Unsupervised Labeling of Noun Clusters

Theresa Jickels and Grzegorz Kondrak

Department of Computing Science,
University of Alberta,
Edmonton, AB, T6G 2E8, Canada
{theresa,kondrak}@cs.ualberta.ca
http://www.cs.ualberta.ca/~kondrak

Abstract. Semantic knowledge is important in many areas of natural language processing. We propose a new unsupervised learning algorithm to annotate groups of nouns with hypernym labels. Several variations of the algorithm are presented, including a method that utilizes semantic information from WordNet. The algorithm's results are compared against an independently-developed labeling method. The evaluation is performed using labels assigned to noun clusters by several participants of a specially designed human study.

1 Introduction

Ontologies, also known as semantic networks and lexical databases, are important resources in natural language processing. These resources all encode a specific type of semantic knowledge. Constructing them manually is expensive and time-consuming. In this paper, we focus on the task of automatic construction of ontologies from large corpora of texts.

A well-known example of a hand-built ontology is WordNet [8]. Lexical entries in WordNet are organized into comprehensive networks of synonym sets (*synsets*). Words that have more than one meaning (polysemous words) may participate in several different synsets. The synsets are linked by various *lexical relations*. The principal lexical relation in WordNet is *hypernymy*, the *is-a* relation between nouns. For example, *bird* is a hypernym of *robin*. Hyponymy is the inverse of hypernymy. The hypernymy/hyponymy links form the backbone of the noun hierarchy. They link each synset to its immediately more general and more specific synsets. A chain of hypernymy links can be traversed from each synset to one of the eleven abstract concepts that are at the top of the hierarchy.

The labeling of groups of nouns can be seen as automatic identification of hyponymy relations. We do not differentiate between instances and hyponyms. For example, the set of terms (*table, chair, desk, sofa, dresser, bookcase*) can be labeled with the term *furniture*. For a group of movie titles such as (*Deep Impact, Armageddon, Godzilla, Titanic, Truman Show*), possible labels include *movie*, or *film*. By providing a label for a set of terms, we implicitly define a number of hypernymy relations between the label and the terms in the set.

We propose an unsupervised algorithm for the purpose of labeling groups of nouns. It addresses a specific subset of the problem of automatic gathering of semantic relations. Since the algorithm is unsupervised, it can produce labels quickly, and without the prohibitive cost of human involvement in annotating training data.

The principal features that we use for labeling noun clusters are dependency relations, which are syntactic relationships between words in a sentence. For example, from the sentence *"Smith, the chairman, is a workaholic"*, we can extract the nominal *subject* relation (*N:subj:N*) between *Smith* and *workaholic*, and the *appositive* relation (*N:appo:N*) between *Smith* and *chairman.* The presence or lack of a minus sign indicates whether the head of the relationship is the right or left noun. For example, the above appositive relation is denoted as a *N:appo:N* relation in the context of the word *Smith*, and a *-N:appo:N* relation in the context of the word *chairman.* We use dependency relations extracted from a corpus by a dependency parser Minipar [6] to build clusters of nouns.

The noun clusters that are labeled by our algorithm are constructed using the *Clustering by Committee* method [10]. The basic idea behind the method is to select a small number of representative terms (a *committee*) that form the core of each cluster. This approach prevents polysemous concepts from interfering with the cluster creation. The algorithm can be summarized as follows. For each element, a small number of its closest neighbors are identified. Next, the algorithm defines as many cluster *committees* as possible, but discards those that are too similar to other committees. Finally, each element in the data is assigned to its most similar cluster.

The applications of noun labeling include question answering and named entity classification. In generating an answer to a question such as "Who was the first prime minister of Canada?", it may be very helpful to know that there exist a hypernymy relationship between *Sir John A. Macdonald* and *prime minister.* Given the named entity *Carnegie Mellon*, it is more useful to classify it specifically as a *university* than as an *organization.*

The organization of this paper is as follows. Section 2 is devoted to the related work. Section 3 introduces our approach to unsupervised labeling of noun clusters. Section 4 contains description of the experiments and their results. Section 5 concludes the paper.

2 Related Work

In this section, we discuss several techniques that have been used in the automatic gathering of semantic relations, including the method of Pantel and Ravichandran [9] which addresses virtually the same task as this paper.

Chodorow et al. [4] extract hypernyms from the definitions of nouns in machine readable dictionaries with the goal of producing a semantic network. Typically, the head of the definition phrase is taken to be the hypernym of the noun. For example, if a definition of *golden retriever* is *a golden-haired dog*, then *dog* would be identified as the hypernym of *golden retriever.* Heuristic rules are used to extract the head of the definition, and then the information is worked into a hierarchical structure, with substantial human involvement.

A later paper by Ide and Veronis [5], examines the progress made with these methods, with the conclusion that a dictionary by itself is not enough to automatically produce the correct results. This conclusion is attributed to the varied structure in the dictionary definitions, missing information, and the fact that different dictionaries produce markedly different hierarchies.

Hearst [3] extracts hypernymy relations directly from text by the means of syntactic patterns. For instance, a phrase such as *dolls, tops and other toys* yields *toy* as a hypernym of *doll*. She proposed six different patterns, but used only one of them for the evaluation against WordNet. Out of 152 hypernymy relations extracted from an encyclopedia using the pattern, 106 had both terms present in WordNet, including 61 in existing hypernymy relations.

Fleischman and Hovy [2] propose a feature-based system for classifying names into a few categories, such as *politician*, *businessman* and *entertainer*. Features like previous and following *n*-grams, topic features, and WordNet features were used to train several classifiers on a semi-automatically produced training set. The classifiers included decision trees, neural networks, SVM, and a Naive Bayes classifier. The best results were obtained with the decision tree method, which achieved 70.4% accuracy.

Caraballo [1] aims at automatically creating a hierarchical semantic lexicon from a body of text. The hierarchy consists of noun clusters grouped under their hypernyms. First, groups of similar nouns are extracted from a few syntactic relationships, such as conjunctions and appositives. The hypernyms are added during the latter part of the construction. Clustering the nouns with a bottom-up method produces the general hierarchy, as similar nouns are placed under the same parent in a binary tree. The percentage of correct hypernyms ranged from 33% under a strict evaluation, to 60% under a lenient evaluation.

Independently of our work, Pantel and Ravichandran [9] proposed a method for automatically labeling the clusters produced by the *Clustering by Committee* method. Their method involves three stages. Stage 1 requires calculating two vectors for each word in the clusters: a frequency count vector, and a mutual information vector, which is discounted to reduce the effect of data sparseness. Stage 2 produces a committee for each cluster, with the goal of isolating the most representative members of the cluster. Stage 3 forms a signature for each cluster, derived from the feature vectors of the committee members. The candidate labels are selected by considering a set of four dependency relations, which the authors identified as the most important. The relative scores for possible labels are calculated using the pointwise mutual information values between the labels and the dependency relations.

Pantel and Ravichandran tested their method on 1432 noun clusters extracted from the Aquaint corpus. The results of automatic labeling were evaluated by human judges. The top answer was judged correct 72% of the time. Among the top five answers, a correct label was present in 85.6% of the cases. No name was produced for 21 (1.5%) of the clusters.

3 Cluster Labeling

Our new algorithm for cluster labeling is an unsupervised learning method, which removes the need for the time-consuming manual annotation of the data. The idea is to utilize the labels that are frequently present in the aggregate data.

Consider Table 1, which shows a cluster containing horse-race names taken from the dependency data. We refer to terms that occur in the dependency relations as *feature words*. A term is said to be a feature word of a feature if it is listed as an instance of that feature. It turns out that good labels for clusters, such as *race* in this case, are often present among feature words in their dependency data. However, most of the feature words are not appropriate as labels, leaving the problem of how to identify the good labels.

Cluster	Features	Feature words
Preakness Stakes	**-N:before:N** 23	
Preakness		day 19
Belmont Stakes		start 2
Travers		race 2
Santa Anita Derby	**-N:subj:N** 80	
Kentucky Derby		race 51
Florida Derby		run 7
		goal 7
		event 8
		victory 3
		start 2
		history 2
	⋮	⋮

Table 1. Sample dependency data for a noun group

Our algorithm learns weights for each feature in order to pick out the good labels from the rest of the data. It is not merely a case of choosing the most common feature or the most frequently occurring feature word. Our solution is to assign variable weights to features, which reflect their relative importance with respect to the likelihood of containing appropriate cluster labels. We refer to these weights as *feature scores*. Each iteration of the algorithm redistributes the feature scores to better represent their values in relation to possible labels. As a feature score increases (or decreases) through iterations, so do the label scores of the feature words associated with that feature. As a result, the feature scores of important features are amplified by the presence of good labels in that feature, while unimportant features are given low scores. Consequently, the good labels are more likely to be present in features with high scores than in features with low scores.

3.1 The Algorithm

The input to our unsupervised labeling algorithm is the dependency data of the clusters, and the list of features. The dependency data are extracted with the LaTaT package [7]. The output of the training process is a feature score for each feature f in the feature set F. The labeling is performed on the basis of the feature scores computed by the algorithm. The final output is a ranked list of possible labels for each cluster.

```
1) For each f in F do
2)         $FS(f) := 1$
   end for
3) Repeat
4)         For each f in F do
5)                 $FS'(f) := 0$
           end for
6)         For each Cluster in the training set do
7)                 For each feature word a in the cluster do
8)                         $LS(a) := \sum_{\substack{f \in F \\ C(f,a)}} FS(f)$
                   end for
9)                 For each f in F do
10)                        $FS'(f) := FS'(f) + \sum_{C(f,a)} LS(a)$
                   end for
           end for
11)        $\delta := \sum_{f \in F} |FS'(f)b - FS(f)|$
12)        For each f in F do
13)                $FS(f) := \frac{|F|}{\sum_{f \in F} FS'(f)} FS'(f)$
           end for
14) Until $\delta = 0$
15) For each Cluster in the testing set do
16)                Label:= $\underset{a}{Max}\{LS(a) := \sum_{\substack{f \in F \\ C(f,a)}} FS(f)\}$
17)                Print Label
    end for
```

Fig. 1. The cluster labeling algorithm.

The algorithm is shown in Figure 1. First, the feature scores *FS(f)* are initialized (lines 1–2). The main loop (lines 3–14) encompasses the learning process. During an iteration, the label scores *LS(a)* for each feature word are calculated by summing the scores of the features in which the feature word occurs (lines 7–8). (The predicate $C(f, a)$ is true if and only if the feature f has a as a feature word.) For each feature, a temporary score (*FS'(f)*) accumulates the label scores of each feature word in that feature (lines 9–10). After each cluster has had the values calculated, the feature scores of each feature are replaced with the normalized temporary score (lines 12–13). The iterative learning process continues until the feature scores converge. In principle, a threshold might be set for convergence, but in practice the values for the feature scores have always converged, so the algorithm stops when the values of the feature scores stop changing. The feature word with the maximum sum of feature scores is selected as the final cluster label (lines 15-17).

3.2 Variations on the Basic Method

The algorithm described in the previous section to automatically assigns weights to all features for the purpose of selecting labels. We also experimented with methods that use fixed subsets of features with equal weights. The baseline approach is to use all the available features. The second set, referred to as Answer Distribution Features (ADF), contains four features and their complements that we identified as the most important on the basis of the analysis of our development set. The final set, referred to as PRF, is composed of the four features selected by Pantel and Ravichandran [9] in their labeling method. The ADF and PRF feature sets are shown in Table 2.

The ADF subset:	-N:appo:N, -N:subj:N, -N:conj:N, -N:nn:N, N:appo:N, N:subj:N, N:conj:N, N:nn:N
The PRF subset:	N:appo:N, -N:subj:N, -N:like:N, -N:such as:N

Table 2. The feature subsets used in the experiments.

WordNet is a useful source of hypernymy relations.[1] For all terms in a given cluster, we recursively collected all WordNet hypernyms up to the top of the hierarchy. Then, we intersected the set of hypernyms with the output of our learning algorithm. We refer to this combined approach as the intersection method.

We also experimented with two other methods of improving the accuracy of the algorithm: filtering out low-frequency feature words, and considering only clusters containing mostly names. However, in both cases, the accuracy gains on the development set did not translate into substantial improvements on the test set.

4 Experimental Setup and Results

The entire data consisted of 1088 clusters. We used the first one hundred clusters as the development set. The test set was constructed by randomly selecting one hundred clusters form the remaining data. The training set consisted of all available clusters, except the ones included in the test set.

[1] For accessing WordNet, version 2.0, we used the QueryData interface [11].

4.1 Evaluation

We conducted a study in order to produce the answer key to our test set, and to examine human performance on the task of classifying clusters. Eight participants were asked to come up with one or more labels for all clusters in the test set. without accessing any external resources nor conferring with other participants.

For example, for the set of terms {*din, noise, roar, sound, rumble, sonic boom, explosion, buzz, hum, noise level, cacophony, echo, drone, clatter, flash, loud noise, thunder, gunfire, whir*}, the participants of the study provided the following labels: *sound* (4 times), *noise* (2), *audio* (1), *decibel* (1), *level of noise* (1), *noise pollution* (1), and *can hear these* (1). For comparison, the algorithm described in Section 3 produces the following labels and scores: *sound* 113.4, *noise* 106.5, *light* 74.9, *rework* 70.6, *echo* 68.7, *all* 64.8, *scream* 63.4, *hiss* 63.4, *smoke* 63.4, *band* 63.4.

Several of the participants indicated that the labeling was quite difficult. The average number of labels given by a participant was .96 per cluster. Agreement between labels from different testers was at 42.8%. This was calculated as the average number of participants who agree on a label that is the result for the highest number of participants. In some instances the most common response was "no label". There were no instances where all the participants assigned the same label to a cluster, but there were several where no two participants gave the same label.[2]

The performance of various methods was measured by computing precision and recall against the labels provided by the participants of the study. A cluster label was considered correct if it was proposed by at least one of the participants. Precision was the percentage of generated labels that were correct. Recall was calculated as the percentage of clusters that were assigned at least one correct label. The maximum possible recall was 86%, because none of the human-proposed labels occurs in the dependency data for 14% of the clusters.

Our evaluation method is different from the one adopted by Pantel and Ravichandran [9]. They presented human evaluators with a list of possible labels that included the top five labels generated by their algorithm, one label proposed by an independent annotator, and up to five names extracted from WordNet. The evaluators were asked to judge the labels as correct, partially correct, or incorrect. In contrast, our evaluation approach did not restrict the choice of labels to a fixed list. In order to perform a fair comparison, we asked the authors of the other labeling algorithm to run it on our test set. The results are discussed in the following section.

4.2 The Test Set Results

Figure 2 shows the results of various methods on the test set. In most cases. the graph has data points corresponding to recall and precision for the following

[2] The complete cluster data and detailed results of the study are publicly available at `http://www.cs.ualberta.ca/~kondrak`

numbers of possible labels returned: 1, 2, 4, 8, 10, 14, 18, 20. The intersection between WordNet and our iterative training method is represented by data points for 1 and 5 labels returned, reflecting a small size of the majority of intersection sets. The results of Pantel and Ravichandran have data points for 1, 2, and 3 labels returned. The pure WordNet method, which produces an unordered set of hypernyms, is represented by a single data point.

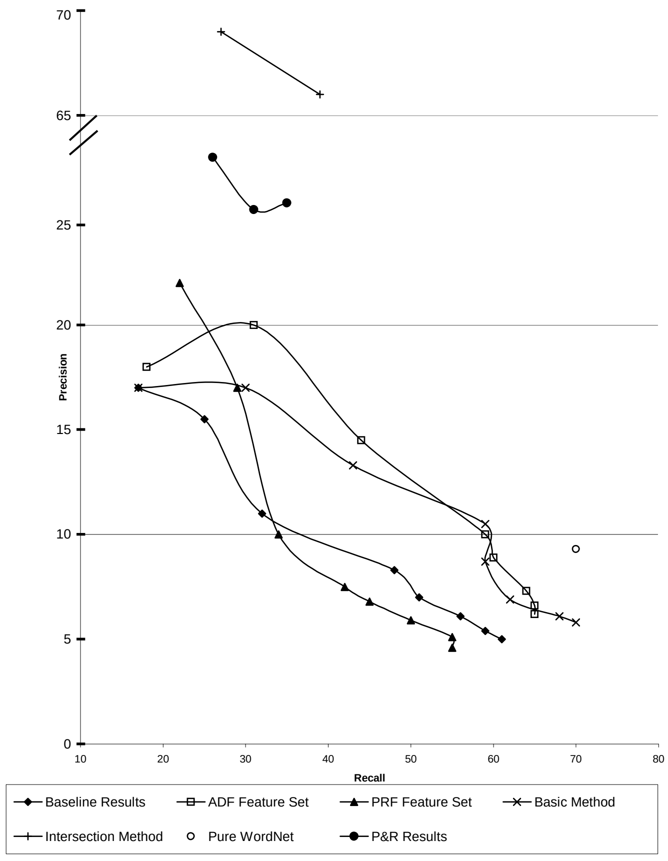

Fig. 2. Results on the test set.

The algorithm presented in Section 3 performs substantially better than the baseline and the PRF feature set, except when only one label is returned. However, it does not do better than the ADF Feature Set. The pure WordNet method yields a high recall rate of 70%, combined with a low precision rate of 9.3%. Compared to our results, the algorithm of Pantel and Ravichandran, which makes extensive use of word mutual information vectors, appears to achieve similar recall, but with higher precision. However, our intersection method yields much higher precision at a comparable recall level.

4.3 Discussion

The overall accuracy of most of the approaches tested on the Test Set is quite low. The labels assigned by humans were often compound phrases, while the automated methods generate mostly single word labels. In addition, many of the human labels are too general, and some are plain wrong. While most names are easily labeled as referring to people by the human participants, the specific professions or roles that those people have in common are not so easily identified. There were frequent cases of an algorithmic method identifying more specific labels than the human participants. For example, in the case of a cluster including names *Tim Couch*, *Peyton Manning* and *Doug Pederson*, only one person gave a relatively specific label: *sportsman*. The remaining participants provided general labels such as *people*, an inaccurate one (*football*), or no label at all. In contrast, the top label generated by our algorithm was *quarterback*.

5 Conclusion

We proposed a new unsupervised learning algorithm for labeling clusters of nouns with hypernyms. The algorithm uses only the dependency information for the clusters, and does not require annotated data. We investigated several variations of the algorithm, including an intersection method that combined the results of the algorithm with information obtained by WordNet. For the purpose of an unbiased evaluation of various methods and a comparison with an independently proposed alternative approach, we conducted a human study that included several participants. The results of the experiments indicate that our algorithm does substantially better than the baseline, and that the combination of the algorithm with WordNet achieves over 65% precision.

Acknowledgments

Thanks to Dekang Lin for sharing the dependency data and the LaTaT package, to Patrick Pantel for running his system on our test set, and to the participants of the human study for devoting their valuable time to labeling the noun clusters. This research was supported by the Natural Sciences and Engineering Research Council of Canada.

References

1. Sharon A. Caraballo. 1999. Automatic construction of a hypernym-labeled noun hierarchy from text. In *Proceeding of the 37th annual meeting of the Association for Computational Linguistics (ACL-99)*, pages 120–126.
2. M. Fleischman and E. Hovy. 2002. Fine grained classification of named entities. In *19th International Conference on Computational Linguistics (COLING)*.
3. Marti A. Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *14th International Conference on Computational Linguistics*.

4. Martin S. Chodorow; Roy J. Byrd; George E. Heidorn. 1985. Extracting semantic hierarchies from a large on-line dictionary. In *Proceeding of the 23rd annual meeting of the Association for Computational Linguistics (ACL-85)*, pages 299–304.
5. Nancy Ide and Jean Veronis. 1994. *Knowledge Extraction from Machine-Readable Dicitionaries: An Evaluation*.
6. Dekang Lin. 1998. Dependency-based evaluation of MINIPAR. In *Workshop on the Evaluation of Parsing Systems*.
7. Dekang Lin. 2001. Latat: Language and text analysis tools. In *Proceedings of Human Language Technology Conference*, pages 222–227.
8. George A. Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine J. Miller. 1993. Introduction to wordnet: An on-line lexical database.
9. Patrick Pantel and Deepak Ravichandran. 2004. Automatically labeling semantic classes. In *Proceedings of Human Language Technology / North American chapter of the Association for Computational Linguistics (HLT/NAACL-04)*, pages 321–328.
10. Patrick Pantel. 2003. *Clustering by Committee*. Department of Computing Science, University of Alberta.
11. Jason Rennie. 2000. Wordnet::querydata: a Perl module for accessing the WordNet database. http://www.ai.mit.edu/people/jrennie/WordNet.