A Probabilistic Framework for Answer Selection in Question Answering

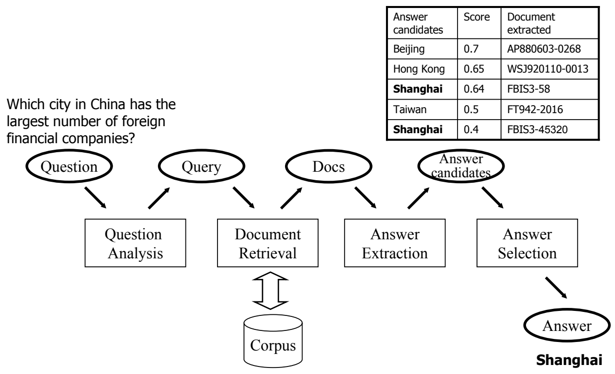
Jeongwoo Ko[1], **Luo Si**[2], **Eric Nyberg**[1]
[1]Language Technologies Institute, Carnegie Mellon, Pittsburgh, PA 15213
[2]Department of Computer Science, Purdue University, West Lafayette, IN 47907
`jko@cs.cmu.edu, lsi@cs.purdue.edu, ehn@cs.cmu.edu`

Abstract

This paper describes a probabilistic answer selection framework for question answering. In contrast with previous work using individual resources such as ontologies and the Web to validate answer candidates, our work focuses on developing a unified framework that not only uses multiple resources for validating answer candidates, but also considers evidence of similarity among answer candidates in order to boost the ranking of the correct answer. This framework has been used to select answers from candidates generated by four different answer extraction methods. An extensive set of empirical results based on TREC factoid questions demonstrates the effectiveness of the unified framework.

1 Introduction

Question answering aims at finding exact answers to a user's natural language question from a large collection of documents. Most QA systems combine information retrieval with extraction techniques to identify a set of likely candidates and then utilize some selection strategy to generate the final answers (Prager et al., 2000; Clarke et al., 2001; Harabagiu et al., 2001). Since answer extractors may be based on imprecise empirical methods, the selection process can be very challenging, as it often entails identifying correct answer(s) amongst many incorrect ones.



Answer candidates	Score	Document extracted
Beijing	0.7	AP880603-0268
Hong Kong	0.65	WSJ920110-0013
Shanghai	0.64	FBIS3-58
Taiwan	0.5	FT942-2016
Shanghai	0.4	FBIS3-45320

Figure 1: A traditional QA pipeline architecture

Figure 1 shows a traditional QA architecture with an example question. Given the question "*Which city in China has the largest number of foreign financial companies?*", the answer extraction component produces a ranked list of five answer candidates. Due to imprecision in answer extraction, an incorrect answer ("Beijing") was ranked at the top position. The correct answer ("Shanghai") was extracted from two documents with different confidence scores and ranked at the third and the fifth positions. In order to select "Shanghai" as the final answer, we need to address two issues:

- *Answer Validation*. How do we identify correct answer(s) amongst incorrect ones? Validating an answer may involve searching for facts in a knowledge base, e.g. `IS-A(Shanghai, city)`, `IS-IN(Shanghai, China)`.
- *Answer Similarity*. How do we exploit evidence of similarity among answer candidates?

Proceedings of NAACL HLT 2007, pages 524–531,
Rochester, NY, April 2007. ©2007 Association for Computational Linguistics

For example, when there are redundant answers ("Shanghai", as above) or several answers which represent a single instance (e.g. "Clinton, Bill" and "William Jefferson Clinton") in the candidate list, how much should we boost the answer candidate scores?

To address the first issue, several answer selection approaches have used semantic resources. One of the most common approaches relies on WordNet, CYC and gazetteers for answer validation or answer reranking; answer candidates are pruned or discounted if they are not found within a resource's hierarchy corresponding to the expected answer type (Xu et al., 2003; Moldovan et al., 2003; Prager et al., 2004). In addition, the Web has been used for answer reranking by exploiting search engine results produced by queries containing the answer candidate and question keywords (Magnini et al., 2002), and Wikipedia's structured information has been used for answer type checking (Buscaldi and Rosso, 2006).

To use more than one resource for answer type checking of location questions, Schlobach et al. (2004) combined WordNet with geographical databases. However, in their experiments the combination actually hurt performance because of the increased semantic ambiguity that accompanies broader coverage of location names. This demonstrates that the method used to combine potential answers may matter as much as the choice of resources.

To address the second issue we must determine how to detect and exploit answer similarity. As answer candidates are extracted from different documents, they may contain identical, similar or complementary text snippets. For example, the United States may be represented by the strings "U.S.", "United States" or "USA" in different documents. It is important to detect this type of similarity and exploit it to boost answer confidence, especially for list questions that require a set of unique answers. One approach is to incorporate answer clustering (Kwok et al., 2001; Nyberg et al., 2003; Jijkoun et al., 2006). For example, we might merge "April 1912" and "14 Apr 1912" into a cluster and then choose one answer as the cluster head. However, clustering raises new issues: how to choose the cluster head and how to calculate the scores of the clustered answers.

Although many QA systems individually address these issues in answer selection, there has been little research on generating a generalized probabilistic framework that allows any validation and similarity features to be easily incorporated.

In this paper we describe a probabilistic answer selection framework to address the two issues. The framework uses logistic regression to estimate the probability that an answer candidate is correct given multiple answer validation features and answer similarity features. Experimental results on TREC factoid questions (Voorhees, 2004) show that our framework significantly improved answer selection performance for four different extraction techniques, when compared to default selection using the individual candidate scores produced by each extractor.

This paper is organized as follows: Section 2 describes our answer selection framework and Section 3 lists the features that generate similarity and validity scores for factoid questions. In Section 4, we describe the experimental methodology and the results. Section 5 describes how we intend to extend our framework to handle complex questions. Finally Section 6 concludes with suggestions for future research.

2 Method

Answer validation is based on an estimate of the probability $P(correct(A_i)|A_i, Q)$, where Q is a question and A_i is an answer candidate to the question. Answer similarity is is based on an estimate of the probability $P(correct(A_i)|A_i, A_j)$, where A$_j$ is similar to A$_i$. Since both probabilities influence answer selection performance, it is important to combine them in a unified framework and estimate the probability of an answer candidate as: $P(correct(A_i)|Q, A_1, ..., A_n)$.

In this paper, we propose a probabilistic framework that directly estimates $P(correct(A_i)|Q, A_1, ..., A_n)$ using multiple answer validation features and answer similarity features. The framework was implemented with logistic regression, which is a statistical machine learning technique used to predict the probability of a binary variable from input variables. Logistic

$$\begin{aligned}
&P(correct(A_i)|Q, A_1, ..., A_n) \quad (1)\\
&\quad \approx P(correct(A_i)|val_1(A_i), ..., val_{K1}(A_i), sim_1(A_i), ..., sim_{K2}(A_i))\\
&\quad = \frac{exp(\alpha_0 + \sum_{k=1}^{K1} \beta_k val_k(A_i) + \sum_{k=1}^{K2} \lambda_k sim_k(A_i))}{1 + exp(\alpha_0 + \sum_{k=1}^{K1} \beta_k val_k(A_i) + \sum_{k=1}^{K2} \lambda_k sim_k(A_i))}\\
&\quad where, sim_k(A_i) = \sum_{j=1(j\neq i)}^{N} sim'_k(A_i, A_j).
\end{aligned}$$

$$\vec{\alpha}, \vec{\beta}, \vec{\lambda} = \underset{\vec{\alpha},\vec{\beta},\vec{\lambda}}{\operatorname{argmax}} \sum_{j=1}^{R} \sum_{i=1}^{N_j} logP(correct(A_i)|val_1(A_i), ..., val_{K1}(A_i), sim_1(A_i), ..., sim_{K2}(A_i)) \quad (2)$$

regression has been successfully employed in many applications including multilingual document merging (Si and Callan, 2005). In our previous work (Ko et al., 2006), we showed that logistic regression performed well in merging three resources to validate answers to location and proper name questions. We extended this approach to combine multiple similarity features with multiple answer validation features. The extended framework estimates the probability that an answer candidate is correct given the degree of answer correctness and the amount of supporting evidence provided in a set of answer candidates (Equation 1).

In Equation 1, each val$_k$(A$_i$) is a feature function used to produce an answer validity score for an answer candidate A$_i$. Each sim$'_k(A_i, A_j)$ is a similarity function used to calculate an answer similarity between A$_i$ and A$_j$. K1 and K2 are the number of answer validation and answer similarity features, respectively. N is the number of answer candidates.

To incorporate multiple similarity features, each sim$_k(A_i)$ is obtained from an individual similarity metric. For example, if Levenshtein distance is used as one similarity metric, sim$_k(A_i)$ is calculated by summing N-1 Levenshtein distances between one answer candidate and all other candidates. As some string similarity metrics (e.g. Levenshtein distance) produce a number between 0 and 1 (where 1 means two strings are identical and 0 means they are different), similarity scores less than some threshold value are ignored.

The parameters α, β, λ were estimated from training data by maximizing the log likelihood as shown in Equation 2, where R is the number of training questions and N$_j$ is the number of answer candidates for each question Q$_j$. For parameter estimation, we used the Quasi-Newton algorithm (Minka, 2003).

To select correct answers, the initial answer candidate set is reranked according to the estimated probability of each candidate. For factoid questions, the top answer is selected as the final answer to the question. As logistic regression can be used for binary classification with a default threshold of 0.5, we can also use the framework to classify incorrect answers: if the probability of an answer candidate is lower than 0.5, it is considered to be a wrong answer and is filtered out of the answer list. This is useful in deciding whether or not a valid answer exists in the corpus, an important aspect of the TREC QA evaluation (Voorhees, 2004).

3 Feature Representation

This section details the features used to generate answer validity scores and answer similarity scores for our answer selection framework.

3.1 Answer Validation Features

Each answer validation feature produces a validity score which predicts whether or not an answer candidate is a correct answer for the question. This task can be done by exploiting external QA resources such as the Web, databases, and ontologies. For factoid questions, we used gazetteers and WordNet in a knowledge-based approach; we also used Wikipedia and Google in a data-driven approach.

3.1.1 Knowledge-based Features

In order to generate answer validity scores using gazetteers and WordNet, we reused the algorithms described in our previous work (Ko et al., 2006).

Gazetteers: Gazetteers provide geographic information, which allows us to identify strings as instances of countries, their cities, continents, capitals, etc. For answer selection, we used three gazetteer resources: the Tipster Gazetteer, the CIA World Factbook (https://www.cia.gov/cia/publications/factbook/index.html) and information about the US states provided by 50states.com (http://www.50states.com). These resources were used to assign an answer validity score between -1 and 1 to each candidate (Figure 2). A score of 0 means the gazetteers did not contribute to the answer selection process for that candidate. For some numeric questions, range checking was added to validate numeric questions similarly to Prager et al. (2004). For example, given the question *"How many people live in Chile?"*, if an answer candidate is within $\pm$ 10% of the population stated in the CIA World Factbook, it receives a score of 1.0. If it is in the range of 20%, its score is 0.5. If it significantly differs by more than 20%, it receives a score of -1.0. The threshold may vary based on when the document was written and when the census was taken[1].

WordNet: The WordNet lexical database includes English words organized in synonym sets, called *synsets* (Fellbaum, 1998). We used WordNet in order to produce an answer validity score between -1 and 1, following the algorithm in Figure 3. A score of 0 means that WordNet does not contribute to the answer selection process for a candidate.

[1]The ranges used here were found to work effectively, but were not explicitly validated or tuned.

1) If the answer candidate directly matches the gazetteer answer for the question, its gazetteer score is **1.0**. (e.g. Given the question "*What continent is Togo on?*", the candidate "*Africa*" receives a score of 1.0.)
2) If the answer candidate occurs in the gazetteer within the subcategory of the expected answer type, its score is **0.5**. (e.g., Given the question "*Which city in China has the largest number of foreign financial companies?*", the candidates "*Shanghai*" and "Boston" receive a score of 0.5 because they are both cities.)
3) If the answer candidate is not the correct semantic type, its score is **-1**. (e.g., Given the question "*Which city in China has the largest number of foreign financial companies?*", the candidate "*Taiwan*" receives a score of -1 because it is not a city.)
4) Otherwise, the score is **0.0**.

Figure 2: Validity scoring with gazetteers.

1) If the answer candidate directly matches WordNet, its WordNet score is **1.0**. (e.g. Given the question "*What is the capital of Uruguay?*", the candidate "*Montevideo*" receives a score of 1.0.)
2) If the answer candidate's hypernyms include a subcategory of the expected answer type, its score is **0.5**. (e.g., Given the question "*Who wrote the book 'Song of Solomon'?*", the candidate "*Mark Twain*" receives a score of 0.5 because its hypernyms include "*writer*".)
3) If the answer candidate is not the correct semantic type, this candidate receives a score of **-1**. (e.g., Given the question "*What state is Niagara Falls located in?*", the candidate "*Toronto*" gets a score of -1 because it is not a state.)
4) Otherwise, the score is **0.0**.

Figure 3: Validity scoring with WordNet.

3.1.2 Data-driven Features

Wikipedia and Google were used in a data-driven approach to generate answer validity scores.

Wikipedia: Wikipedia (http://www.wikipedia.org) is a multilingual free on-line encyclopedia. Figure 4 shows the algorithm used to generate an answer validity score from Wikipedia. If there is a Wikipedia document whose title matches an answer candidate, the document is analyzed to obtain the term frequency (tf) and the inverse term

```
For each answer candidate $A_i$,
  1. Initialize the Wikipedia score: $ws(A_i) = 0$
  2. Search for a Wikipedia document whose title is $A_i$
  3. If a document is found, calculate tf.idf score of $A_i$ in the
     retrieved Wikipedia document
       $ws(A_i) = (1+\log(tf)) \times (1+\log(idf))$
  4. If not, for each question keyword $K_j$ ,
    4.1. Search for a Wikipedia document that includes $K_j$
    4.2. If a document is found, calculate tf.idf score of $A_i$
       $ws(A_i) \mathrel{+}= (1+\log(tf)) \times (1+\log(idf))$
```

Figure 4: Validity scoring with Wikipedia

```
For each answer candidate $A_i$,
  1. Initialize the Google score: $gs(A_i) = 0$
  2. For each snippet $s$:
    2.1. Initialize the snippet co-occurrence score: $cs(s) = 1$
    2.2. For each question keyword $k$ in $s$:
      2.2.1 Compute distance $d$, the minimum number of
            words between $k$ and the answer candidate
      2.2.2 Update the snippet co-occurrence score:
            $cs(s) = cs(s) \times 2^{(1+d)^{-1}}$
    2.3. $gs(A_i) = gs(A_i) + cs(s)$
  3. Normalize the Google score (dividing it by a constant $C$ )
```

Figure 5: Validity scoring with Google

frequency (idf) of the candidate, from which a tf.idf score is calculated. When there is no matched document, each question keyword is also processed as a back-off strategy, and the answer validity score is calculated by summing the tf.idf scores. To calculate word frequency, the TREC Web Corpus (http://ir.dcs.gla.ac.uk/test_collections/wt10g.html) was used as a large background corpus.

Google: Following Magnini et al. (2002), we used Google to generate a numeric score. A query consisting of an answer candidate and question keywords was sent to the Google search engine. To calculate a score, the top 10 text snippets returned by Google were then analyzed using the algorithm in Figure 5.

3.2 Answer Similarity Features

We calculate the similarity between two answer candidates using multiple string distance metrics and a list of synonyms.

3.2.1 String Distance Metrics

There are several different string distance metrics to calculate the similarity of short strings. We used five popular string distance metrics: Levenshtein, Jaccard, Jaro, Jaro-Winkler, and Cosine similarity.

3.2.2 Synonyms

Synonyms can be used as another metric to calculate answer similarity. We defined a binary similarity score for synonyms.

$$sim(A_i, A_j) = \begin{cases} 1, & if\ A_i\ is\ a\ synonym\ of\ A_j \\ 0, & otherwise \end{cases}$$

To get a list of synonyms, we used three knowledge bases: WordNet, Wikipedia and the CIA World Factbook. WordNet includes synonyms for English words. Wikipedia redirection is used to obtain another set of synonyms. For example, “Calif.” is redirected to “California” in Wikipedia, and “William Jefferson Clinton” is redirected to “Bill Clinton”.

The CIA World Factbook includes five different names for a country: conventional long form, conventional short form, local long form, local short form and former name. For example, the conventional long form of Egypt is “Arab Republic of Egypt”, the conventional short form is “Egypt”, the local short form is “Misr”, the local long form is “Jumhuriyat Misr al-Arabiyah” and the former name is “United Arab Republic (with Syria)”. All are considered to be synonyms of “Egypt”.

In addition, manually generated rules are used to obtain synonyms for different types of answer candidates (Nyberg et al., 2003):

- Dates are converted into the ISO 8601 date format (YYYY-MM-DD) (e.g., “April 12 1914” and “12th Apr. 1914” are converted into “1914-04-12” and considered as synonyms).

- Temporal expressions are converted into the HH:MM:SS format (e.g., “six thirty five p.m.” and “6:35 pm” are converted into “18:35:xx” and considered as synonyms).

- Numeric expression are converted into scientific notation (e.g, “one million” and “1,000,000” are converted into “1e+06” and considered as synonyms).

- Representative entities are converted into the represented entity when the expected answer type is `COUNTRY` (e.g., "the Egyptian government" is changed to "Egypt" and "Clinton administration" is changed to "U.S.").

4 Experiment

This section describes the experiments we used to evaluate our answer selection framework. The JAVELIN QA system (Nyberg et al., 2006) was used as a testbed for the evaluation.

4.1 Experimental Setup

A total of 1760 factoid questions from the TREC8-12 QA evaluations served as a dataset, with 5-fold cross validation.

To better understand how the performance of our framework varies for different extraction techniques, we tested it with four JAVELIN answer extraction modules: FST, LIGHTv1, LIGHTv2 and SVM (Nyberg et al., 2006). FST is an answer extractor based on finite state transducers that incorporate a set of extraction patterns (both manually-created and generalized patterns). LIGHTv1 is an extractor that selects answer candidates using a non-linear distance heuristic between the keywords and an answer candidate. LIGHTv2 is another extractor based on a different distance heuristic, originally developed as part of a multilingual QA system. SVM is an extractor that uses Support Vector Machines to discriminate between correct and incorrect answers.

Answer selection performance was measured by average accuracy: the number of correct top answers divided by the number of questions where at least one correct answer exists in the candidate list provided by an extractor. The baseline was calculated with the answer candidate scores provided by each individual extractor; the answer with the best extractor score was chosen, and no validation or similarity processing was performed. For Wikipedia, we used a version downloaded in Nov. 2005, which contained 1,811,554 articles.

4.2 Results and Analysis

We first analyzed the average accuracy when using individual validation features. Figure 6 shows the effect of the individual answer validation features on different extraction outputs. The combination of all features significantly improved the performance when compared to answer selection using a single feature. Comparing the data-driven features with the knowledge-based features, the data-driven features (such as Wikipedia and Google) increased performance more than the knowledge-based features (such as gazetteers and WordNet); our intuition is that the knowledge-based features covered fewer questions. The biggest improvement was found with candidates produced by the SVM extractor: a 242% improvement over the baseline. It was mostly because SVM tended to produce several answer candidates with the same or very similar confidence scores, but our framework could select the correct answer among many incorrect ones by exploiting answer validation features.

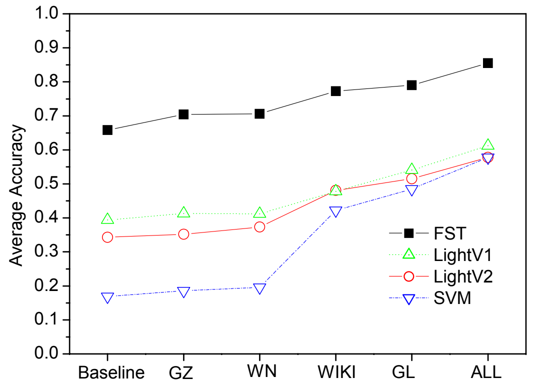


Figure 6: Average accuracy of individual answer validation features (GZ: gazetteers, WN: WordNet, WIKI: Wikipedia, GL: Google, ALL: combination of all features).

Table 1 shows the effect of individual similarity features on different extractors when using 0.3 and 0.5 as a similarity threshold, respectively. When comparing five different string similarity features (Levenshtein, Jaro, Jaro-Winkler, Jaccard and Cosine similarity), Levenshtein and Jaccard tended to perform better than the others. When comparing synonym features with string similarity features, synonyms performed slightly better.

We also analyzed answer selection performance when combining all six similarity features ("All" in Table 1). Combining all similarity features did not improve the performance except for the FST extractor, because including five string similarity features

Similarity feature	FST		LIGHTv1		LIGHTv2		SVM	
	0.3	0.5	0.3	0.5	0.3	0.5	0.3	0.5
Levenshtein	0.728	0.728	**0.471**	0.455	0.399	0.400	0.381	0.383
Jaro	0.708	0.705	0.422	0.440	0.373	0.378	0.274	0.282
Jaro-Winkler	0.701	0.705	0.426	0.442	0.374	0.379	0.277	0.275
Jaccard	0.738	0.738	0.438	0.448	**0.452**	0.448	0.382	0.390
Cosine	0.738	0.738	0.436	0.435	0.418	0.422	0.380	0.378
Synonyms	**0.745**	**0.745**	0.458	0.458	0.442	0.442	**0.412**	**0.412**
Lev+Syn	0.748	**0.751**	0.460	**0.466**	0.445	**0.448**	**0.420**	0.412
Jac+Syn	0.742	0.742	0.456	0.465	0.440	0.445	0.396	0.396
All	0.755	0.755	0.405	0.425	0.435	0.431	0.303	0.302

Table 1: Average accuracy using individual similarity features under different thresholds: 0.3 and 0.5 ("Lev+Syn": the combination of Levenshtein with synonyms, "Jac+Syn": the combination of Jaccard and synonyms, "All": the combination of all similarity metrics)

	Baseline	Sim	Val	All
FST	0.658	0.751	0.855	0.877
LIGHTv1	0.394	0.466	0.612	0.628
LIGHTv2	0.343	0.448	0.578	0.582
SVM	0.169	0.420	0.578	0.586

Table 2: Average accuracy of individual features (Sim: merging similarity features, Val: merging validation features, ALL: combination of all features).

provided too much redundancy to the logistic regression. We also compared the combination of Levenshtein with synonyms and the combination of Jaccard with synonyms, and then chose Levenshtein and synonyms as the two best similarity features in our framework.

We also analyzed the degree to which the average accuracy was affected by answer similarity and validation features. Table 2 compares the average accuracy using the baseline, the answer similarity features, the answer validation features and all feature combinations. As can be seen, the similarity features significantly improved performance, so we can conclude that exploiting answer similarity improves answer selection performance. The validation features also significantly improved the performance.

When combining both sets of features together, the answer selection performance increased for all four extractors: an average of 102% over the baseline, 30% over the similarity features and 1.82% over the validation features. Adding the similarity features to the validation features generated small but consistent improvement in all configurations. We expect more performance gain from similarity features when merging similar answers returned from all four extractors.

5 Extensions for Complex Questions

Although we conducted our experiments on factoid questions, our framework can be easily extended to handle complex questions, which require longer answers representing facts or relations (e.g., *"What is the relationship between Alan Greenspan and Robert Rubin?"*). As answer candidates are long text snippets, different features should be used for answer selection. Possible validation features include question keyword inclusion and predicate structure match (Nyberg et al., 2005). For example, given the question *"Did Egypt sell Scud missiles to Syria?"*, the key predicate from the question is *Sell(Egypt, Syria, Scud missile)*. If there is a sentence which contains the predicate structure *Buy(Syria, Scud missile, Egypt)*, we can calculate the predicate structure distance and use it as a validation feature. For answer similarity, we intend to explore novelty detection approaches evaluated in Allan et al. (2003).

6 Conclusion

In this paper, we described our answer selection framework for estimating the probability that an answer candidate is correct given multiple answer validation and similarity features. We conducted a series of experiments to evaluate the performance of the framework and analyzed the effect of individual validation and similarity features. Empirical results on TREC questions show that our framework improved answer selection performance in the JAVELIN QA system by an average of 102% over the baseline, 30% over the similarity features alone and 1.82% over the validation features alone.

We plan to improve our framework by adding regularization and selecting the final answers among candidates returned from all extractors. As our current framework is based on the assumption that each answer is independent, we are building another probabilistic framework which does not require any independence assumption, and uses an undirected graphical model to estimate the joint probability of all answer candidates.

7 Acknowledgments

This work was supported in part by ARDA/DTO Advanced Question Answering for Intelligence (AQUAINT) program award number NBCHC040164.

References

J. Allan, C. Wade, and A. Bolivar. 2003. Retrieval and novelty detection at the sentence level. In *Proceedings of SIGIR*.

D. Buscaldi and P. Rosso. 2006. Mining Knowledge from Wikipedia for the Question Answering task. In *Proceedings of the International Conference on Language Resources and Evaluation*.

C. Clarke, G. Cormack, and T. Lynam. 2001. Exploiting redundancy in question answering. In *Proceedings of SIGIR*.

C. Fellbaum. 1998. *WordNet: An Electronic Lexical Database*. MIT Press.

S. Harabagiu, D. Moldovan, M. Pasca, R. Mihalcea, M. Surdeanu, R. Bunescu, R. Grju, V. Rus, and P. Morarescu. 2001. FALCON: Boosting knowledge for answer engines. In *Proceedings of TREC*.

V. Jijkoun, J. van Rantwijk, D. Ahn, E. Tjong Kim Sang, and M. de Rijke. 2006. The University of Amsterdam at CLEF@QA 2006. In *Working Notes CLEF*.

J. Ko, L. Hiyakumoto, and E. Nyberg. 2006. Exploiting semantic resources for answer selection. In *Proceedings of the International Conference on Language Resources and Evaluation*.

C. Kwok, O. Etzioni, and D. S. Weld. 2001. Scaling question answering to the web. In *Proceedings of WWW10 Conference*.

B. Magnini, M. Negri, R. Pervete, and H. Tanev. 2002. Comparing statistical and content-based techniques for answer validation on the web. In *Proceedings of the VIII Convegno AI*IA*.

T. Minka. 2003. A Comparison of Numerical Optimizers for Logistic Regression. Unpublished draft.

D. Moldovan, D. Clark, S. Harabagiu, and S. Maiorano. 2003. Cogex: A logic prover for question answering. In *Proceedings of HLT-NAACL*.

E. Nyberg, T. Mitamura, J. Carbonell, J. Callan, K. Collins-Thompson, K. Czuba, M. Duggan, L. Hiyakumoto, N. Hu, Y. Huang, J. Ko, L. Lita, S. Murtagh, V. Pedro, and D. Svoboda. 2003. The JAVELIN Question-Answering System at TREC 2002. In *Proceedings of the Text REtrieval Conference*.

E. Nyberg, T. Mitamura, R. Frederking, M. Bilotti, K. Hannan, L. Hiyakumoto, J. Ko, F. Lin, V. Pedro, and A. Schlaikjer. 2006. JAVELIN I and II Systems at TREC 2005. In *Proceedings of TREC*.

E. Nyberg, T. Mitamura, R. Frederking, V. Pedro, M. Bilotti, A. Schlaikjer, and K. Hannan. 2005. Extending the javelin qa system with domain semantics. In *Proceedings of AAAI-05 Workshop on Question Answering in Restricted Domains*.

J. Prager, E. Brown, A. Coden, and D. Radev. 2000. Question answering by predictive annotation. In *Proceedings of SIGIR*.

J. Prager, J. Chu-Carroll, K. Czuba, C. Welty, A. Ittycheriah, and R. Mahindru. 2004 IBM's Piquant in Trec2003. In *Proceedings of TREC*.

S. Schlobach, M. Olsthoorn, and M. de Rijke. 2004. Type checking in open-domain question answering. In *Proceedings of European Conference on Artificial Intelligence*.

L. Si and J. Callan. 2005 CLEF2005: Multilingual retrieval by combining multiple multilingual ranked lists. In *Proceedings of Cross-Language Evaluation Forum*.

E. Voorhees. 2004. Overview of the TREC 2003 question answering track. In *Proceedings of TREC*.

J. Xu, A. Licuanan, J. May, S. Miller, and R. Weischedel. 2003. TREC 2002 QA at BBN: Answer Selection and Confidence Estimation. In *Proceedings of TREC*.