Semi-Automatic Entity Set Refinement

Vishnu Vyas and Patrick Pantel
Yahoo! Labs
Santa Clara, CA 95054
{vishnu,ppantel}@yahoo-inc.com

Abstract

State of the art set expansion algorithms produce varying quality expansions for different entity types. Even for the highest quality expansions, errors still occur and manual refinements are necessary for most practical uses. In this paper, we propose algorithms to aide this refinement process, greatly reducing the amount of manual labor required. The methods rely on the fact that most expansion errors are systematic, often stemming from the fact that some seed elements are ambiguous. Using our methods, empirical evidence shows that average R-precision over random entity sets improves by 26% to 51% when given from 5 to 10 manually tagged errors. Both proposed refinement models have linear time complexity in set size allowing for practical online use in set expansion systems.

1 Introduction

Sets of named entities are extremely useful in a variety of natural language and information retrieval tasks. For example, companies such as Yahoo! and Google maintain sets of named entities such as cities, products and celebrities to improve search engine relevance.

Manually creating and maintaining large sets of named entities is expensive and laborious. In response, many automatic and semi-automatic methods of creating sets of named entities have been proposed, some are supervised (Zhou and Su, 2001), unsupervised (Pantel and Lin 2002, Nadeau et al. 2006), and others semi-supervised (Kozareva et al. 2008). Semi-supervised approaches are often used in practice since they allow for targeting specific entity classes such as *European Cities* and *French Impressionist Painters*. Methods differ in complexity from simple ones using lexico-syntactic patterns (Hearst 1992) to more complicated techniques based on distributional similarity (Paşca 2007a).

Even for state of the art methods, expansion errors inevitably occur and manual refinements are necessary for most practical uses requiring high precision (such as for query interpretation at commercial search engines). Looking at expansions from state of the art systems such as GoogleSets[1] , we found systematic errors such as those resulting from ambiguous seed instances. For example, consider the following seed instances for the target set *Roman Gods*:

`Minerva, Neptune, Baccus, Juno, Apollo`

GoogleSet's expansion as well others employing distributional expansion techniques consists of a mishmash of Roman Gods and celestial bodies, originating most likely from the fact that *Neptune* is both a *Roman God* and a *Planet*. Below is an excerpt of the GoogleSet expansion:

`Mars, Venus, *Moon, Mercury, *asteroid, Jupiter, *Earth, *comet, *Sonne, *Sun, …`

The inherent semantic similarity between the errors can be leveraged to quickly clean up the expansion. For example, given a manually tagged error "*asteroid*", a distributional similarity thesaurus

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

such as (Lin 1998)[2] can identify *comet* as similar to *asteroid* and therefore potentially also as an error. This method has its limitations since a manually tagged error such as *Earth* would correctly remove *Moon* and *Sun*, but it would also incorrectly remove *Mars*, *Venus* and *Jupiter* since they are also similar to *Earth*[3].

In this paper, we propose two algorithms to improve the precision of automatically expanded entity sets by using minimal human negative judgments. The algorithms leverage the fact that set expansion errors are systematically caused by ambiguous seed instances which attract incorrect instances of an unintended entity type. We use distributional similarity and sense feature modeling to identify such unintended entity types in order to quickly clean up errors with minimal manual labor. We show empirical evidence that average R-precision over random entity sets improves by 26% to 51% when given from 5 to 10 manually tagged errors. Both proposed refinement models have linear time complexity in set size allowing for practical online use in set expansion systems.

The remainder of this paper is organized as follows. In the next section we review related work and position our contribution within its landscape. Section 3 presents our task of dynamically modeling the similarity of a set of words and describes algorithms for refining sets of named entities. The datasets and our evaluation methodology used to perform our experiments are presented in Section 4 and in Section 5 we describe experimental results. Finally, we conclude with some discussion and future work.

2 Related Work

There is a large body of work for automatically building sets of named entities using various techniques including supervised, unsupervised and semi-supervised methods. Supervised techniques use large amounts of training data to detect and classify entities into coarse grained classes such as *People*, *Organizations*, and *Places* (Bunescu and Mooney 2004; Etzioni et al. 2005). On the other hand, unsupervised methods require no training data and rely on approaches such as clustering, targeted patterns and co-occurrences to extract sets of entities (Pantel and Lin 2002; Downey et al. 2007).

Semi-supervised approaches are often used in practice since they allow for targeting specific entity classes. These methods rely on a small set of seed examples to extract sets of entities. They either are based on distributional approaches or employ lexico-syntactic patterns to expand a small set of seeds to a larger set of candidate expansions. Some methods such as (Riloff and Shepherd 1997; Riloff and Jones 1999; Banko et al. 2007;Paşca 2007a) use lexico-syntactic patterns to expand a set of seeds from web text and query logs. Others such as (Paşca et al. 2006; Paşca 2007b; Paşca and Durme 2008) use distributional approaches. Wang and Cohen (2007) use structural cues in semi-structured text to expand sets of seed elements. In all methods however, expansion errors inevitably occur. This paper focuses on the task of post processing any such system's expansion output using minimal human judgments in order to remove expansion errors.

Using user feedback to improve a system's performance is a common theme within many information retrieval and machine learning tasks. One form of user feedback is active learning (Cohn et al. 1994), where one or more classifiers are used to focus human annotation efforts on the most beneficial test cases. Active learning has been successfully applied to various natural language tasks such as parsing (Tang et al. 2001), POS tagging (Dagan and Engelson 1995) and providing large amounts of annotations for common natural language processing tasks such as word sense disambiguation (Banko and Brill 2001). Relevance feedback is another popular feedback paradigm commonly used in information retrieval (Harman 1992), where user feedback (either explicit or implicit) is used to refine the search results of an IR system. Relevance feedback has been successfully applied to many IR applications including content-based image retrieval (Zhouand Huang 2003) and web search (Vishwa et al. 2005). Within NLP applications relevance feedback has also been used to generate sense tagged examples for WSD tasks (Stevenson et al. 2008), and Question Answering (Negri 2004). Our methods use relevance feedback in the form of negative examples to refine the results of a set expansion system.

[2] See http://demo.patrickpantel.com/ for a demonstration of the distributional thesaurus.

[3] In practice, this problem is rare since most terms that are similar in one of their senses tend not to be similar in their other senses.

3 Dynamic Similarity Modeling

The set expansion algorithms discussed in Section 2 often produce high quality entity sets, however inevitably errors are introduced. Applications requiring high precision sets must invest significantly in editorial efforts to clean up the sets. Although companies like Yahoo! and Google can afford to routinely support such manual labor, there is a large opportunity to reduce the refinement cost (i.e., number of required human judgments).

Recall the set expansion example of *Roman Gods* from Section 1. Key to our approach is the hypothesis that *most expansion errors result from some systematic cause*. Manual inspection of expansions from GoogleSets and distributional set expansion techniques revealed that most errors are due to the inherent ambiguity of seed terms (such as *Neptune* in our example) and data sparseness (such as *Sonne* in our example, a very rare term). The former kind of error is systematic and can be leveraged by an automatic method by assuming that any entity semantically similar to an identified error will also be erroneous.

In this section, we propose two methods for leveraging this hypothesis. In the first method, described in Section 3.1, we use a simple distributional thesaurus and remove all entities which are distributionally similar to manually identified errors. In the second method, described in Section 3.2, we model the semantics of the seeds using distributional features and then dynamically change the feature space according to the manually identified errors and rerank the entities in the set. Both methods rely on the following two observations:

a) **Many expansion errors are systematically caused by ambiguous seed examples** which draw in several incorrect entities of its unintended senses (such as seed *Neptune* in our *Roman Gods* example which drew in celestial bodies such as *Earth* and *Sun*);
b) **Entities which are similar in one sense are usually not similar in their other senses**. For example, *Apple* and *Sun* are similar in their *Company* sense but their other senses (*Fruit* and *Celestial Body*) are not similar. Our example in Section 1 illustrates a rare counterexample where *Neptune* and *Mercury* are similar in both their *Planets* and *Roman Gods* senses.

Task Outline: Our task is to remove errors from entity sets by using a minimal amount of manual judgments. Incorporating feedback into this process can be done in multiple ways. The most flexible system would allow a judge to iteratively remove as many errors as desired and then have the system automatically remove other errors in each iteration. Because it is intractable to test arbitrary numbers of manually identified errors in each iteration, we constrain the judge to identify at most one error in each iteration.

Although this paper focuses solely on removing errors in an entity set, it is also possible to improve expanded sets by using feedback to add new elements to the sets. We consider this task out of scope for this paper.

3.1 Similarity Method (SIM)

Our first method directly models observation *a*) in the previous section. Following Lin (1998), we model the similarity between entities using the distributional hypothesis, which states that similar terms tend to occur in similar contexts (Harris 1985). A semantic model can be obtained by recording the surrounding contexts for each term in a large collection of unstructured text. Methods differ in their definition of a context (e.g., *text window* or *syntactic relations*), or a means to weigh contexts (e.g., *frequency*, *tf-idf*, *pointwise mutual information*), or ultimately in measuring the similarity between two context vectors (e.g., using *Euclidean distance*, *Cosine*, *Dice*). In this paper, we use a text window of size 1, we weigh our contexts using pointwise mutual information, and we use the cosine score to compute the similarity between context vectors (i.e., terms). Section 5.1 describes our source corpus and extraction details. Computing the full similarity matrix for many terms over a very large corpus is computationally intensive. Our specific implementation follows the one presented in (Bayardo et al. 2007).

The similarity matrix computed above is then directly used to refine entity sets. Given a manually identified error at each iteration, we automatically remove each entity in the set that is found to be semantically similar to the error. The similarity threshold was determined by manual inspection and is reported in Section 5.1.

Due to observation *b*) in the previous section, we expect that this method will perform poorly on

entity sets such as the one presented in our example of Section 1 where the manual removal of *Earth* would likely remove correct entities such as *Mars*, *Venus* and *Jupiter*. The method presented in the next section attempts to alleviate this problem.

3.2 Feature Modification Method (FMM)

Under the distributional hypothesis, the semantics of a term are captured by the contexts in which it occurs. The *Feature Modification Method* (FMM), in short, attempts to automatically discover the incorrect contexts of the unintended senses of seed elements and then filters out expanded terms whose contexts do not overlap with the other contexts of the seed elements.

Consider the set of seed terms *S* and an erroneous expanded instance *e*. In the SIM method of Section 3.1 all set elements that have a feature vector (i.e., context vector) similar to *e* are removed. The *Feature Modification Method* (FMM) instead tries to identify the subset of features of the error *e* which represent the unintended sense of the seed terms *S*. For example, let *S* = {*Minerva*, *Neptune*, *Baccus*, *Juno*, *Apollo*}. Looking at the contexts of these words in a large corpus, we construct a centroid context vector for *S* by taking a weighted average of the contexts of the seeds in *S*. In Wikipedia articles we see contexts (i.e., features) such as[4]:

```
attack, kill, *planet, destroy,
Goddess, *observe, statue, *launch,
Rome, *orbit, …
```

Given an erroneous expansion such as *e* = *Earth*, we postulate that removing the intersecting features from *Earth*'s feature vector and the above feature vector will remove the unintended *Planet* sense of the seed set caused by the seed element *Neptune*. The intersecting features that are removed are bolded in the above feature vector for *S*. The similarity between this modified feature vector for *S* and all entities in the expansion set can be recomputed as described in Section 3.1. Entities with a low similarity score are removed from the expanded set since they are assumed to be part of the unintended semantic class (*Planet* in this example).

Unlike the SIM method from Section 3.1, this method is more stable with respect to observation *b*) in Section 3. We showed that SIM would incorrectly remove expansions such as *Mars*, *Venus* and *Jupiter* given the erroneous expansion *Earth*. The FMM method would instead remove the *Planet* features from the seed feature vectors and the remaining features would still overlap with *Mars*, *Venus* and *Jupiter*'s *Roman God* sense.

Efficiency: FMM requires online similarity computations between centroid vectors and all elements of the expanded set. For large corpora such as Wikipedia articles or the Web, feature vectors are large and storing them in memory and performing similarity computations repeatedly for each editorial judgment is computationally intensive. For example, the size of the feature vector for a single word extracted from Wikipedia can be in the order of a few gigabytes. Storing the feature vectors for all candidate expansions and the seed set is inefficient and too slow for an interactive system. The next section proposes a solution that makes this computation very fast, requires little memory, and produces near perfect approximations of the similarity scores.

3.3 Approximating Cosine Similarity

There are engineering optimizations that are available that allow us to perform a near perfect approximation of the similarity computation from the previous section. The proposed method requires us to only store the shared features between the centroid and the words rather than the complete feature vectors, thus reducing our space requirements dramatically. Also, FMM requires us to repeatedly calculate the cosine similarity between a modified centroid feature vector and each candidate expansion at each iteration. Without the full context vectors of all candidate expansions, computing the exact cosine similarity is impossible. Given, however, the original cosine scores between the seed elements and the candidate expansions before the first refinement iteration as well as the shared features, we can approximate with very high accuracy the updated cosine score between the modified centroid and each candidate expansion. Our method relies on the fact that features (i.e., contexts) are only ever removed from the original centroid – no new features are ever added.

Let μ be the original centroid representing the seed instances. Given an expansion error *e*, FMM creates a modified centroid by removing all fea-

[4] The full feature vector for these and all other terms in Wikipedia can be found at http://demo.patrickpantel.com/..

tures intersecting between e and μ. Let μ' be this modified centroid. FMM requires us to compute the similarity between μ' and all candidate expansions x as:

$$\cos(x,\mu') = \frac{\sum x_i \mu'_i}{\|x\| \cdot \|\mu'\|}$$

where i iterates over the feature space.

In our efficient setting, the only element that we do not have for calculating the exact cosine similarity is the norm of x, $\|x\|$. Given that we have the original cosine similarity score, $cos(x, \mu)$ and that we have the shared features between the original centroid μ and the candidate expansion x we can calculate $\|x\|$ as:

$$\|x\| = \frac{\sum x_i \mu_i}{\|\mu\| \cdot \cos(x,\mu)}$$

Combining the two equations, have:

$$\cos(x,\mu') = \cos(x,\mu) \cdot \frac{\sum x_i \mu'_i}{\sum x_i \mu_i} \cdot \frac{\|\mu\|}{\|\mu'\|}$$

In the above equation, the modified cosine score can be considered as an update to the original cosine score, where the update depends only on the shared features and the original centroid. The above update equation can be used to recalculate the similarity scores without resorting to an expensive computation involving complete feature vectors.

Storing the original centroid is expensive and can be approximated instead from only the shared features between the centroid and all instances in the expanded set. We empirically tested this approximation by comparing the cosine scores between the candidate expansions and both the true centroid and the approximated centroid. The average error in cosine score was 9.5E-04 ± 7.83E-05 (95% confidence interval).

4 Datasets and Baseline Algorithm

We evaluate our algorithms against manually scraped gold standard sets, which were extracted from Wikipedia to represent a random collection of concepts. Section 4.1 discusses the gold standard sets and the criteria behind their selection. To present a statistically significant view of our results we generated a set of trials from gold standard sets to use as seeds for our seed set expansion algorithm. Also, in section 4.2 we discuss how we can simulate editorial feedback using our gold standard sets.

4.1 Gold Standard Entity Sets

The gold standard sets form an essential part of our evaluation. These sets were chosen to represent a single concept such as *Countries* and *Archbishops of Canterbury*. These sets were selected from the *List of* pages from Wikipedia[5]. We randomly sorted the list of every noun occurring in Wikipedia. Then, for each noun we verified whether or not it existed in a Wikipedia list, and if so we extracted this list – up to a maximum of 50 lists. If a noun belonged to multiple lists, the authors chose the list that seemed most appropriate. Although this does not generate a perfect random sample, diversity is ensured by the random selection of nouns and relevancy is ensured by the author adjudication.

Lists were then scraped from the Wikipedia website and they went through a manual cleanup process which included merging variants. . The 50 sets contain on average 208 elements (with a minimum of 11 and a maximum of 1116 elements) for a total of 10,377 elements. The final gold standard lists contain 50 sets including *classical pianists*, *Spanish provinces*, *Texas counties*, *male tennis players*, *first ladies*, *cocktails*, *bottled water brands*, and *Archbishops of Canterbury*[6].

4.2 Generation of Experimental Trials

To provide a statistically significant view of the performance of our algorithm, we created more than 1000 trials as follows. For each of the gold standard seed sets, we created 30 random sortings. These 30 random sortings were then used to generate trial seed sets with a maximum size of 20 seeds.

4.3 Simulating User Feedback and Baseline Algorithm

User feedback forms an integral part of our algorithm. We used the gold standard sets to judge the

[5] In this paper, extractions from Wikipedia are taken from a snapshot of the resource in December 2007.

[6] The gold standard is available for download at http://www.patrickpantel.com/cgi-bin/Web/Tools/getfile.pl?type=data&id=sse-gold/wikipedia.20071218.goldsets.tgz

candidate expansions. The judged expansions were used to simulate user feedback by marking those candidate expansions that were incorrect. The first candidate expansion that was marked incorrect in each editorial iteration was used as the editor's negative example and was given to the system as an error.

In the next section, we report R-precision gains at each iteration in the editorial process for our two methods described in Section 3. Our baseline method simply measures the gains obtained by removing the first incorrect entry in a candidate expansion set at each iteration. This simulates the process of manually cleaning a set by removing one error at a time.

Table 1. R-precision of the three methods with 95% confidence bounds.

ITERATION	*BASELINE*	*SIM*	*FMM*
1	0.219±0.012	**0.234±0.013**	0.220±0.015
2	0.223±0.013	**0.242±0.014**	0.227±0.017
3	0.227±0.013	**0.251±0.015**	0.235±0.019
4	0.232±0.013	**0.26±0.016**	0.252±0.021
5	0.235±0.014	**0.266±0.017**	**0.267±0.022**
6	0.236±0.014	0.269±0.017	**0.282±0.023**
7	0.238±0.014	0.273±0.018	**0.294±0.023**
8	0.24±0.014	0.28±0.018	**0.303±0.024**
9	0.242±0.014	0.285±0.018	**0.315±0.025**
10	0.243±0.014	0.286±0.018	**0.322±0.025**

5 Experimental Results

5.1 Experimental Setup

Wikipedia[5] served as the source corpus for our algorithms described in Sections 3.1 and 3.2. All articles were POS-tagged using (Brill 1995) and later chunked using a variant of (Abney 1991). Corpus statistics from this processed text were collected to build the similarity matrix for the SIM method (Section 3.1) and to extract the features required for the FMM method (Section 3.2). In both cases corpus statistics were extracted over the semi-syntactic contexts (chunks) to approximate term meanings. The minimum similarity thresholds were experimentally set to 0.15 and 0.11 for the SIM and FMM algorithms respectively.

Each experimental trial described in Section 4.2, which consists of a set of seed instances of one of our 50 random semantic classes, was expanded using a variant of the distributional set expansion algorithm from Sarmento et al. (2007). The expansions were judged against the gold standard and each candidate expansion was marked as either correct or incorrect. This set of expanded and judged candidate files were used as inputs to the algorithms described in Sections 3.1 and 3.2. Choosing the first candidate expansion that was judged as incorrect simulated our user feedback. This process was repeated for each iteration of the algorithm and results are reported for 10 iterations.

The outputs of our algorithms were again judged against the gold standard lists and the performance was measured in terms of precision gains over the baseline at various ranks. Precision gain for an algorithm over a baseline is the percentage increase in precision for the same values of parameters of the algorithm over the baseline. Also, as the size of our gold standard lists vary, we report another commonly used statistic, R-precision. R-precision for any set is the precision at the size of the gold standard set. For example, if a gold standard set contains 20 elements, then R-precision for any set expansion is measured as the precision at rank 20. The average R-precision over each set is then reported.

5.2 Quantitative Analysis

Table 1 lists the performance of our baseline algorithm (Section 4.3) and our proposed methods SIM and FMM (Sections 3.1 and 3.2) in terms of their R-precision with 95% confidence bounds over 10 iterations of each algorithm.

The FMM of Section 3.2 is the best performing method in terms of R-precision reaching a maximum value of 0.322 after the 10^{th} iteration. For small numbers of iterations, however, the SIM method outperforms FMM since it is bolder in its refinements by removing all elements similar to the tagged error. Inspection of FMM results showed that bad instances get ranked lower in early iterations but it is only after 4 or 5 iterations that they get pushed passed the similarity threshold (accounting for the low marginal increase in precision gain for FMM in the first 4 to 5 iterations).

FMM outperforms the SIM method by an average of 4% increase in performance (13% improvement after 10 iterations). However both the FMM and the SIM method are able to outperform

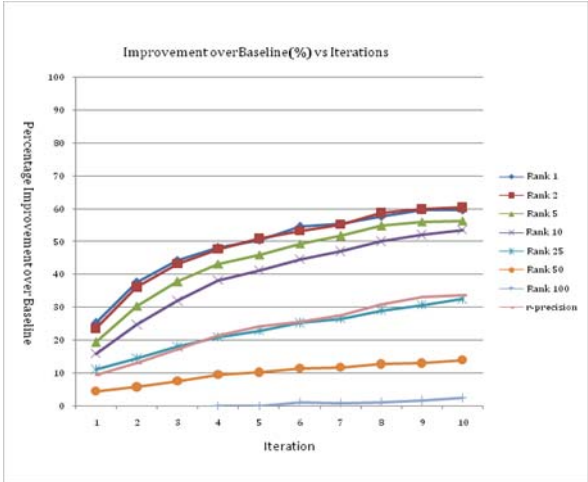

Figure 1. Precision gain over baseline algorithm for SIM method.

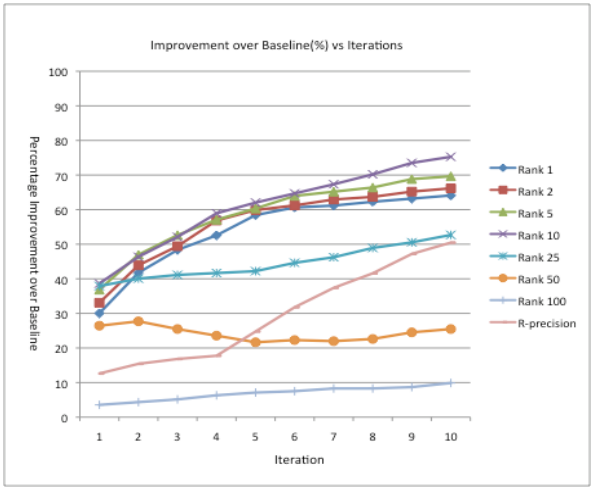

Figure 2. Precision gain over baseline algorithm for FMM method.

the baseline method. Using the FMM method one would achieve an average of 17% improvement in R-precision over manually cleaning up the set (32.5% improvement after 10 iterations). Using the SIM method one would achieve an average of 13% improvement in R-precision over manually cleaning up the set (17.7% improvement after 10 iterations).

5.3 Intrinsic Analysis of the SIM Algorithm

Figure 1 shows the precision gain of the similarity matrix based algorithm over the baseline algorithm. The results are shown for precision at ranks 1, 2, 5, 10, 25, 50 and 100, as well as for R-precision. The results are also shown for the first 10 iterations of the algorithm.

SIM outperforms the baseline algorithm for all ranks and increases in gain throughout the 10 iterations. As the number of iterations increases the change in precision gain levels off. This behavior can be attributed to the fact that we start removing errors from top to bottom and in each iteration the rank of the error candidate provided to the algorithm is lower than in the previous iteration. This results in errors which are not similar to any other candidate expansions. These are random errors and the discriminative capacity of this method reduces severely.

Figure 1 also shows that the precision gain of the similarity matrix algorithm over the baseline algorithm is higher at ranks 1, 2 and 5. Also, the performance increase drops at ranks 50 and 100. This is because low ranks contain candidate expansions that are random errors introduced due to data sparsity. Such unsystematic errors are not detectable by the SIM method.

5.4 Intrinsic Analysis of the FMM Algorithm

The feature modification method of Section 3.2 shows similar behavior to the SIM method, however as Figure 2 shows, it outperforms SIM method in terms of precision gain for all values of ranks tested. This is because the FMM method is able to achieve fine-grained control over what it removes and what it doesn't, as described in Section 5.2.

Another interesting aspect of FMM is illustrated in the R-precision curve. There is a sudden jump in precision gain after the fifth iteration of the algorithm. In the first iterations only few errors are pushed beneath the similarity threshold as centroid features intersecting with tagged errors are slowly removed. As the feature vector for the centroid gets smaller and smaller, remaining features look more and more unambiguous to the target entity type and erroneous example have less chance of overlapping with the centroid causing them to be pushed pass the conservative similarity threshold. Different conservative thresholds yielded similar curves. High thresholds yield bad performance since all but the only very prototypical set instances are removed as errors.

The R-precision measure indirectly models recall as a function of the target coverage of each set. We also directly measured recall at various ranks

and FMM outperformed SIM at all ranks and iterations.

5.5 Discussion

In this paper we proposed two techniques which use user feedback to remove systematic errors in set expansion systems caused by ambiguous seed instances. Inspection of expansion errors yielded other types of errors.

First, model errors are introduced in candidate expansion sets by noise from various preprocessing steps involved in generating the expansions. Such errors cause incorrect contexts (or features) to be extracted for seed instances and ultimately can cause erroneous expansions to be produced. These errors do not seem to be systematic and are hence not discoverable by our proposed method.

Other errors are due to data sparsity. As the feature space can be very large, the difference in similarity between a correct candidate expansion and an incorrect expansion can be very small for sparse entities. Previous approaches have suggested removing candidate expansions for which too few statistics can be extracted, however at the great cost of recall (and lower R-precision).

6 Conclusion

In this paper we presented two algorithms for improving the precision of automatically expanded entity sets by using minimal human negative judgments. We showed that systematic errors which arise from the semantic ambiguity inherent in seed instances can be leveraged to automatically refine entity sets. We proposed two techniques: SIM which boldly removes instances that are distributionally similar to errors, and FMM which more conservatively removes features from the seed set representing its unintended (ambiguous) concept in order to rank lower potential errors.

We showed empirical evidence that average R-precision over random entity sets improves by 26% to 51% when given from 5 to 10 manually tagged errors. These results were reported by testing the refinement algorithms on a set of 50 randomly chosen entity sets expanded using a state of the art expansion algorithm. Given very small amounts of manual judgments, the SIM method outperformed FMM (up to 4 manual judgments). FMM outperformed the SIM method given more than 6 manual judgments. Both proposed refinement models have linear time complexity in set size allowing for practical online use in set expansion systems.

This paper only addresses techniques for removing erroneous entities from expanded entity sets. A complimentary way to improve performance would be to investigate the addition of relevant candidate expansions that are not already in the initial expansion. We are currently investigating extensions to FMM that can efficiently add new candidate expansions to the set by computing the similarity between modified centroids and all terms occurring in a large body of text.

We are also investigating ways to use the findings of this work to a priori remove ambiguous seed instances (or their ambiguous contexts) before running the initial expansion algorithm. It is our hope that most of the errors identified in this work could be automatically discovered without any manual judgments.

References

Abney, S. Parsing by Chunks. 1991. In: Robert Berwick, Steven Abney and Carol Tenny (eds.), *Principle-Based Parsing*. Kluwer Academic Publishers, Dordrecht.

Banko, M. and Brill, E. 2001. Scaling to very large corpora for natural language disambiguation. In *Proceedings of ACL-2001*.pp 26-33. Morristown, NJ.

Banko, M.; Cafarella, M.; Soderland, S.; Broadhead, M.; Etzioni, O. 2007. Open Information Extraction from the Web. In *Proceedings of IJCAI-07*.

Bayardo, R. J; Yiming Ma,; Ramakrishnan Srikant.; Scaling Up All-Pairs Similarity Search. In Proc. of the 16th Int'l Conf. on World Wide Web. pp 131-140 2007.

Brill, E. 1995. Transformation-Based Error-Driven Learning and Natural Language Processing: A Case Study in Part of Speech Tagging. *Computational Linguistics*.

Bunescu, R. and Mooney, R. 2004 Collective Information Extraction with Relational Markov Networks. In *Proceedings of ACL-04*.pp. 438-445.

Cohn, D. A., Atlas, L., and Ladner, R. E. 1994. Improving Generalization with Active Learning. Machine Learning, 15(2):201-221. Springer, Netherlands.

Dagan, I. and Engelson, S. P. 1995. Selective Sampling in Natural Language Learning. In *Proceedings of IJCAI-95 Workshop on New Approaches to Learning for Natural Language Processing*. Montreal, Canada.

Downey, D.; Broadhead, M; Etzioni, O. 2007. Locating Complex Named Entities in Web Text. In *Proceedings of IJCAI-07.*

Etzioni, O.; Cafarella, M.; Downey. D.; Popescu, A.; Shaked, T; Soderland, S.; Weld, D.; Yates, A. 2005. Unsupervised named-entity extraction from the Web: An Experimental Study. In Artificial Intelligence, 165(1):91-134.

Harris, Z. 1985. Distributional Structure. In: Katz, J. J. (ed.), The Philosophy of Linguistics. New York: Oxford University Press. pp. 26-47.

Harman, D. 1992. Relevance feedback revisited. In *Proceeedings of SIGIR-92*. Copenhagen, Denmark.

Hearst, M. A. 1992.Automatic acquisition of hyponyms from large text corpora.In *Proceedings of COLING-92*. Nantes, France.

Kozareva, Z., Riloff, E. and Hovy, E. 2008. Semantic Class Learning from the Web with Hyponym Pattern Linkage Graphs.In *Proceedings of ACL-08*.pp 1048-1056. Columbus, OH

Lin, D. 1998.Automatic retrieval and clustering of similar words.In *Proceedings of COLING/ACL-98*.pp. 768–774. Montreal, Canada.

Nadeau, D., Turney, P. D. and Matwin., S. 2006. Unsupervised Named-Entity Recognition: Generating Gazetteers and Resolving Ambiguity. In *Advances in Artifical Intelligence*.pp 266-277. Springer Berlin, Heidelberg.

Negri, M. 2004. Sense-based blind relevance feedback for question answering. In *Proceedings of SIGIR-04 Workshop on Information Retrieval For Question Answering (IR4QA)*. Sheffield, UK,

Pantel, P. and Lin, D. 2002. Discovering Word Senses from Text. In *Proceedings of KDD-02*.pp. 613-619. Edmonton, Canada.

Paşca, M. 2007a.Weakly-supervised discovery of named entities using web search queries. In *Proceedings of CIKM-07*.pp. 683-690.

Pasca, M. 2007b. Organizing and Searching the World Wide Web of Facts - Step Two: Harnessing the Wisdom of the Crowds. In *Proceedings of WWW-07*. pp. 101-110.

Paşca, M.; Lin, D.; Bigham, J.; Lifchits, A.; Jain, A. 2006. Names and Similarities on the Web: Fact Extraction in the Fast Lane. In *Proceedings of ACL-2006*.pp. 113-120.

Paşca, M. and Durme, B.J. 2008. Weakly-supervised Acquisition of Open-Domain Classes and Class Attributes from Web Documents and Query Logs. In *Proceedings of ACL-08*.

Riloff, E. and Jones, R. 1999 Learning Dictionaries for Information Extraction by Multi-Level Boostrapping.In *Proceedings of AAAI/IAAAI-99*.

Riloff, E. and Shepherd, J. 1997. A corpus-based approach for building semantic lexicons.In *Proceedings of EMNLP-97*.

Sarmento, L.; Jijkuon, V.; de Rijke, M.; and Oliveira, E. 2007. "More like these": growing entity classes from seeds. In *Proceedings of CIKM-07*. pp. 959-962. Lisbon, Portugal.

Stevenson, M., Guo, Y. and Gaizauskas, R. 2008. Acquiring Sense Tagged Examples using Relevance Feedback. In *Proceedings ofCOLING-08.* Manchester UK.

Tang, M., Luo, X., and Roukos, S. 2001. Active learning for statistical natural language parsing.In *Proceedings of ACL-2001*.pp 120 -127. Philadelphia, PA.

Vishwa. V, Wood, K., Milic-Frayling, N. and Cox, I. J. 2005. Comparing Relevance Feedback Algorithms for Web Search. In *Proceedings of WWW 2005*. Chiba, Japan.

Wang. R.C. and Cohen, W.W. 2007.Language-Independent Set Expansion of Named Entities Using the Web.In *Proceedings of ICDM-07*.

Zhou, X. S. and Huang, S. T. 2003. Relevance Feedback in Image Retrieval: A Comprehensive Review - Xiang Sean Zhou, Thomas S. Huang Multimedia Systems. pp 8:536-544.

Zhou, G. and Su, J. 2001. Named entity recognition using an HMM-based chunk tagger. In *Proceedings of ACL-2001*.pp. 473-480. Morristown, NJ.