Fine-grained Entity Set Refinement with User Feedback

Bonan Min
New York University
715 Broadway, 7th floor
New York, NY 10003 USA
min@cs.nyu.edu

Ralph Grishman
New York University
715 Broadway, 7th floor
New York, NY 10003 USA
grishman@cs.nyu.edu

Abstract

State of the art semi-supervised entity set expansion algorithms produce noisy results, which need to be refined manually. Sets expanded for intended fine-grained concepts are especially noisy because these concepts are not well represented by the limited number of seeds. Such sets are usually incorrectly expanded to contain elements of a more general concept. We show that fine-grained control is necessary for refining such sets and propose an algorithm which uses both positive and negative user feedback for iterative refinement. Experimental results show that it improves the quality of fine-grained sets significantly.

1 Introduction

Entity set expansion is a well-studied problem with several techniques proposed (Bunescu and Mooney 2004, Etzioni et al. 2005, Wang and Cohen 2007, Sarmento et al. 2007, Pasca 2007, Pasca 2004, Pantel et al. 2009, Pantel and Lin 2002, Vickrey et al. 2010). In practice, semi-supervised methods are preferred since they require only a handful of seeds and are more flexible for growing various types of entity sets. However, they usually produce noisy sets, which need to be refined (Vyas and Pantel, 2009). Fine-grained sets such as *National Capitals* are particularly noisy. Such concepts are intrinsically hard because they're not well represented by initial seeds. Moreover, most related instances have a limited number of features, thus making it hard to retrieve them.

We examined a few sets expanded for fine-grained concepts and observed that lots of erroneous expansions are elements of a more general concept, whose sense overlaps and subsumes the intended sense. For example, the concept *National Capitals* is expanded to contain *Major cities*. In such cases, a proposed feature-pruning technique using user-tagged expansion errors to refine sets (Vyas and Pantel 2009) removes some informative features of the target concept. Moreover, since refining such sets needs more information about the target concept, it is natural to use user-tagged correct expansions as well for the refinement.

In this paper, we refer to the problem of fine-grained concepts being erroneously extended as *semantic spread*. We show that a rich feature representation of the target concept, coupled with appropriate weighting of features, is necessary for reducing *semantic spread* when refining fine-grained sets. We propose an algorithm using relevance feedback, including both positive and negative user feedback, for set refinement. By expanding the set of features and weighting them appropriately, our algorithm is able to retrieve more related instances and provide better ranking. Experimental results show that it improves the quality of fine-grained sets significantly.

2 Related work

There is a large body of research on growing named entity sets from a handful of seeds. Some are pattern-based algorithms. Sarmento et al. (2007) uses explicit patterns, e.g. *"...NE_a, NE_b and NE_c..."*, to find named entities of the same class. Pasca (2004) uses the pattern *<[StartOfSent] X [such as|including] N [and|,|.]>* (Hearst 1992) to find instances and their class labels from web logs. Some are based on distributional similarity. The distributional hypothesis states that similar terms tend to appear with similar contexts (Harris 1954). For example, Pasca (2007) extracts templates (prefixes and suffixes around seeds) from search engine query logs as features, and then ranks new instances by their similarity with the seeds in the vector space of pattern features for growing sets. Their method outperforms methods based on handcrafted patterns (Pasca 2004) but requires extensive query logs to tolerate noisy queries. Calculating the similarity matrix between all pairs of named entities is expensive. Pantel et.al (2009) proposed a web-scale parallel implementation on the MapReduce distributed computing framework.

Observing the low quality of expanded sets, Vyas and Pantel (2009) uses negative user feedback for set refinement. They propose the Similarity Method (SIM) and Feature Modification Method (FMM), to refine entity sets by removing expansions which are similar to user-tagged errors, and removing features related to the erroneous sense from the centroid of the seed set for better ranking, respectively. Their algorithms rely on two assumptions 1) most expansion errors are caused by ambiguous seeds, and 2) entities which are similar in one sense are usually not similar in their other senses. They show average performance gain over a few sets. Vyas et al. (2009) studied the problem from the other side by selecting better seeds. They proposed three metrics and three corresponding algorithms to guide editors to choose better seeds. All three algorithms outperform the baseline.

3 Similarity modeling revisited

Given a set of candidate named entities represented by vectors of features, the goal of set refinement is to find a subset of entities which are similar to the target concept, based on a certain similarity metric (Cosine, Dice, etc). The concept is usually approximated with a set of seed instances. A previous feature pruning technique (Vyas and Pantel 2009) aims at reducing semantic drift introduced by ambiguous seeds.

We're particularly interested in fine-grained classes since they're intrinsically hard to expand because of the crude representation from the limited number of seeds. In practice, we observed, when expanding fine-grained classes, that *semantic spread* instead of semantic drift (McIntosh 2010) severely affects expansion quality. By *semantic spread* we mean a situation where an initial concept, represented by its member entities, changes in the course of entity set expansion into a broader concept which subsumes the original concept.

Semantic spread is usually introduced when erroneous instances, which belong to a more general concept, are incorrectly included during the set expansion process. For example, when using Google Sets (labs.google.com/sets) to expand *National Capitals*, we found a highly ranked error *New York*. By checking with our distributional thesaurus extracted from 37 years' newspaper, we notice the following features: *prep_in(embassy, *)* [1], *nn(*, capital), nn (*, president)*. These are indicators of capital cities. However, as the financial "capital" and a politically important city, *New York* shares lots of informative features with the *National Capitals* concept. Therefore, we need more sophisticated techniques for the refinement process for fine-grained concepts.

4 Refine fine-grained classes with user feedback

User feedback is a valuable resource for learning the target concept. We propose to use both positive and negative feedback to learn a rich set of features for the target concept while weighting them appropriately. Our algorithm chooses informative instances to query the user, uses positive feedback for expanding the feature set, and negative feedback for feature weight adjustment.

Relevance feedback (Harman 1992) is widely applied to improve search engine performance by modifying queries based on user feedback. Various techniques are proposed for both the vector space model and probabilistic model. Since set refinement is done in the vector space of features, we only consider techniques for the vector space model. To refine entity sets, the centroid of all vectors of seeds is used as a query for retrieving related named entities from the candidate pool. Observing that errors are usually caused by incorrect or overweighted features of seeds, we propose to incorporate user feedback for set refinement with a variant of the Rocchio algorithm (Rocchio 1971). The new centroid is calculated as follows:

$$Centroid = \frac{\sum_{I \in S \cup P} I}{|S \cup P|} - \gamma \cdot \frac{\sum_{C_N \in N} C_N}{|N|}$$

where I is an entity that is a member of seed set S or the set of user-tagged positive entities P, and C_N is a member of the set of user-tagged negative entities N. γ is the parameter penalizing features of irrelevant entities. This method does feature set expansion and iterative adjustment of feature weights for the centroid. It adds features from informative instances back into the centroid

[1] Syntactic context is used in our experiment. For the format of dependencies, please refer to the Stanford typed dependencies manual.

and penalizes inaccurate features based on user-tagged errors, thus modifying the centroid to be a better representation of the target class.

4.1 Query strategy

To be practical, we should ask the user to review as few instances as possible, while obtaining as much information as possible. Observing that 1) top-ranked instances are likely to be positive 2) random instances of a fine-grained class usually contain relatively few features with non-zero weight, thus not providing much information for approaching the target concept, our procedure selects at each iteration the *n* instances most similar to the centroid and presents them to the user in descending order of their number of features with non-zero weight (the user will review higher-dimension ones first). This ranking strategy prefers more representative instances with more features (Shen et al., 2004). The user is asked to pick the first positive instance.

A similar idea applies to negative instance finding. We use co-testing (Muslea et al., 2006) to construct two ranking-based classifiers on randomly split views of the feature space. Instances are ranked by their similarity to the centroid. The classifiers classify instances which ranked higher than the golden set size as correct, and classify others as incorrect. We select *n* contention instances – instances identified as correct expansions by one of the classifiers and incorrect by the other. These instances are more ambiguous and likely to be negative. Instances are also presented to the user in descending order of number of features with non-zero weight. Coupled with the strategy for positive instance finding, it helps to reweight a rich set of features.

Since we asked the user to review instances that are most likely to be positive and negative, and these instances are presented to the user in sequence, the user only has to review very few examples to find a positive and a negative instance in each iteration. In practice we set *n*=10. We observed that around 85% of the time the user only has to review 1 instance to find a correct one, and over 90% of the time has to review 3 or fewer instances to find a negative one.

5 Experiment

Corpus: we used 37 years newspaper corpus[2] which is dependency parsed with the Stanford Parser[3] and has all named entities tagged with Jet[4] NE tagger (we didn't use the NE tags reported by the tagger but only the fact that it is a name). We use syntactic context, which is the grammatical relation in conjunction with the words as feature, and we replace the word in the candidate NE with *. Both syntactic contexts in which the candidate entities are the heads and contexts in which the candidate entities are the dependents are used. The feature set is created from syntactic contexts of all entities tagged in the corpus. An example common feature for class National Capital is prep_in(ministry, *). We remove features in which the dependent is a stop word, and remove a limited number of less useful dependency types such as numerical modifier and determiner. We use pointwise mutual information (PMI) to weight features for entities, and cosine as the similarity measure between the centroid of the seeds and candidate instances. PMI scores are generated from the newspaper corpus statistics. Candidates are then ranked by similarity. We construct each named entity candidate pool by including similar instances with cosine score greater than 0.05 with the centroid of the corresponding golden set. This ensures that each candidate pool contains tens of thousands of elements so that it contains all similar instances with high probability.

Golden sets[5]: Several golden sets are prepared by hand. We start from lists from Wikipedia, and then manually refine the sets[6] by removing incorrect instances and adding correct instances found as distributionally-similar instances from the corpus. The criteria for choosing the lists is 1) our corpus covers most elements of the list, 2) the list represents a fine-grained concept, 3) it contains hundreds of elements for reasons of fairness, since we don't want the added positive examples themselves to overshadow other aspects of the evaluated algorithms. Based on these criteria, we chose three lists: *National Capitals, IT companies*[7] and *New York City (NYC) neighborhoods*. All three sets have more than 200 elements. User feedback is simulated by checking membership in the golden set. Since existing

[2] It contains news articles from: TDT5, NYT(94-00), APW(98-00), XINHUA(96-00), WSJ(94-96), LATWP(94-97), REUFF(94-96), REUTE(94-96), and WSJSF(87-94). It contains roughly 65 million sentences and 1.3 billion tokens.

[3] http://nlp.stanford.edu/software/lex-parser.shtml

[4] http://cs.nyu.edu/grishman/jet/license.html

[5] Golden sets are available for download at http://www.cs.nyu.edu/~min/goldset_37news.tgz

[6] Manually checking indicates the golden sets are complete with high probability.

[7] Set contains both software and hardware companies

golden sets such as the sets from Vyas and Pantel (2009) are not designed specifically for evaluating refinement on fine-grained concepts and they are quite small for evaluating positive feedback (with less than 70 elements after removing low frequency ones in our corpus), we decided to construct our own.

Algorithms evaluated: The following algorithms are applied for iteratively updating the centroid using user-tagged examples: 1) **baseline algorithm (BS),** an algorithm adding the correct example most similar to the centroid as a new seed for each iteration; this simulates using the user-tagged first positive example to assist refinement, 2) **RF-P**, relevance feedback algorithm using only positive feedback by adding one informative instance (selected using the method described in section 4.1) into seed set, 3) **FMM** (Vyas and Pantel, 2009) which uses the first user-tagged negative example for feature pruning in each iteration. 4) **RF-N**, relevance feedback algorithm using only negative feedback (selected using the method described in section 4.1), 5) Relevance feedback (**RF-all**) using both positive and negative user feedback selected using methods from Section 4.1. We use 6 seeds for all experiments, and set γ=0.25 for all RF experiments.

For each algorithm, we evaluate the results after each iteration as follows: we calculate a centroid feature vector and then rank all candidates based on their similarity to the centroid. We add sufficient top-ranked candidates to the seed and user-tagged positive items to form a set equal in size to the golden set. This set, the *refined set*, is then compared to the golden set. The following tables show a commonly reported metric, average R-precision[8] of 40 runs starting with randomly picked initial seeds (The first column shows the number of iterations.):

	BS	RF-P	FMM	RF-N	RF-all
2	0.258	0.298	0.253	0.246	0.286
4	0.260	0.317	0.250	0.251	0.316
6	0.260	0.323	0.244	0.255	0.332
8	0.260	0.325	0.243	0.255	0.342
10	0.265	0.325	0.245	0.256	0.343

Table 1. Performance on class *national capitals*

	BS	RF-P	FMM	RF-N	RF-all
2	0.303	0.340	0.301	0.303	0.319
4	0.312	0.403	0.303	0.311	0.406
6	0.317	0.432	0.312	0.312	0.451
8	0.323	0.442	0.312	0.314	0.467
10	0.327	0.445	0.306	0.314	0.481

Table 2. Performance on class *IT companies*

	BS	RF-P	FMM	RF-N	RF-all
2	0.168	0.190	0.159	0.161	0.191
4	0.179	0.217	0.164	0.163	0.232
6	0.189	0.235	0.163	0.166	0.249
8	0.198	0.243	0.166	0.176	0.259
10	0.206	0.248	0.169	0.181	0.262

Table 3. Performance on class *NYC neighborhoods*

Results show that RF-P outperforms the baseline algorithm by using positive examples with rich contexts rather than the first positive example for each iteration. The baseline algorithm shows small improvement over 10 iterations. This shows that simply adding the example which is most similar to the centroid is not very helpful. Comparing R-precision gain between RF-P and the baseline suggests that selecting informative examples is critical for refining fine-grained sets. By enriching the feature set of the centroid, RF-P is able to retrieve instances with a limited number of features overlapping the original centroid. RF-N outperforms FMM since it only reweights (penalizes some weights) but doesn't prune out intersection features between user-tagged errors and the centroid. This flexibility avoids over-penalizing weak but informative features of the intended concept. For FMM, we observe a small performance gain with successsive iterations over *IT companies* and *NYC neighborhoods* but a performance decrease for *National Capitals*. Inspection of results shows that FMM tends to retrieve more *capital cities* for small geographical regions because of removal of weak features for informative sense such as *Major Cities*.

Combining RF-P and RF-N, RF-all uses both positive informative examples and negative informative examples to expand feature sets of the centroid and weight them appropriately, thus achieving the most performance gain. RF-N by itself doesn't improve performance significantly. Comparing RF-all with RF-P, using informative negative examples helps to improve performance substantially because only when both informative positive examples and informative negative examples are used can we learn a significantly large set of features and appropriate weights for them.

[8] Precision at the rank of golden set size

We also implemented a few methods combining positive feedback and FMM, and didn't observe encouraging performance. RF-all also has the highest Average Precision (AP) for all sets, thus showing that it provides better ranking over candidates. Due to space limitations, tables of AP are not included. The quality of the top ranked elements with RF-all can be seen in the precision at rank 50 for the three sets: 84.6%, 81.6%, and 71.7%.

6 Conclusion and Future work

We propose an algorithm using both positive and negative user feedback to reduce semantic spread for fine-grained entity set refinement. Our experimental results show performance improvement over baseline and existing solutions.

Our next step is to investigate feature clustering techniques since we observe that data sparseness severely affects set refinement.

Acknowledgments

We are grateful to the anonymous reviewers for their valuable comments. We would like to also thank Prof. Satoshi Sekine and Ang Sun for their helpful discussion and comments on an early draft of this paper.

References

Razvan Bunescu and Raymond J. Mooney. 2004. Collective Information Extraction with Relational Markov Networks. In Proceedings of ACL-04.

Oren Etzioni, Michael Cafarella, Doug Downey, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S. Weld and Alexander Yates. 2005. Unsupervised named-entity extraction from the Web: An Experimental Study. In Artificial Intelligence, 165(1):91-134.

Donna Harman. 1992. Relevance feedback revisited. In Proceedings of SIGIR-92.

Zellig S. Harris. 1954. Distributional Structure. Word. Vol 10: 146-162.

Marti A. Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In Proceedings of COLING-92.

Tara McIntosh. 2010. Unsupervised discovery of negative categories in lexicon bootstrapping, In Proceedings of EMNLP-10.

Ion Muslea, Steven Minton and Craig A. Knoblock. 2006. Active Learning with Multiple Views, Journal of Artificial Intelligence Research 27: 203-233.

Patrick Pantel, Eric Crestan, Arkady Borkovsky, Ana-Maria Popescu and Vishnu Vyas. 2009. Web-Scale Distributional Similarity and Entity Set Expansion. In Proceedings of EMNLP-09.

Patrick Pantel and Dekang Lin. 2002. Discovering word senses from text. In Proceedings of KDD-02.

Marius Pasca. 2004. Acquisition of Categorized Named Entities for Web Search, In Proceedings of CIKM-04.

Marius Pasca. 2007. Weakly-supervised discovery of named entities using web search queries. In Proceedings of CIKM-07.

Marco Pennacchiotti and Patrick Pantel. 2009. Entity Extraction via Ensemble Semantics. In Proceedings of EMNLP-09.

J. J. Rocchio. 1971. Relevance feedback in information retrieval. The SMART Retrieval System: Experiments in Automatic Document Processing: 313-323.

Luis Sarmento, Valentin Jijkoun, Maarten de Rijke and Eugenio Oliveira. 2007. "More like these": growing entity classes from seeds. In Proceedings of CIKM-07.

Dan Shen, Jie Zhang, Jian Su, Guodong Zhou and Chew-Lim Tan. 2004. Multi-criteria-based active learning for named entity recognition. In Proceedings of ACL-04.

David Vickrey, Oscar Kipersztok and Daphne Koller. 2010. An Active Learning Approach to Finding Related Terms. In Proceedings of ACL-10.

Vishnu Vyas and Patrick Pantel. 2009. Semi-Automatic Entity Set Refinement. In Proceedings of NAACL/HLT-09.

Vishnu Vyas, Patrick Pantel and Eric Crestan. 2009, Helping Editors Choose Better Seed Sets for Entity Set Expansion, In Proceedings of CIKM-09.

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