Automatic construction of a hypernym-labeled noun hierarchy from text

Sharon A. Caraballo
Dept. of Computer Science
Brown University
Providence, RI 02912
`sc@cs.brown.edu`

Abstract

Previous work has shown that automatic methods can be used in building semantic lexicons. This work goes a step further by automatically creating not just clusters of related words, but a hierarchy of nouns and their hypernyms, akin to the hand-built hierarchy in WordNet.

1 Introduction

The purpose of this work is to build something like the hypernym-labeled noun hierarchy of WordNet (Fellbaum, 1998) automatically from text using no other lexical resources. WordNet has been an important research tool, but it is insufficient for domain-specific text, such as that encountered in the MUCs (Message Understanding Conferences). Our work develops a labeled hierarchy based on a text corpus.

In this project, nouns are clustered into a hierarchy using data on conjunctions and appositives appearing in the Wall Street Journal. The internal nodes of the resulting tree are then labeled with hypernyms for the nouns clustered underneath them, also based on data extracted from the Wall Street Journal. The resulting hierarchy is evaluated by human judges, and future research directions are discussed.

2 Building the noun hierarchy

The first stage in constructing our hierarchy is to build an unlabeled hierarchy of nouns using bottom-up clustering methods (see, e.g., Brown et al. (1992)). Nouns are clustered based on conjunction and appositive data collected from the Wall Street Journal corpus. Some of the data comes from the parsed files 2-21 of the Wall Street Journal Penn Treebank corpus (Marcus et al., 1993), and additional parsed text was obtained by parsing the 1987 Wall Street Journal text using the parser described in Charniak et al. (1998).

From this parsed text, we identified all conjunctions of noun phrases (e.g., "executive vice-president and treasurer" or "scientific equipment, apparatus and disposables") and all appositives (e.g., "James H. Rosenfield, a former CBS Inc. executive" or "Boeing, a defense contractor"). The idea here is that nouns in conjunctions or appositives tend to be semantically related, as discussed in Riloff and Shepherd (1997) and Roark and Charniak (1998). Taking the head words of each NP and stemming them results in data for about 50,000 distinct nouns.

A vector is created for each noun containing counts for how many times each other noun appears in a conjunction or appositive with it. We can then measure the similarity of the vectors for two nouns by computing the cosine of the angle between these vectors, as

$$\cos(\mathbf{v}, \mathbf{w}) = \frac{\mathbf{v} \cdot \mathbf{w}}{|\mathbf{v}|\,|\mathbf{w}|}.$$

To compare the similarity of two *groups* of nouns, we define similarity as the average of the cosines between each pair of nouns made up of one noun from each of the two groups.

$$sim(A, B) = \frac{\sum_{\mathbf{v},\mathbf{w}} \cos(\mathbf{v}, \mathbf{w})}{size(A)size(B)}$$

where $\mathbf{v}$ ranges over all vectors for nouns in group A, $\mathbf{w}$ ranges over the vectors for group B, and $size(x)$ represents the number of nouns which are descendants of node x.

We want to create a tree of all of the nouns in this data using standard bottom-up clustering techniques as follows: Put each noun into its own node. Compute the similarity between each pair of nodes using the cosine method. Find the two most similar nouns and combine them by giving them a common parent (and removing the child nodes from future consideration). We can then compute the new node's similarity to each other node by computing a weighted average of the similarities between each of its children and the other node.

In other words, assuming nodes A and B have been combined under a new parent C, the similarity between C and any other node i can be computed as

$$sim(C,i) = \frac{sim(A,i)size(A) + sim(B,i)size(B)}{size(A) + size(B)}$$

Once again, we combine the two most similar nodes under a common parent. Repeat until all nouns have been placed under a common ancestor.

Nouns which have a cosine of 0 with every other noun are not included in the final tree.

In practice, we cannot follow exactly that algorithm, because maintaining a list of the cosines between every pair of nodes requires a tremendous amount of memory. With 50,000 nouns, we would initially require a 50,000 x 50,000 array of values (or a triangular array of about half this size). With our current hardware, the largest array we can comfortably handle is about 100 times smaller; that is, we can build a tree starting from approximately 5,000 nouns.

The way we handled this limitation is to process the nouns in batches. Initially 5,000 nouns are read in. We cluster these until we have 2,500 nodes. Then 2,500 more nouns are read in, to bring the total to 5,000 again, and once again we cluster until 2,500 nodes remain. This process is repeated until all nouns have been processed.

Since the lowest-frequency nouns are clustered based on very little information and have a greater tendency to be clustered badly, we chose to filter some of these out. By reducing the number of nouns to be read, a much nicer structure is obtained. We now only consider nouns with a vector of length at least 2.

There are approximately 20,000 nouns as the leaves in our final binary tree structure. Our next step is to try to label each of the internal nodes with a hypernym describing its descendant nouns.

3 Assigning hypernyms

Following WordNet, a word A is said to be a *hypernym* of a word B if native speakers of English accept the sentence "B is a (kind of) A."

To determine possible hypernyms for a particular noun, we use the same parsed text described in the previous section. As suggested in Hearst (1992), we can find some hypernym data in the text by looking for conjunctions involving the word "other", as in "X, Y, and other Zs" (patterns 3 and 4 in Hearst). From this phrase we can extract that Z is likely a hypernym for both X and Y.

This data is extracted from the parsed text, and for each noun we construct a vector of hypernyms, with a value of 1 if a word has been seen as a hypernym for this noun and 0 otherwise. These vectors are associated with the leaves of the binary tree constructed in the previous section.

For each internal node of the tree, we construct a vector of hypernyms by adding together the vectors of its children. We then assign a hypernym to this node by simply choosing the hypernym with the largest value in this vector; that is, the hypernym which appeared with the largest number of the node's descendant nouns. (In case of ties, the hypernyms are ordered arbitrarily.) We also list the second- and third-best hypernyms, to account for cases where a single word does not describe the cluster adequately, or cases where there are a few good hypernyms which tend to alternate, such as "country" and "nation". (There may or may not be any kind of semantic relationship among the hypernyms listed. Because of the method of selecting hypernyms, the hypernyms may be synonyms of each other, have hypernym-hyponym relationships of their own, or be completely unrelated.) If a hypernym has occurred with only one of the descendant nouns, it is not listed as one of the best hypernyms, since we have insufficient evidence that the word could describe this class of nouns. Not every node has sufficient data to be assigned a hypernym.

4 Compressing the tree

The labeled tree constructed in the previous section tends to be extremely redundant. Recall that the tree is binary. In many cases, a group of nouns really do not have an inherent tree structure, for example, a cluster of countries. Although it is possible that a reasonable tree structure could be created with subtrees of, say, European countries, Asian countries, etc., recall that we are using single-word hypernyms. A large binary tree of countries would ideally have "country" (or "nation") as the best hypernym at every level. We would like to combine these subtrees into a single parent labeled "country" or "nation", with each country appearing as a leaf directly beneath this parent. (Obviously, the tree will no longer be binary).

Another type of redundancy can occur when an internal node is unlabeled, meaning a hypernym could not be found to describe its descendant nouns. Since the tree's root is labeled, somewhere above this node there is necessarily a node labeled with a hypernym which applies to its descendant nouns, including those which are a descendant of this node. We want to move this node's children directly under the nearest labeled ancestor.

We compress the tree using the following very simple algorithm: in depth-first order, examine the children of each internal node. If the child is itself an internal node, and it either has no best hypernym or the same three best hypernyms as its parent, delete this child and make its children into children of the parent instead.

Hypernyms	# nouns
vision	22
bank/group/bond	95
conductor	51
problem	151
apparel/clothing/knitwear	113
item/paraphernalia/car	226
felony/charge/activity	109
system	47
official/product/right	88
official/company/product	10,266
product/factor/service	6,056
agency/area	60
event/item	135
animal/group/people	188
country/nation/producer	348
product/item/crop	300
diversion	130
problem/drug/disorder	306
wildlife	35

Table 1: The children of the root node.

5 Results and evaluation

There are 20,014 leaves (nouns) and 654 internal nodes in the final tree (reduced from 20,013 internal nodes in the uncompressed tree). The top-level node in our learned tree is labeled "product/analyst/official". (Recall from the previous discussion that we do not assume any kind of semantic relationship among the hypernyms listed for a particular cluster.) Since these hypernyms are learned from the Wall Street Journal, they are domain-specific labels rather than the more general "thing/person". However, if the hierarchy were to be used for text from the financial domain, these labels may be preferred.

The next level of the hierarchy, the children of the root, is as shown in Table 1. ("Conductor" seems out-of-place on this list; see the next section for discussion.) These numbers do not add up to 20,014 because 1,288 nouns are attached directly to the root, meaning that they couldn't be clustered to any greater level of detail. These tend to be nouns for which little data was available, generally proper nouns (e.g., Reindel, Yaghoubi, Igoe).

To evaluate the hierarchy, 10 internal nodes dominating at least 20 nouns were selected at random. For each of these nodes, we randomly selected 20 of the nouns from the cluster under that node. Three human judges were asked to evaluate for each noun and each of the (up to) three hypernyms listed as "best" for that cluster, whether they were actually in a hyponym-hypernym relation. The judges were students working in natural language processing or computational linguistics at our institution who were not directly involved in the research for this project. 5 "noise" nouns randomly selected from elsewhere in the tree were also added to each cluster without the judges' knowledge to verify that the judges were not overly generous.

Some nouns, especially proper nouns, were not recognized by the judges. For any noun that was not evaluated by at least two judges, we evaluated the noun/hypernym pair by examining the appearances of that noun in the source text and verifying that the hypernym was correct for the predominant sense of the noun.

Table 2 presents the results of this evaluation. The table lists only results for the actual candidate hyponym nouns, not the noise words. The "Hypernym 1" column indicates whether the "best" hypernym was considered correct, while the "Any hypernym" column indicates whether any of the listed hypernyms were accepted. Within those columns, "majority" lists the opinion of the majority of judges, and "any" indicates the hypernyms that were accepted by even one of the judges.

The "Hypernym 1/any" column can be used to compare results to Riloff and Shepherd (1997). For five hand-selected categories, each with a single hypernym, and the 20 nouns their algorithm scored as the best members of each category, at least one judge marked on average about 31% of the nouns as correct. Using randomly-selected categories and randomly-selected category members we achieved 39%.

By the strictest criteria, our algorithm produces correct hyponyms for a randomly-selected hypernym 33% of the time. Roark and Charniak (1998) report that for a hand-selected category, their algorithm generally produces 20% to 40% correct entries.

Furthermore, if we loosen our criteria to consider also the second- and third-best hypernyms, 60% of the nouns evaluated were assigned to at least one correct hypernym according to at least one judge.

The "bank/firm/station" cluster consists largely of investment firms, which were marked as incorrect for "bank", resulting in the poor performance on the Hypernym 1 measures for this cluster. The last cluster in the list, labeled "company", is actually a very good cluster of cities that because of sparse data was assigned a poor hypernym. Some of the suggestions in the following section might correct this problem.

Of the 50 noise words, a few of them were actually rated as correct as well, as shown in Table 3.

This is largely because the noise words were selected truly at random, so that a noise word for the "company" cluster may not have been in that particular cluster but may still have appeared under a "company" hypernym elsewhere in the hierarchy.

6 Discussion and future directions

Future work should benefit greatly by using data on the hypernyms of hypernyms. In our current tree, the best hypernym for the entire tree is "product"; however, many times nodes deeper in the tree are given this label also. For example, we have a cluster including many forms of currency, but because there is little data for these particular words, the only hypernym found was "product". However, the parent of this node has the best hypernym of "currency". If

Three best hypernyms	Hypernym 1		Any hypernym	
	majority	any	majority	any
worker/craftsmen/personnel	13	13	13	13
cost/expense/area	7	10	9	10
cost/operation/problem	6	8	11	17
legislation/measure/proposal	3	5	9	18
benefit/business/factor	2	2	2	5
factor	2	7	2	7
lawyer	14	14	14	14
firm/investor/analyst	13	13	14	14
bank/firm/station	0	0	15	17
company	6	6	6	6
AVERAGE	6.6 / 33.0%	7.8 / 39.0%	9.5 / 47.5%	12.1 / 60.5%

Table 2: The results of the judges' evaluation.

Three best hypernyms	Hypernym 1		Any hypernym	
	majority	any	majority	any
noise words	1 / 2.0%	4 / 8.0%	2 / 4.0%	4 / 8.0%

Table 3: The results of the judges' evaluation of noise words.

we knew that "product" was a hypernym of "currency", we could detect that the parent node's label is more specific and simply absorb the child node into the parent. Furthermore, we may be able to use data on the hypernyms of hypernyms to give better labels to some nodes that are currently labeled simply with the best hypernyms of their subtrees, such as a node labeled "product/analyst" which has two subtrees, one labeled "product" and containing words for things, the other labeled "analyst" and containing names of people. We would like to instead label this node something like "entity". It is not yet clear whether corpus data will provide sufficient data for hypernyms at such a high level of the tree, but depending on the intended application for the hierarchy, this level of generality might not be required.

As noted in the previous section, one major spurious result is a cluster of 51 nouns, mainly people, which is given the hypernym "conductor". The reason for this is that few of the nouns appear with hypernyms, and two of them (Giulini and Ozawa) appear in the same phrase listing conductors, thus giving "conductor" a count of two, sufficient to be listed as the only hypernym for the cluster. It might be useful to have some stricter criterion for hypernyms, say, that they occur with a certain percentage of the nouns below them in the tree. Additional hypernym data would also be helpful in this case, and should be easily obtainable by looking for other patterns in the text as suggested by Hearst (1992).

Because the tree is built in a binary fashion, when, e.g., three clusters should all be distinct children of a common parent, two of them must merge first, giving an artificial intermediate level in the tree. For example, in the current tree a cluster with best hypernym "agency" and one with best hypernym "exchange" (as in "stock exchange") have a parent with two best hypernyms "agency/exchange", rather than both of these nodes simply being attached to the next level up with best hypernym "group". It might be possible to correct for this situation by comparing the hypernyms for the two clusters and if there is little overlap, deleting their parent node and attaching them to their grandparent instead.

It would be useful to try to identify terms made up of multiple words, rather than just using the head nouns of the noun phrases.

Not only would this provide a more useful hierarchy, or at least perhaps one that is more useful for certain applications, but it would also help to prevent some errors. Hearst (1992) gives an example of a potential hyponym-hypernym pair "broken bone/injury". Using our algorithm, we would learn that "injury" is a hypernym of "bone". Ideally, this would not appear in our hierarchy since a more common hypernym would be chosen instead, but it is possible that in some cases a bad hypernym would be found based on multiple word phrases. A discussion of the difficulties in deciding how much of a noun phrase to use can be found in Hearst.

Ideally, a useful hierarchy should allow for multiple senses of a word, and this is an area which can be explored in future work. However, domain-specific text tends to greatly constrain which senses of a word will appear, and if the learned hierarchy is intended for use with the same type of text from which it was learned, it is possible that this would be of limited benefit.

We used parsed text for these experiments because we believed we would get better results and the parsed data was readily available. However, it would be interesting to see if parsing is necessary or if we can get equivalent or nearly-equivalent results doing some simpler text processing, as suggested in Ahlswede and Evens (1988). Both Hearst (1992) and Riloff and Shepherd (1997) use unparsed text.

7 Related work

Pereira et al. (1993) used clustering to build an unlabeled hierarchy of nouns. Their hierarchy is constructed top-down, rather than bottom-up, with nouns being allowed membership in multiple clusters. Their clustering is based on verb-object relations rather than on the noun-noun relations that we use. Future work on our project will include an attempt to incorporate verb-object data as well in the clustering process. The tree they construct is also binary with some internal nodes which seem to be "artificial", but for evaluation purposes they disregard the tree structure and consider only the leaf nodes. Unfortunately it is difficult to compare their results to ours since their evaluation is based on the verb-object relations.

Riloff and Shepherd (1997) suggested using conjunction and appositive data to cluster nouns; however, they approximated this data by just looking at the nearest NP on each side of a particular NP. Roark and Charniak (1998) built on that work by actually using conjunction and appositive data for noun clustering, as we do here. (They also use noun compound data, but in a separate stage of processing.) Both of these projects have the goal of building a single cluster of, e.g., vehicles, and both use seed words to initialize a cluster with nouns belonging to it.

Hearst (1992) introduced the idea of learning hypernym-hyponym relationships from text and gives several examples of patterns that can be used to detect these relationships including those used here, along with an algorithm for identifying new patterns. This work shares with ours the feature that it does not need large amounts of data to learn a hypernym; unlike in much statistical work, a single occurrence is sufficient.

The hyponym-hypernym pairs found by Hearst's algorithm include some that Hearst describes as "context and point-of-view dependent," such as "Washington/nationalist" and "aircraft/target". Our work is somewhat less sensitive to this kind of problem since only the most common hypernym of an entire cluster of nouns is reported, so much of the noise is filtered.

8 Conclusion

We have shown that hypernym hierarchies of nouns can be constructed automatically from text with similar performance to semantic lexicons built automatically for hand-selected hypernyms. With the addition of some improvements we have identified, we believe that these automatic methods can be used to construct truly useful hierarchies. Since the hierarchy is learned from sample text, it could be trained on domain-specific text to create a hierarchy that is more applicable to a particular domain than a general-purpose resource such as WordNet.

9 Acknowledgments

Thanks to Eugene Charniak for helpful discussions and for the data used in this project. Thanks also to Brian Roark, Heidi J. Fox, and Keith Hall for acting as judges in the project evaluation. This research is supported in part by NSF grant IRI-9319516 and by ONR grant N0014-96-1-0549.

References

Thomas Ahlswede and Martha Evens. 1988. Parsing vs. text processing in the analysis of dictionary definitions. In *Proceedings of the 29th Annual Meeting of the Association for Computational Linguistics*, pages 217–224.

Peter F. Brown, Vincent J. Della Pietra, Peter V. DeSouza, Jennifer C. Lai, and Robert L. Mercer. 1992. Class-based n-gram models of natural language. *Computational Linguistics*, 18:467–479.

Eugene Charniak, Sharon Goldwater, and Mark Johnson. 1998. Edge-based best-first chart parsing. In *Proceedings of the Sixth Workshop on Very Large Corpora*, pages 127–133. Association for Computational Linguistics.

Christiane Fellbaum, editor. 1998. *WordNet: An Electronic Lexical Database.* MIT Press.

Marti A. Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the Fourteenth International Conference on Computational Linguistics.*

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, 19:313–330.

Fernando Pereira, Naftali Tishby, and Lillian Lee. 1993. Distributional clustering of English words. In *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics*, pages 183–190.

Ellen Riloff and Jessica Shepherd. 1997. A corpus-based approach for building semantic lexicons. In *Proceedings of the Second Conference on Empirical Methods in Natural Language Processing*, pages 117–124.

Brian Roark and Eugene Charniak. 1998. Noun-phrase co-occurrence statistics for semi-automatic semantic lexicon construction. In *COLING-ACL '98: 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics: Proceedings of the Conference*, pages 1110–1116.