifier. The second classifier depends on the first in

---

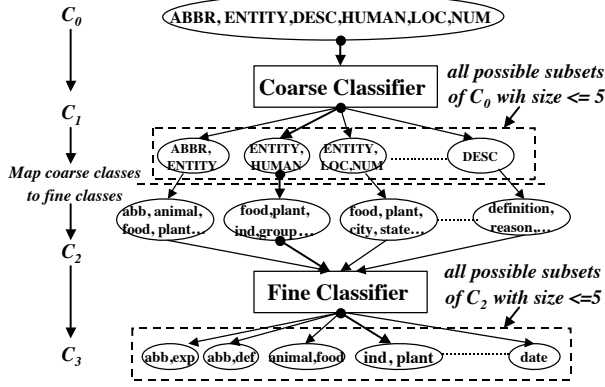[2]Freely available at http://L2R.cs.uiuc.edu/~cogcomp/cc-software.html

Figure 1: The hierarchical classifier

that its candidate labels are generated by expanding the set of retained coarse classes from the first into a set of fine classes; this set is then treated as the confusion set for the second classifier.

Figure 1 shows the basic structure of the hierarchical classifier. During either the training or the testing stage, a question is processed along one path top-down to get classified.

The initial confusion set of any question is $C_0 = \{c_1, c_2, \ldots, c_n\}$, the set of all the coarse classes. The coarse classifier determines a set of preferred labels, $C_1 = Coarse\_Classifier(C_0)$, $C_1 \subseteq C_0$ so that $|C_1| \leq 5$. Then each coarse class label in C_1 is expanded to a fixed set of fine classes determined by the class hierarchy. That is, suppose the coarse class c_i is mapped into the set $c_i = \{f_{i1}, f_{i2}, \ldots, f_{im}\}$ of fine classes, then $C_2 = \bigcup_{c_i \in C_1} c_i$. The fine classifier determines a set of preferred labels, $C_3 = Fine\_Classifier(C_2)$ so that $C_3 \subseteq C_2$ and $|C_3| \leq 5$. C_1 and C_3 are the ultimate outputs from the whole classifier which are used in our evaluation.

3.2 Feature Space

Each question is analyzed and represented as a list of features to be treated as a training or test example for learning. We use several types of features and investigate below their contribution to the QC accuracy.

The primitive feature types extracted for each question include *words*, *pos tags*, *chunks* (non-overlapping phrases) (Abney, 1991), *named entities*, *head chunks* (e.g., the first noun chunk in a sentence) and *semantically related words* (words that often occur with a specific question class).

Over these primitive features (which we call "sensors") we use a set of operators to compose

more complex features, such as conjunctive (n-grams) and relational features, as in (Cumby and Roth, 2000; Roth and Yih, 2001). A simple script that describes the "types" of features used, (e.g., conjunction of two consecutive words and their pos tags) is written and the features themselves are extracted in a data driven way. Only "active" features are listed in our representation so that despite the large number of potential features, the size of each example is small.

Among the 6 primitive feature types, pos tags, chunks and head chunks are syntactic features while named entities and semantically related words are semantic features. Pos tags are extracted using a SNoW-based pos tagger (Even-Zohar and Roth, 2001). Chunks are extracted using a previously learned classifier (Punyakanok and Roth, 2001; Li and Roth, 2001). The named entity classifier is also learned and makes use of the same technology developed for the chunker (Roth et al., 2002). The 'related word' sensors were constructed semi-automatically.

Most question classes have a semantically related word list. Features will be extracted for this class if a word in a question belongs to the list. For example, when "away", which belongs to a list of words semantically related to the class **distance**, occurs in the sentence, the sensor **Rel(distance)** will be active. We note that the features from these sensors are different from those achieved using named entity since they support more general "semantic categorization" and include nouns, verbs, adjectives rather than just named entities.

For the sake of the experimental comparison, we define six feature sets, each of which is an incremental combination of the primitive feature types. That is, Feature set 1 (denoted by *Word*) contains word features; Feature set 2 (*Pos*) contains features composed of words and pos tags and so on; The final feature set, Feature set 6 (*RelWord*) contains all the feature types and is the only one that contains the related words lists. The classifiers will be experimented with different feature sets to test the influence of different features. Overall, there are about 200,000 features in the feature space of *RelWord* due to the generation of complex features over simple feature types. For each question, up to a couple of hundreds of them are active.

3.3 Decision Model

For both the coarse and fine classifiers, the same decision model is used to choose class labels for

a question. Given a confusion set and a question, SNoW outputs a density over the classes derived from the activation of each class. After ranking the classes in the decreasing order of density values, we have the possible class labels $C = \{c_1, c_2, \ldots, c_n\}$, with their densities $P = \{p_1, p_2, \ldots, p_n\}$ (where, $\sum_1^n p_i = 1, 0 \leq p_i \leq 1, 1 \leq i \leq n$). As discussed earlier, for each question we output the first k classes $(1 \leq k \leq 5)$, $c_1, c_2, \ldots c_k$ where k satisfies,

$$k = min(argmin_t(\sum^t p_i \geq T), 5) \qquad (1)$$

T is a threshold value in [0,1]. If we treat p_i as the probability that a question belongs to Class i, the decision model yields a reasonable probabilistic interpretation. We use $T = 0.95$ in the experiments.

4 Experimental Study

We designed two experiments to test the accuracy of our classifier on TREC questions. The first experiment evaluates the contribution of different feature types to the quality of the classification. Our hierarchical classifier is trained and tested using one of the six feature sets defined in Sect. 3.2 (we repeated the experiments on several different training and test sets). In the second experiment, we evaluate the advantage we get from the hierarchical classifier. We construct a multi-class classifier only for fine classes. This flat classifier takes all fine classes as its initial confusion set and classifies a question into fine classes directly. Its parameters and decision model are the same as those of the hierarchical one. By comparing this flat classifier with our hierarchical classifier in classifying fine classes, we hope to know whether the hierarchical classifier has any advantage in performance, in addition to the advantages it might have in downstream processing and comprehensibility.

4.1 Data

Data are collected from four sources: 4,500 English questions published by USC (Hovy et al., 2001), about 500 manually constructed questions for a few rare classes, 894 TREC 8 and TREC 9 questions, and also 500 questions from TREC 10 which serves as our test set[3].

These questions were manually labeled according to our question hierarchy. Although we allow multiple labels for one question in our classifiers, in our labeling, for simplicity, we assigned exactly

---

[3]The annotated data and experimental results are available from `http://L2R.cs.uiuc.edu/~cogcomp/`

one label to each question. Our annotators were requested to choose the most suitable class according to their own understanding. This methodology might cause slight problems in training, when the labels are ambiguous, since some questions are not treated as positive examples for possible classes as they should be. In training, we divide the 5,500 questions from the first three sources randomly into 5 training sets of 1,000, 2,000, 3,000, 4,000 and 5,500 questions. All 500 TREC 10 questions are used as the test set.

4.2 Evaluation

In this paper, we count the number of correctly classified questions by two different precision standards P_1 and $P_{\leq 5}$. Suppose k_i labels are output for the i-th question ($k_i \leq 5$) and are ranked in a decreasing order according to their density values. We define

$$I_{ij} = \left\{ \begin{array}{l} 1, \; if \; the \; correct \; label \; of \; the \; ith \\ \quad question \; is \; output \; in \; rank \; j; \\ 0, \; otherwise. \end{array} \right. \qquad (2)$$

Then, $P_1 = \sum_{i=1}^m I_{i1}/m$ and $P_{\leq 5} = \sum_{i=1}^m \sum_{j=1}^{k_i} I_{ij}/m$ where m is the total number of test examples. P_1 corresponds to the usual definition of precision which allows only one label for each question, while $P_{\leq 5}$ allows multiple labels. $P_{\leq 5}$ reflects the accuracy of our classifier with respect to later stages in a question answering system. As the results below show, although question classes are still ambiguous, few mistakes are introduced by our classifier in this step.

4.3 Experimental Results

Performance of the hierarchical classifier

Table 2 shows the $P_{\leq 5}$ precision of the hierarchical classifier when trained on 5,500 examples and tested on the 500 TREC 10 questions. The results are quite encouraging; question classification is shown to be solved effectively using machine learning techniques. It also shows the contribution of the feature sets we defined. Overall, we get a 98.80% precision for coarse classes with all the features and 95% for the fine classes.

$P_{\leq = 5}$	Word	Pos	Chunk	NE	Head	RelWord
Coarse	92.00	96.60	97.00	97.00	97.80	**98.80**
Fine	86.00	86.60	87.60	88.60	89.40	**95.00**

Table 2: Classification results of the hierarchical classifier on 500 TREC 10 questions. Training is done on 5,500 questions. Columns show the performance for difference feature sets and rows show the precision for coarse and fine classes, resp. All the results are evaluated using $P_{\leq 5}$.

Inspecting the data carefully, we can observe the significant contribution of the features constructed based on semantically related words sensors. It is interesting to observe that this improvement is even more significant for fine classes.

No.	Train	Test	P_1	$P_{\leq=5}$
1	1000	500	83.80	95.60
2	2000	500	84.80	96.40
3	3000	500	91.00	98.00
4	4000	500	90.80	98.00
5	5500	500	91.00	98.80

Table 3: Classification accuracy for coarse classes on different training sets using the feature set RelWord. Results are evaluated using P_1 and $P_{\leq 5}$.

No.	Train	Test	P_1	$P_{\leq=5}$
1	1000	500	71.00	83.80
2	2000	500	77.80	88.20
3	3000	500	79.80	90.60
4	4000	500	80.00	91.20
5	5500	500	84.20	95.00

Table 4: Classification accuracy for fine classes on different training sets using the feature set RelWord. Results are evaluated using P_1 and $P_{\leq 5}$.

Tables 3 and 4 show the P_1 and $P_{\leq 5}$ accuracy of the hierarchical classifier on training sets of different sizes and exhibit the learning curve for this problem.

We note that the average numbers of labels output by the coarse and fine classifiers are 1.54 and 2.05 resp., (using the feature set RelWord and 5,500 training examples), which shows the decision model is accurate as well as efficient.

Comparison of the hierarchical and the flat classifier

The flat classifier consists of one classifier which is almost the same as the fine classifier in the hierarchical case, except that its initial confusion set is the whole set of fine classes. Our original hope was that the hierarchical classifier would have a better performance, given that its fine classifier only needs to deal with a smaller confusion set. However, it turns out that there is a tradeoff between this factor and the inaccuracy, albeit small, of the coarse level prediction. As the results show, there is no performance advantage for using a level of coarse classes, and the semantically appealing coarse classes do not contribute to better performance.

Figure 2 give some more intuition on the flat vs. hierarchical issue. We define the tendency of Class i to be confused with Class j as follows:

$$D_{ij} = Err_{ij} * 2/(N_i + N_j), \qquad (3)$$

where (when using P_1), Err_{ij} is the number of questions in Class i that are misclassified as belong-

P_1	Word	Pos	Chunk	NE	Head	RelWord
h	77.60	78.20	77.40	78.80	78.80	84.20
f	52.40	77.20	77.00	78.40	76.80	84.00
$P_{\leq=5}$	Word	Pos	Chunk	NE	Head	RelWord
h	86.00	86.60	87.60	88.60	89.40	95.00
f	83.20	86.80	86.60	88.40	89.80	95.60

Table 5: Comparing accuracy of the hierarchical (h) and flat (f) classifiers on 500 TREC 10 question; training is done on 5,500 questions. Results are shown for different feature sets using P_1 and $P_{\leq 5}$.
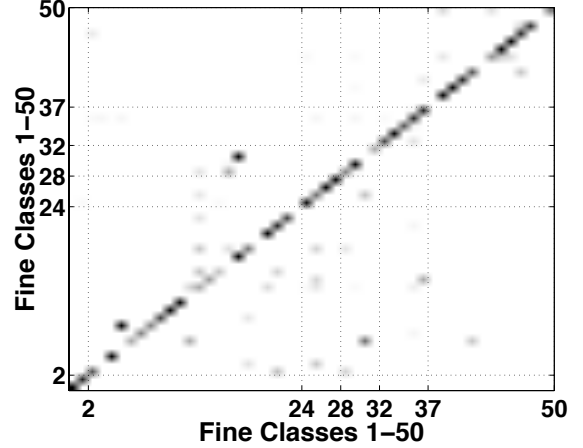

Figure 2: The gray–scale map of the matrix D[n,n]. The color of the small box in position (i,j) denotes D_{ij}. The larger D_{ij} is, the darker the color is. The dotted lines separate the 6 coarse classes.

ing to Class j, and N_i, N_j are the numbers of questions in Class i and j resp.

Figure 2 is a gray-scale map of the matrix D[n,n]. D[n,n] is so sparse that most parts of the graph are blank. We can see that there is no good clustering of fine classes mistakes within a coarse class, which explains intuitively why the hierarchical classifier with an additional level coarse classes does not work much better.

4.4 Discussion and Examples

We have shown that the overall accuracy of our classifier is satisfactory. Indeed, all the reformulation questions that we exemplified in Sec. 3 have been correctly classified. Nevertheless, it is constructive to consider some cases in which the classifier fails. Below are some examples misclassified by the hierarchical classifier.

What French ruler was defeated at the battle of Waterloo?

The correct label is **individual**, but the classifier, failing to relate the word "ruler" to a person, since it was not in any semantic list, outputs **event**.

What is the speed hummingbirds fly ?

The correct label is **speed**, but the classifier outputs

animal. Our feature sensors fail to determine that the focus of the question is 'speed'. This example illustrates the necessity of identifying the question focus by analyzing syntactic structures.

What do you call a professional map drawer ?
The classifier returns **other entities** instead of **equivalent term**. In this case, both classes are acceptable. The ambiguity causes the classifier not to output **equivalent term** as the first choice.

5 Conclusion

This paper presents a machine learning approach to question classification. We developed a hierarchical classifier that is guided by a layered semantic hierarchy of answers types, and used it to classify questions into fine-grained classes. Our experimental results prove that the question classification problem can be solved quite accurately using a learning approach, and exhibit the benefits of features based on semantic analysis.

In future work we plan to investigate further the application of deeper semantic analysis (including better named entity and semantic categorization) to feature extraction, automate the generation of the semantic features and develop a better understanding to some of the learning issues involved in the difference between a flat and a hierarchical classifier.

References

S. P. Abney. 1991. Parsing by chunks. In S. P. Abney R. C. Berwick and C. Tenny, editors, *Principle-based parsing: Computation and Psycholinguistics*, pages 257–278. Kluwer, Dordrecht.

A. Carlson, C. Cumby, J. Rosen, and D. Roth. 1999. The SNoW learning architecture. Technical Report UIUCDCS-R-99-2101, UIUC Computer Science Department, May.

C. Cumby and D. Roth. 2000. Relational representations that facilitate learning. In *Proc. of the International Conference on the Principles of Knowledge Representation and Reasoning*, pages 425–434.

Y. Even-Zohar and D. Roth. 2001. A sequential model for multi class classification. In *EMNLP-2001, the SIGDAT Conference on Empirical Methods in Natural Language Processing*, pages 10–19.

A. R. Golding and D. Roth. 1999. A Winnow based approach to context-sensitive spelling correction. *Machine Learning*, 34(1-3):107–130.

S. Harabagiu, D. Moldovan, M. Pasca, R. Mihalcea, M. Surdeanu, R. Bunescu, R. Girju, V. Rus, and P. Morarescu. 2001. Falcon: Boosting knowledge for answer engines. In *Proceedings of the 9th Text Retrieval Conference, NIST*.

U. Hermjakob. 2001. Parsing and question classification for question answering. In *ACL-2001 Workshop on Open-Domain Question Answering*.

L. Hirschman, M. Light, E. Breck, and J. Burger. 1999. Deep read: A reading comprehension system. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*.

E. Hovy, L. Gerber, U. Hermjakob, C. Lin, and D. Ravichandran. 2001. Toward semantics-based answer pinpointing. In *Proceedings of the DARPA Human Language Technology conference (HLT). San Diego, CA*.

A. Ittycheriah, M. Franz, W-J Zhu, A. Ratnaparkhi, and R.J. Mammone. 2001. IBM's statistical question answering system. In *Proceedings of the 9th Text Retrieval Conference, NIST*.

W. G. Lehnert. 1986. A conceptual theory of question answering. In B. J. Grosz, K. Sparck Jones, and B. L. Webber, editors, *Natural Language Processing*, pages 651–657. Kaufmann, Los Altos, CA.

X. Li and D. Roth. 2001. Exploring evidence for shallow parsing. In *Proc. of the Annual Conference on Computational Natural Language Learning*.

M. Light, G. Mann, E. Riloff, and E. Breck. 2001. Analyses for Elucidating Current Question Answering Technology. *Journal for Natural Language Engineering*. forthcoming.

V. Punyakanok and D. Roth. 2001. The use of classifiers in sequential inference. In *NIPS-13; The 2000 Conference on Advances in Neural Information Processing Systems*, pages 995–1001. MIT Press.

D. Roth and W. Yih. 2001. Relational learning via propositional algorithms: An information extraction case study. In *Proc. of the International Joint Conference on Artificial Intelligence*, pages 1257–1263.

D. Roth, G. Kao, X. Li, R. Nagarajan, V. Punyakanok, N. Rizzolo, W. Yih, C. O. Alm, and L. G. Moran. 2002. Learning components for a question answering system. In *TREC-2001*.

D. Roth. 1998. Learning to resolve natural language ambiguities: A unified approach. In *Proc. of the American Association of Artificial Intelligence*, pages 806–813.

A. Singhal, S. Abney, M. Bacchiani, M. Collins, D. Hindle, and F. Pereira. 2000. AT&T at TREC-8. In *Proceedings of the 8th Text Retrieval Conference, NIST*.

E. Voorhees. 2000. Overview of the TREC-9 question answering track. In *The Ninth Text Retrieval Conference (TREC-9)*, pages 71–80. NIST SP 500-249.