Automatic Set Expansion for List Question Answering

Richard C. Wang, Nico Schlaefer, William W. Cohen, and Eric Nyberg
Language Technologies Institute
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh PA 15213
{rcwang,nico,wcohen,ehn}@cs.cmu.edu

Abstract

This paper explores the use of set expansion (SE) to improve question answering (QA) when the expected answer is a list of entities belonging to a certain class. Given a small set of seeds, SE algorithms mine textual resources to produce an extended list including additional members of the class represented by the seeds. We explore the hypothesis that a noise-resistant SE algorithm can be used to extend candidate answers produced by a QA system and generate a new list of answers that is better than the original list produced by the QA system. We further introduce a hybrid approach which combines the original answers from the QA system with the output from the SE algorithm. Experimental results for several state-of-the-art QA systems show that the hybrid system performs better than the QA systems alone when tested on list question data from past TREC evaluations.

1 Introduction

Question answering (QA) systems are designed to retrieve precise answers to questions posed in natural language. A *list question* expects a list as its answer, e.g. *Name the coffee-producing countries in South America*. The ability to answer list questions has been tested as part of the yearly TREC QA evaluation (Dang et al., 2006; Dang et al., 2007). This paper focuses on the use of *set expansion* to improve list question answering. A set expansion (SE) algorithm receives as input a few members of a class or set, and mines various textual resources (e.g. web pages) to produce an extended list including additional members of the class or set that are not in the input. A well-known online SE system is Google Sets[1]. This system is publicly accessible, but since it is a proprietary system that might be changed at any time, its results cannot be replicated reliably. We explore the hypothesis that a SE algorithm, when carefully designed to handle noisy inputs, can be applied to the output from a QA system to produce an overall list of answers for a given question that is better than the answers produced by the QA system itself. We propose to exploit large, redundant sources of structured and/or semi-structured data and use linguistic analysis to seed a shallow analysis of these sources. This is a hard problem since the linguistic evidence used as seeds is noisy. More precisely, we combine the QA system Ephyra (Schlaefer et al., 2007) with the SE system SEAL (Wang and Cohen, 2007) to create a hybrid approach that performs better than either system by itself when tested on data from the TREC 13-15 evaluations. In addition, we apply our SE algorithm to answers generated by the five QA systems that performed the best on the list questions in the TREC 15 evaluation and report improvements in F_1 scores for four of these systems.

Section 2 of the paper gives an overview of the QA and SE systems used for our experiments. Section 3 describes how the SE system was adapted to deal with noisy seeds produced by QA systems, and Section 4 presents the details of the experimental design. Experimental results are discussed in Section 5, and the paper concludes in Section 6 with a discussion of planned future work.

[1] http://labs.google.com/sets

2 System Overview

2.1 Ephyra Question Answering System

Ephyra (Schlaefer et al., 2006; Schlaefer et al., 2007) is a QA system that has been evaluated in the TREC QA track (Dang et al., 2006; Dang et al., 2007). The system combines three answer extraction techniques for factoid and list questions: (1) an answer type classification approach; (2) a syntactic pattern learning and matching approach; and (3) a semantic extractor that uses a semantic role labeling system. The answer type based extractor classifies questions by their answer types and extracts candidates of the expected types. The Ephyra pattern matching approach learns textual patterns that relate question key terms to possible answers and applies these patterns to candidate sentences to extract factoid answers. The semantic approach generates a semantic representation of the question that is based on predicate-argument structures and extracts answer candidates from similar structures in the corpus. The source code of the answer extractors is included in OpenEphyra, an open source release of the system.[2]

The answer candidates from these extractors are combined and ranked by a statistical answer selection framework (Ko et al., 2007), which estimates the probability of an answer based on a number of answer validation and similarity features. Validation features use resources such as gazetteers and Wikipedia to verify an answer, whereas similarity features measure the syntactic and semantic similarity to other candidates, e.g. using string distance measures and WordNet relations.

2.2 Set Expander for Any Language (SEAL)

Set expansion (SE) refers to expanding a given partial set of objects into a more complete set. SEAL[3] (Wang and Cohen, 2007) is a SE system which accepts input elements (seeds) of some target set S_t and automatically finds other probable elements of S_t in semi-structured documents such as web pages. SEAL also works on unstructured text, but its extraction mechanism benefits from structuring elements such as HTML tags. The algorithm is independent of the human language from which the seeds are taken, and also independent of the markup language used to annotate the documents. Examples of SEAL's input and output are shown in Figure 1.

	English	Chinese	Japanese
input seed	Amazing Race	豬血糕	ドラえもん
	Survivor	臭豆腐	ハローキティ
output	Big Brother	蘿蔔糕	アンパンマン
	The Mole	蚵仔煎	シナモロール
	The Apprentice	肉圓	ウルトラマン
	Project Runway	滷肉飯	スヌーピー
	The Bachelor	春捲	くまのプーさん

Figure 1: Examples of SEAL's input and output. English entities are reality TV shows, Chinese entities are popular Taiwanese food, and Japanese entities are famous cartoon characters.

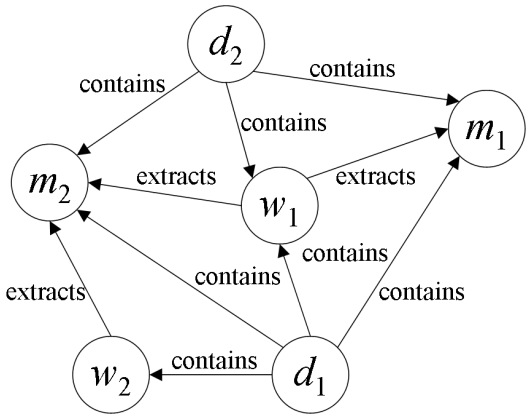

Figure 2: An example graph constructed by SEAL. Every edge from node x to y actually has an inverse relation edge from node y to x that is not shown here (e.g. m_1 *is extracted by* w_1).

In more detail, SEAL comprises three major components: the Fetcher, the Extractor, and the Ranker. The *Fetcher* focuses on retrieving web pages. The URLs of the web pages come from top results retrieved from Google and Yahoo! using the concatenation of all seeds as the query. The *Extractor* automatically constructs page-specific extraction rules, or *wrappers*, for each page that contains the seeds. Every wrapper is defined by two character strings, which specify the left-context and right-context necessary for an entity to be extracted from a page. These strings are chosen to be maximally-long contexts that bracket at least one occurrence of every seed string on a page. Most of the wrappers con-

[2] http://www.ephyra.info/
[3] http://rcwang.com/seal

tain HTML tags, which illustrates the importance of structuring information in the source documents. All entity mentions bracketed by these contextual strings derived from a particular page are extracted from the same page. Finally, the *Ranker* builds a graph, and then ranks the extracted mentions globally based on the weights computed by performing a random graph walk.

An example graph is shown in Figure 2, where each node d_i represents a document, w_i a wrapper, and m_i an extracted entity mention. The graph models the relationship between documents, wrappers, and mentions. In order to measure the relative importance of each node within the graph, the Ranker performs a graph walk until all node weights converge. The idea is that nodes are weighted higher if they are connected to many other highly weighted nodes.

We apply this SE algorithm to answer candidates for list questions generated by Ephyra and other TREC QA systems to find additional instances of correct answers that were not in the original candidate set.

3 Proposed Approach

SEAL was originally designed to handle only relevant input seeds. When provided with a mixture of relevant and irrelevant answers from a QA system, the performance would suffer. In this section, we propose three modifications to SEAL to improve its ability to handle noisy input seeds.

3.1 Aggressive Fetcher

For each expansion, SEAL's fetcher concatenates all seeds and sends them as one query to the search engines. However, when the seeds are noisy, the documents fetched are constrained by the irrelevant seeds, which decreases the chance of finding good documents. To overcome this problem, we designed an *aggressive fetcher* (AF) that increases the chance of composing queries containing only relevant seeds. It sends a two-seed query for every possible pair of seeds to the search engines. If there are n input seeds, then the total number of queries sent would be $\binom{n}{2}$. For example, suppose SEAL is given a set of noisy seeds: *Boston*, *Seattle* and *Carnegie-Mellon* (assuming *Carnegie-Mellon* is irrelevant), then by using AF, one query will contain only relevant seeds (as shown in Table 1). The documents are then collected and sent to SEAL's extractor for learning wrappers.

	Queries	Quality
-AF	#1: Boston Seattle Carnegie-Mellon	Low
+AF	#1: Boston Seattle	High
	#2: Boston Carnegie-Mellon	Low
	#3: Seattle Carnegie-Mellon	Low

Table 1: Example queries and their quality given the seeds *Boston*, *Seattle* and *Carnegie-Mellon*, where *Carnegie-Mellon* is assumed to be irrelevant.

3.2 Lenient Extractor

SEAL's extractor requires the longest common contexts to bracket at least one instance of every seed per web page. However, when seeds are noisy, such common contexts usually do not exist or are too short to be useful. To solve this problem, we propose a *lenient extractor* (LE) which only requires the contexts to bracket at least one instance of *a minimum of two* seeds, instead of *every* seed. This increases the chance of finding longest common contexts that bracket only relevant seeds. For instance, suppose SEAL is given the seeds from the previous example (*Boston*, *Seattle* and *Carnegie-Mellon*) and the passage below. Then the extractor would learn the wrappers shown in Table 2.

"While attending a hearing in Boston City Hall, Alan, a professor at Boston University, met Tina, his former student at Seattle University, who is studying at Carnegie-Mellon University Art School and will be working in Seattle City Hall."

	Learned Wrappers
-LE	#1: `at [...] University`
+LE	#1: `at [...] University`
	#2: `in [...] City Hall`

Table 2: Wrappers learned by SEAL's extractor when given the passage in Section 3.2 and the seeds *Boston*, *Seattle* and *Carnegie-Mellon*.

As illustrated, with lenient extraction, SEAL is now able to learn the second wrapper because it brackets one instance of at least two seeds (*Boston* and *Seattle*). This can be very helpful if the list question is asking for city names rather than university names. The extractor then uses these wrappers to extract additional answer candidates, by searching for other strings that fit into the placeholders of the wrappers. Note that the example was simplified for ease of presentation. The wrappers are actually character-based (as opposed to word-based) and are likely to contain HTML tags when generated from real web pages.

3.3 Hinted Expander

Most QA systems use keywords from the question to guide the retrieval of relevant documents and the extraction of answer candidates. We believe these keywords are also important for SEAL to identify additional instances of correct answers. For example, if the seeds are *George Washington*, *John Adams*, and *Thomas Jefferson*, then without using any context from the question, SEAL would output a mixture of founding fathers and presidents of the U.S.A. To solve this problem, we devised a *hinted expansion* (HE) technique that utilizes the context given in the question to constrain SEAL's search space on the Web. This is achieved by appending keywords from the question to every query that is sent to the search engines. The rationale is that the retrieved documents will also match the keywords, which may increase the chance of finding those documents that contain our desired set of answers.

4 Experimental Design

We conducted experiments in two phases. In the first phase, we evaluated the SE approach by applying SEAL to answers generated by Ephyra. In the second phase, we evaluated the approach by applying SEAL to the output from QA systems that performed the best on the list questions in the TREC 15 evaluation. In both phases, the answers found by SEAL were retrieved from the Web instead of the AQUAINT newswire corpus used in the TREC evaluations. However, we rejected answers if they could only be found in the Web and not in the AQUAINT corpus to avoid an unfair advantage over the QA systems: TREC participants were allowed to extract candidates from the Web (or any other source), but they had to identify a supporting document in the AQUAINT corpus for each answer and thus could not return answers that were not covered by the corpus.

Preliminary experiments showed that we can obtain a good balance between the amount and quality of the documents fetched by using only rare question terms as hint words. In particular, we select the three question words that occur least frequently in a sample of the AQUAINT corpus as hints. The candidate answers were evaluated by using the answer keys, composed of regular expression patterns, obtained from the TREC website. We did not extend the patterns with additional correct answers found in our experiments. These answer keys were not officially used in the TREC evaluation; thus the baseline scores we computed for Ephyra and other QA systems in our experiments are slightly different from those officially reported.

4.1 Ephyra

We evaluated our SE approach on Ephyra using the list questions from TREC 13, 14, and 15 (55, 93, and 89 questions, respectively). For each question, the top four answer candidates from Ephyra were given as input seeds to SEAL. Initial experiments showed that by adding additional seeds, the effectiveness of our approach can be improved at the expense of a longer runtime.

We report both mean average precision (MAP) and F_1 scores. For the F_1 scores, we drop answer candidates with low confidence scores by applying a relative cut-off threshold: an answer candidate is dropped if the ratio of its confidence score and the score of the top answer is below a threshold. An optimal threshold for a question is a threshold that maximizes the F_1 score for that particular question.

For each TREC dataset, we conducted three experiments: (1) evaluation of answer candidates using MAP; (2) evaluation using average F_1 with an optimal threshold for each question; and (3) evaluation using average F_1 with thresholds trained by 5-fold cross validation. For each of those 5-fold validations, only one threshold was determined for all questions in the training folds.

	Ephyra	Ephyra's Top 4 Ans.	SEAL	SEAL+LE	SEAL+LE +AF	SEAL+LE +AF+HE
TREC 13	25.95%	21.39%	23.76%	31.43%	34.22%	35.26%
TREC 14	14.45%	8.71%	14.47%	17.04%	16.58%	18.82%
TREC 15	13.42%	9.02%	13.17%	16.87%	17.12%	18.95%

Table 3: Mean average precision of Ephyra, its top four answers, and various SEAL configurations, where LE is Lenient Extractor, AF is Aggressive Fetcher, and HE is Hinted Expander.

	Ephyra	Ephyra's Top 4 Ans.	SEAL	SEAL+LE	SEAL+LE +AF	SEAL+LE +AF+HE
TREC 13	35.74%	26.29%	30.53%	36.47%	40.08%	40.80%
TREC 14	22.83%	14.05%	20.62%	22.81%	22.66%	24.88%
TREC 15	22.42%	14.57%	19.88%	23.30%	24.04%	25.65%

Table 4: Average F_1 of Ephyra, its top four answers, and various SEAL configurations when using an optimal threshold for each question.

4.2 Top QA Systems

We evaluated two SE approaches, SEAL and Google Sets, on the five QA systems that performed the best on the list questions in TREC 15. For each question, the top four answer candidates[4] from those systems were given as input seeds to SEAL and Google Sets. Unlike the candidates found by Ephyra, these candidates were provided without confidence scores; hence, we assumed they all have a score of 1.0. In our experiments with SEAL, we first determined a single threshold that optimizes the average of the F_1 scores of the top five systems in both TREC 13 and 14. We then obtained evaluation results for the top systems in TREC 15 by using this trained threshold. When performing hinted expansion, the keywords (or hint words) for each question were extracted by Ephyra's question analysis component. In our experiments with Google Sets, we requested *Small Sets* of items and again measured the performance in terms of F_1 scores. We also tried requesting *Large Sets* but the results were worse.

5 Results and Discussion

In Tables 3 and 4, we present evaluation results for all answers from Ephyra, only the top four answers, and various configurations of SEAL using the top four answers as seeds. Table 3 shows the MAP for each dataset (TREC 13, 14, and 15), and Table 4 shows for each dataset the average F_1 score when using optimal per-question thresholds. The results indicate that SEAL achieves the best performance when configured with all three proposed extensions. In terms of MAP, the best-configured SEAL improves the quality of the input answers (relatively) by 65%, 116%, 110% for each dataset respectively, and improves Ephyra's overall performance by 36%, 30%, 41%. In terms of optimal F_1, SEAL improves the quality of the input answers by 55%, 77%, 76% and Ephyra's overall performance by 14%, 9%, 14% respectively. These results illustrate that a SE system is capable of improving a QA system's performance on list questions, if we know how to select good thresholds.

In practice, the thresholds are unknown and must be estimated from a training set. Table 5 shows evaluation results using 5-fold cross validation for each dataset (TREC 13, 14, and 15) independently, and the combination of all three datasets (All). For each validation, we determine the threshold that maximizes the F_1 score on the training folds, and we also determine the F_1 score on the test fold by applying the trained threshold. We repeat this validation for each of the five test folds and present the average threshold and F_1 score for each configuration and dataset. The F_1 scores give an estimate of the performance on unseen data and allow a fair com-

[4]Obtained from `http://trec.nist.gov/results`

	Ephyra		SEAL+LE+AF+HE		Hybrid	
	Avg. F_1	**Avg. Threshold**	**Avg.** F_1	**Avg. Threshold**	**Avg.** F_1	**Avg. Threshold**
TREC 13	25.55%	0.3808	30.71%	0.3257	29.04%	0.0796
TREC 14	15.78%	0.2636	15.60%	0.1889	17.13%	0.0108
TREC 15	15.19%	0.1192	15.64%	0.2581	16.47%	0.0123
All	18.03%	0.2883	19.15%	0.2606	19.59%	0.0164

Table 5: Average F_1 of Ephyra, the best-configured SEAL, and the hybrid system, along with thresholds trained by 5-fold cross validation.

TREC 15	**Baseline**	**Top 4 Ans.**	**Google Sets**		**SEAL+LE+AF+HE**		**Hybrid**	
QA Systems	**Avg.** F_1	**Avg.** F_1	**Avg.** F_1	ΔF_1	**Avg.** F_1	ΔF_1	**Avg.** F_1	ΔF_1
lccPA06	44.96%	32.67%	37.89%	-15.72%	40.00%	-11.04%	45.30%	0.76%
cuhkqaepisto	18.27%	17.02%	15.96%	-12.68%	19.75%	8.08%	19.13%	4.70%
NUSCHUAQA1	18.40%	14.99%	16.70%	-9.21%	18.74%	1.86%	18.06%	-1.81%
FDUQAT15A	19.71%	14.32%	18.79%	-4.63%	19.78%	0.38%	20.61%	4.57%
QACTIS06C	17.52%	15.22%	17.05%	-2.72%	18.45%	5.26%	18.38%	4.85%
Average	23.77%	18.84%	21.28%	-10.49%	23.34%	-1.81%	24.30%	2.20%

Table 6: Average F_1 of the QA systems, their top four answers, Google Sets, the best-configured SEAL, the hybrid system, and their relative improvements over the QA systems.

parison across systems. Here, we also introduce a hybrid system (Hybrid) that intersects the answers found by both systems by multiplying their probabilistic scores.

Tables 3, 4, and 5 show that the effectiveness of the SE approach depends on the quality of the initial answer candidates. The improvements are most apparent for the TREC 13 dataset, where Ephyra has a much higher performance compared to TREC 14 and 15. However, the best-configured SEAL did not improve the F_1 score on TREC 14, as reported in Table 5. We suspect that this is due to the comparatively low quality of Ephyra's top four answers for this dataset. The experiments also illustrate that by intersecting the answer candidates found by Ephyra and SEAL, we can eliminate poor answer candidates and partially compensate for the low precision of Ephyra on the harder TREC datasets. However, this comes at the expense of a lower recall, which slightly hurts the performance on the comparatively easier TREC 13 questions. We also evaluated Google Sets on top four answers from Ephyra for TREC 13-15 and obtained F_1 scores of 12%, 11%, and 9% respectively (compared to 29%, 17%, and 16% for our hybrid approach with trained thresholds).

Table 6 shows F_1 scores for the SE approach applied to the output from the five QA systems with the highest performance on the list questions in TREC 15. Again, Hybrid intersects the answers found by the QA system and SEAL by multiplying their confidence scores. Two thresholds were trained separately on the top five systems in both TREC 13 and 14; one for SEAL (0.2376) and another for Hybrid (0.2463). As shown, the performance of Google Sets is worse than SEAL and Hybrid, but better than the top four answers on average. We believe our SE system outperforms Google Sets because we have methods to handle noisy inputs (i.e. AF, LE) and a method for guiding the SE algorithm to search in the right space on the Web (i.e. HE).

The results show that both SEAL and Hybrid are capable of improving four out of the five systems. We observed that one reason why SEAL did not improve "lccPA06" was the incompleteness of the answer keys. Table 7 shows one of many examples where SEAL was penalized for finding additional correct answers. As illustrated, Hybrid improved all systems except "NUSCHUAQA1". The reason is that even though SEAL improved the baseline, their overlapping answer set is too small; thus hurting the recall of Hybrid substantially. Unfortunately,

Question 154.6: Name titles of movies, other than "Superman" movies, that Christopher Reeve acted in.

lccPA06 (F_1: 75%)	SEAL+LE+AF+HE (F_1: 40%)
+Rear Window	+Rear Window
+The Remains of the Day	+The Remains of the Day
+Snakes and Ladders	-The Bostonians
-Superman	-Somewhere in Time
	-Village of the Damned
	-In the Gloaming

Table 7: Example of SEAL being penalized for finding correct answers (all are correct except the last one). Answers found in the answer keys are marked with "+". All four answers from "lccPA06" were used as seeds.

Question 170.6: What are the titles of songs written by John Prine?

NUSCHUAQA1 (F_1: 25%)	SEAL+LE+AF+HE (F_1: 44%)
+I Just Want to Dance With You	+I Just Want to Dance With You
-Titled In Spite of Ourselves	+Christmas in Prison
+Christmas in Prison	+Sam Stone
-Grammy - Winning	-Grandpa was a Carpenter
	-Sabu Visits the Twin Cities Alone
	+Angel from Montgomery

Table 8: Example demonstrating SEAL's ability to handle noisy input seeds. All four answers from "NUSCHUAQA1" were used as seeds. Again, SEAL is penalized for finding correct answers (all answers are correct).

for the top TREC 15 systems we only had access to the answers that were actually submitted by the participants, whereas for Ephyra we could utilize the entire list of generated answer candidates, including those that fell below the cutoff threshold for list questions. Nevertheless, the hybrid approach could improve the baseline by more than 2% on average in terms of F_1 score. Table 8 shows that the best-configured SEAL is capable of expanding only the relevant seeds even when given a set of noisy seeds. Neither Google Sets nor the original SE algorithm without the proposed extensions could expand these seeds with additional candidates.

On average, SEAL required about 5 seconds for querying the search engines, 10 seconds for crawling the Web, 20 seconds for extracting answer candidates from the web pages, and 5 seconds for ranking the candidates. Note that the SE system has not been optimized extensively. The runtime of the web page retrieval step and much of the search is due to network latency and can be reduced if the search is performed locally.

6 Conclusion and Future Work

We have shown that our SE approach is capable of improving the performance of QA systems on list questions by utilizing only their top four answer candidates as seeds. We have also illustrated a feasible and effective method for integrating a SE approach into any QA system. We would like to emphasize that for each of the experiments we conducted, all that the SE system received as input were the top four noisy answers from a QA system and three keywords from the TREC questions. We have shown that higher quality candidates support more effective set expansion. In the future, we will investigate how to utilize more answer candidates from the QA system and determine the minimal quality of those candidates required for SE approach to make an improvement.

We have also shown that, in terms of F_1 scores with trained thresholds, the hybrid method improves the Ephyra QA system on all datasets and also improves four out of the five systems that performed

the best on the list questions in TREC 15. However, the final list of answers only comprises candidates found by both the QA system and the SE algorithm. In future experiments, we will investigate other methods of merging answer candidates, such as taking the union of answers from both systems. We expect further improvements from adding candidates that are found only by the QA system, but it is unclear how the confidence measures from the two systems can be combined effectively.

We would also like to emphasize that the SE approach is entirely language independent, and thus can be readily applied to answer candidates in other languages. In future experiments, we will investigate its performance on question answering tasks in languages such as Chinese and Japanese.

As pointed out previously, the performance of the SE approach highly depends on the accuracy of the seeds. However, QA systems are usually not optimized to provide few high-precision results, but treat precision and recall as equally important. This leaves room for further improvements, e.g. by applying stricter answer validation techniques to the seeds used for SE.

We also plan to analyze the effectiveness of our approach across different question types and evaluate it on more complex questions such as the rigid list questions in the new TAC QA evaluation, which ask for opinion holders and subjects.

Acknowledgements

This work was supported in part by the Google Research Awards program, IBM Open Collaboration Agreement #W0652159, and the Defense Advanced Research Projects Agency (DARPA) under Contract No. NBCHD030010.

References

H.T. Dang, J. Lin, and D. Kelly. 2006. Overview of the TREC 2006 question answering track. *Proceedings of the Fifteenth Text REtrieval Conference*.

H.T. Dang, D. Kelly, and J. Lin. 2007. Overview of the TREC 2007 question answering track. *Proceedings of the Sixteenth Text REtrieval Conference*.

J. Ko, L. Si, and E. Nyberg. 2007. A probabilistic framework for answer selection in question answering. *Proceedings of NAACL-HLT*.

N. Schlaefer, P. Gieselmann, and G. Sautter. 2006. The Ephyra QA system at TREC 2006. *Proceedings of the Fifteenth Text REtrieval Conference*.

N. Schlaefer, G. Sautter, J. Ko, J. Betteridge, M. Pathak, and E. Nyberg. 2007. Semantic extensions of the Ephyra QA system in TREC 2007. *To appear in: Proceedings of the Sixteenth Text REtrieval Conference*.

R.C. Wang and W.W. Cohen. 2007. Language-independent set expansion of named entities using the web. *Proceedings of IEEE International Conference on Data Mining*.