Automatic Construction of a Hypernym-Labeled Noun Hierarchy from Text

by
Sharon A. Caraballo
B. A., Rutgers University, 1994
Sc. M., Brown University, 1996

A dissertation submitted in partial fulfillment of the
requirements for the Degree of Doctor of Philosophy
in the Department of Computer Science at Brown University

Providence, Rhode Island
May 2001

This dissertation by Sharon A. Caraballo is accepted in its present form by the Department of Computer Science as satisfying the dissertation requirement for the degree of Doctor of Philosophy.

Date 7/28/00 | [illegible]
Eugene Charniak, Director

Recommended to the Graduate Council

Date 7/28/00 | [illegible]
Mark Johnson, Reader

Date 7/28/00 | [illegible]
Thomas Dean, Reader

Date July 28, 2000 | [illegible]
Philip Klein, Reader

Approved by the Graduate Council

Date Sept. 21, 2000 | [illegible]
Peder J. Estrup
Dean of the Graduate School and Research

Vita

Name	Sharon A. Caraballo
Born	May 14, 1969 in Elizabeth, NJ
Education	*Brown University*, Providence, RI Ph.D. in Computer Science, May 2001
	Brown University, Providence, RI Sc.M. in Computer Science, May 1996
	Rutgers University, New Brunswick, NJ B.A. in Computer Science with High Honors, May 1994
Honors	Sigma Xi Full Membership, 2000 Sigma Xi Associate Membership, 1996 Phi Beta Kappa, 1994 Brown University Fellowship, 1994 Tunis Quick Prize (competitive exam in English grammar and spelling open to all Rutgers University freshmen), 1987 National Merit Scholarship, 1986 Rutgers Presidential Scholarship, 1986
Professional Experience	*Nesbit Systems, Inc.*, 1991-94 Senior Programmer (1993-94), Programmer (1992-93), Software Tester (1991-92)
	Siemens Corporate Research, 1990-91 Organization & Information Systems Assistant

Teaching Experience

Brown University, Spring 1996
Teaching Assistant for CS 22, Discrete Mathematics

AT&T MIS Training, Summer 1990
Internship as Teaching Assistant for corporate training classes

Publications

Sharon A. Caraballo. "Automatic Acquisition of a Hypernym-Labeled Noun Hierarchy from Text." *37th Annual Meeting of the Association for Computational Linguistics: Proceedings of the Conference*, 1999, pages 120-126.

Sharon A. Caraballo and Eugene Charniak. "Determining the Specificity of Nouns from Text." *Proceedings of the 1999 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, 1999, pages 63-70.

Sharon A. Caraballo and Eugene Charniak. "New Figures of Merit for Best-First Probabilistic Chart Parsing." *Computational Linguistics* 24:2, 1998, pages 275-298.

Sharon A. Caraballo and Eugene Charniak. "Figures of Merit for Best-First Probabilistic Chart Parsing." *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 1996, pages 127-132.

Acknowledgments

The first person I would like to thank is, of course, my advisor, Eugene Charniak. He introduced me to the field of statistical natural language processing and taught me, both explicitly and by example, how to do good research. This thesis has benefited greatly from our many, many hours of discussion. He also supplied the parser and some of the statistical data on which this work is based. More important than any of that, though, is his understanding and patience, as well as his humor. He has been the advisor I hope to be someday.

Many other Brown faculty members have also helped in the creation of this thesis. I would like to thank my committee, Mark Johnson, Tom Dean, and Philip Klein, as well as all of the faculty who provided useful comments during my proposal, including John Hughes and David Laidlaw.

The evaluation of the results of my work required laborious human judging. I am most grateful to Brian Roark, Heidi Fox, Keith Hall, and Kathy Schark for their efforts, as well as my husband, David Caraballo, for serving as my "practice" judge to test out my evaluation procedure.

In many ways, this thesis would not have been possible without David. When I decided to attend graduate school, he supported me completely, even finishing his own doctorate long-distance to be with me. He has always helped me in every way possible, from helping sort through the technical math details behind some of the statistical techniques, to reading through my research papers even though they were completely out of his own areas of expertise, to simple emotional support; in fact, as much as he has done for me, I know he always wished there was more he could do.

I also want to thank the rest of my family. My parents, Andrew and Constance Adamus, have always been a great source of support, and the seed of my future in computer science was planted one afternoon when I was bored and my mother suggested I take a look at her BASIC textbook. My sister, Sheila Adamus Liotta, and her family, Louis, Nicholas, and

Marissa Liotta, have been particularly helpful, providing me with a home-away-from-home whenever I needed it. My sheepdog, Harry, has taken upon himself the very important job of letting me know when I've been working too hard and it's time to take a break from the computer. And last but not least (except in size), I want to thank my son Andrew, who was born in the middle of all of this, between my thesis proposal and defense, for bringing his sunshine into my life every day.

Contents

⋆ Parts of this thesis have been previously published in the Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics, 1999, and the Proceedings of the 1999 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora (with Eugene Charniak).

List of Tables

List of Figures

Chapter 1

Introduction

The purpose of this research is to develop techniques for extracting semantic information about words from text. In particular, given a large amount of text and no additional sources of semantic information, we will build a hierarchy of nouns appearing in the text. The hierarchy is in the form of an IS-A tree, where the nodes of the tree contain one or more nouns, and the ancestors of a node contain *hypernyms* of the nouns in that node. (An English 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.") Figure 1.1 shows an example of a noun hierarchy for a very small subset of English nouns.

Statistical techniques in natural language processing generally involve examining a large corpus, or body of text, and collecting counts of how many times various linguistic features occur. These counts are then used in some way to learn something about the text. In the work described here, simple counts of how many times words appear together in certain structures are collected, and from just these numbers we are able to deduce something about the meaning of the words. In particular, we are able to learn that certain nouns are superclasses, or hypernyms, of other nouns; e.g., that a "president" is a kind of "executive."

The task studied in this thesis is interesting not only because it demonstrates the ability to learn semantic information from text, but also because of its practical applications. In order to do any kind of linguistic task that involves understanding, a natural-language processing system must have a database of lexical semantic (word meaning) information. Many existing natural language processing systems are dependent on hand-built semantic hierarchies such as WordNet ([10]). However, general-purpose semantic resources like WordNet are insufficient for many problems, such as those involving text from specific domains, and hand-built resources generally are extremely time- and labor-intensive to create and

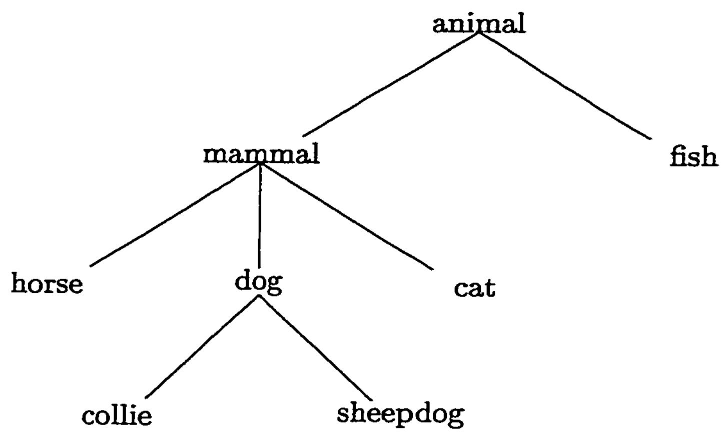

Figure 1.1: A small noun hierarchy.

maintain. The techniques proposed here could be used to build a general or domain-specific hierarchy of nouns or possibly to augment an existing hierarchy with domain-specific or simply updated information.

1.1 Thesis outline

Chapter 2 of this thesis presents an initial experiment in constructing a hierarchy. It also discusses a few simple refinements to that experiment. We give a detailed discussion of the algorithm for constructing an unlabeled hierarchy of nouns, and describe how hypernyms are used to label the internal nodes of that tree. Preliminary results are presented, including an evaluation of the hierarchy by human judges. We also discuss some minor refinements to the algorithm and demonstrate how they improve the quality of the hierarchy.

In Chapter 3, we discuss how a noun hierarchy can be viewed as a generative probability model for the type of data from which it was built. We then present and evaluate two alternative clustering methods inspired by the probability model. We evaluate hierarchies built by our clustering methods both in terms of the probability they assign to held-out data and in terms of human judgments.

Chapter 4 examines the particular subproblem of determining which of a pair of nouns is more specific. Our goal is to find a quantitative measure of specificity which can easily be computed from text. We present several candidates for such a measure and evaluate their performance. We are able to identify specificity measures which can determine which of two nouns is more specific with over 80% accuracy.

In Chapter 5, we incorporate the specificity measure from the previous chapter into our hierarchy-construction techniques. Once again, we use human judges to evaluate the hierarchy our system constructs.

Chapter 6 discusses some ways in which our techniques might be improved in future work, and Chapter 7 discusses related work from the literature.

We present our conclusions in Chapter 8.

Appendix A includes the data sets used for testing in Chapter 4.

Chapter 2

The initial experiment

An initial noun hierarchy has been constructed using simple statistical techniques and is presented in [4]. The techniques used in that paper, as well as some other techniques developed to support the construction of a noun hierarchy, are described in this chapter.

To create the labeled noun hierarchy, nouns are first clustered into an unlabeled hierarchy using data on conjunctions, appositives, and verb-object relations 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 is judged to be of comparable quality to semantic resources constructed in previous work on a much simpler task with more human intervention required. The success of these simple techniques in building a hierarchy demonstrate that automatic construction of a semantic hierarchy of nouns is feasible and give us a baseline level for quality, which is improved upon in later chapters by using techniques with a stronger grounding in theory.

2.1 Building an unlabeled noun hierarchy

The first stage in constructing the hierarchy is to build an unlabeled hierarchy of nouns using bottom-up clustering methods (see, e.g., Brown et al ([3])). Nouns are clustered based on conjunction, appositive, and verb-object data collected from the about 15 million words of 1987 Wall Street Journal text, parsed using the parser described in Charniak ([6]). To expedite the parsing, we used a version of the parser which runs relatively fast but at the expense of accuracy (about the 75% level on standard precision and recall measurements).

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"), along with all verb-object relations. The idea here is that nouns in conjunctions or appositives, or nouns which appear as objects of the same verbs, tend to be semantically related, as discussed in Riloff and Shepherd ([28]) and Roark and Charniak ([29]). Since these nouns have semantic features in common, it is likely that they belong to a common semantic class; in other words, that there is a hypernym which describes them. We do not have a great deal of hypernym data for individual nouns, but by clustering the nouns together in the manner described here and combining the statistical evidence for hypernyms from all nouns in a cluster (as described in the next section), we hope to be able to build a labeled hierarchy of nouns and their hypernyms.

Taking the head words of each noun phrase appearing in one of the structures listed above and stemming them results in data for about 50,000 distinct nouns. For each noun, we compute the total count for each feature, where a feature is a co-occurrence of a particular noun in a conjunction or appositive with it, or a verb appearing with the noun as its object. We then construct a vector of the feature counts. Since the lowest-frequency nouns would be placed into position in the hierarchy based on very little data and would therefore be prone to errors, we chose to filter out nouns with a feature vector of length less than 2. This results in approximately 20,000 nouns to be included in our hierarchy.

There are many possible hierarchical clustering methods which could be used to cluster the nouns. For this experiment, we use a standard technique of clustering "bottom-up" according to the similarity of the nouns, which we define as the cosine of the angle between their feature vectors. In principle any hierarchical clustering method could be used; we selected our particular method simply because it is a well-known, easily implemented technique. Later chapters of this thesis explore other, more theoretically-grounded clustering methods for constructing our hierarchy.

We can 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.

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

where $\mathbf{a}$ ranges over all vectors for nouns in group A, $\mathbf{b}$ ranges over the vectors for group

B, and $size(X)$ represents the number of nouns which are in group X.

We want to create a tree of all of the nouns in this data such that a node in the tree represents a group of semantically related nouns; the leaves of the tree are the individual nouns, and an internal node of the tree represents a cluster of nouns made up of all nouns in its descendant leaves. We construct the tree using standard bottom-up clustering techniques as follows:

1. Put each noun into its own node.

2. Compute the similarity between each pair of nodes using the cosine method.

3. Find the two most similar nodes and combine them by giving them a common parent (and removing the child nodes from future consideration).

4. 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

$$\begin{aligned} sim(C,I) &= \frac{\sum_{\mathbf{c},\mathbf{i}} \cos(\mathbf{c},\mathbf{i})}{size(C)size(I)} \\ &= \frac{\sum_{\mathbf{a},\mathbf{i}} \cos(\mathbf{a},\mathbf{i}) + \sum_{\mathbf{b},\mathbf{i}} \cos(\mathbf{b},\mathbf{i})}{(size(A)+size(B))\,size(I)} \\ &= \frac{sim(A,I)size(A)size(I) + sim(B,I)size(B)size(I)}{(size(A)+size(B))\,size(I)} \\ &= \frac{sim(A,I)size(A) + sim(B,I)size(B)}{size(A)+size(B)}. \end{aligned}$$

5. Repeat steps 3 and 4 until all nouns have been placed under a common ancestor.

Figure 2.1 shows an actual piece of the unlabeled tree constructed by this method. The numbers indicate the order in which the clusters were combined; "PC" and "clones" were the words judged to be most similar, then "minicomputer" and "workstation," then that cluster combined with "microsystems," and finally everything was combined under a common ancestor.

Nouns which have no features in common with any other noun will have a cosine of 0 with every other noun. Such nouns are therefore not included in the final tree. Also, a few

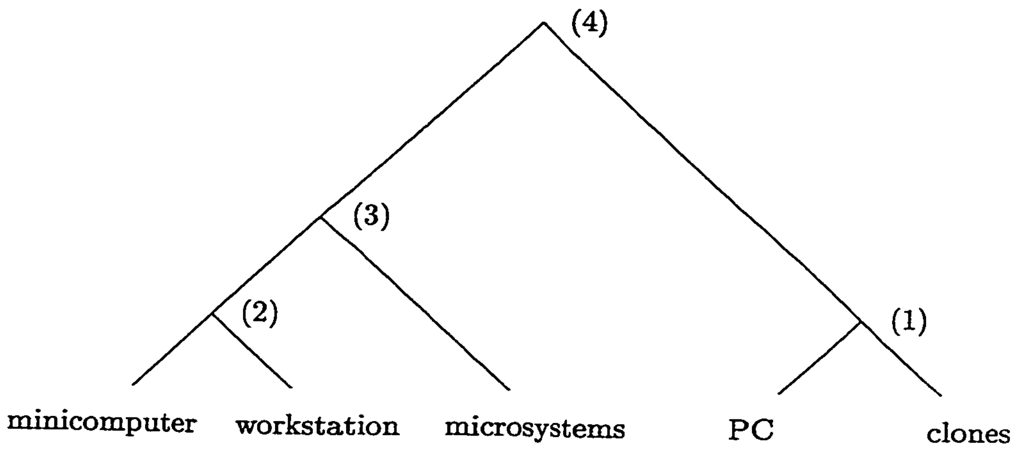

Figure 2.1: A bottom-up clustering example.

small clusters are formed of nouns which have nonzero cosine with each other, but 0 cosine with every noun in the main tree, so these nouns also do not appear in our hierarchy. This is a minor issue; out of our approximately 20,000 nouns, fewer than 100 cannot be added into the tree.

In practice, we cannot follow exactly the algorithm described above, because maintaining a list of the cosines between every pair of nodes requires a tremendous amount of memory. With 20,000 nouns, we would initially require a 20,000 x 20,000 array of values (or a triangular array of about half this size). With our current hardware, the largest array we can comfortably handle is over 10 times smaller; we can only 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 number of nodes to 5,000 again, and once again we cluster until 2,500 nodes remain. This process is repeated until all nouns have been processed, at which point the nodes are clustered until as many as possible are combined into a single cluster. Visual inspection of hierarchies produced for smaller data sets with and without the batch processing show only minor differences, typically in the deepest levels of the tree. Since the fine-grained detail at those levels is generally not significant, and is in fact lost when hypernyms are added to the tree as described in the next two sections, the batch processing has little impact on the ultimate result. It is possible that in future work more specific hypernyms will be found and the detail in the deepest parts of the tree will take on greater importance; in that case, it will become necessary to consider how nouns should be assigned to particular batches for processing. (Currently this is done simply by the order in which the nouns were encountered in the corpus.)

Once the unlabeled tree is constructed, our next step is to try to label each of the internal nodes with a hypernym describing its descendant nouns.

2.2 Assigning hypernyms

To determine possible hypernyms for a particular noun, we use the same parsed text described in the previous section. As suggested in Hearst ([14] and [15]), we can find some hypernym data in the text by looking for conjunctions involving the word "other," as in "X, Y, and other Zs," from which we can extract that Z is likely a hypernym for both X and Y.

These co-occurrences are extracted from the parsed text, and for each noun we construct a vector of hypernym features, 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.

2.3 Compressing the tree

The labeled tree constructed in the previous section tends to be contain a great deal of unnecessary, essentially meaningless structure. 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

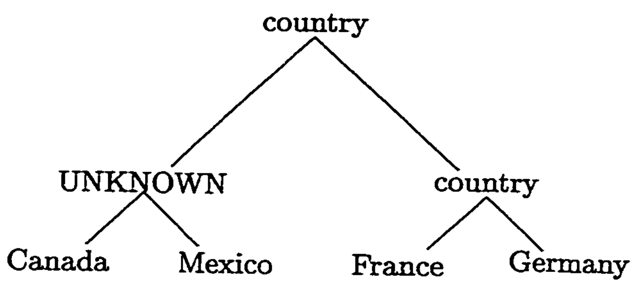

Figure 2.2: A labeled hierarchy before compression.

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. Given a large binary tree of countries, the ideal single-word hypernym at each level would be "country" (or "nation"), so we would like to combine these binary subtrees into a single cluster 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 extraneous structure in the tree 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.

Figure 2.2 shows a section of an uncompressed labeled hierarchy, and Figure 2.3 shows the result of compressing that hierarchy.

2.4 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

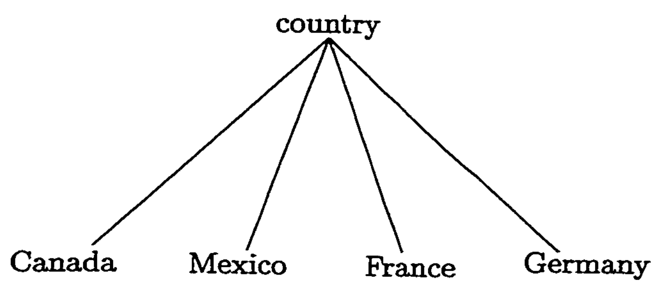

Figure 2.3: The hierarchy from Figure 2.2 after compression.

rather than the more general "thing/person/group." However, if the hierarchy were to be used for text from the financial domain, the domain-specific labels may be preferred.

The next level of the hierarchy, the children of the root, is as shown in Table 2.1. The numbers in the table 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 could be found, 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 native speakers of English who were not otherwise involved in 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 the 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 2.2 presents the results of this evaluation. The table lists results for only 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 having an opinion, 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

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 2.1: The children of the root node in the preliminary experiment.

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 2.2: The results of the judges' evaluation for the preliminary experiment.

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

Table 2.3: The results of the judges' evaluation of noise words.

([28]). 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 ([29]) 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 future work chapter might correct this problem.

Of the 50 noise words, a few were actually rated as correct as well, as shown in Table 2.3. 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.

2.5 The effect of better data

We re-created the hierarchy using a better parser to parse the Wall Street Journal text. This parser, also developed by Eugene Charniak, performs at about the 88% level on standard precision and recall measures. In the new hierarchy, there are 16,826 leaves (nouns) and 493 internal nodes in the final tree, with the top-level node labeled "product/official/company." To get an idea of the difference in result quality, Table 2.4 gives the next level of the new hierarchy.

Comparing Table 2.4 to Table 2.1 shows that using the data obtained with the more accurate parser, the top of the hierarchy is much less messy. The new hierarchy appears to

Hypernyms	# nouns
group/institution/organization	84
official/company/executive	8204
product/factor/item	7542
country/nation/producer	320

Table 2.4: The children of the root node in the hierarchy developed using better-parsed text.

Three best hypernyms	Hypernym 1		Any hypernym	
	majority	any	majority	any
director/official/executive	16	18	16	18
worker/employee/group	5	10	11	17
factor/problem/asset	16	16	17	17
cost/factor/term	3	11	3	15
item/product	3	3	4	4
measure/transaction/purpose	5	11	7	15
condition/approval/disorder	11	11	11	12
help/purpose	9	11	11	12
asset/instrument/investment	8	8	12	12
apparel/knitwear/product	14	17	14	17
AVERAGE	9.0 / 45.0%	11.6 / 58.0%	10.6 / 53.0%	13.9 / 69.5%

Table 2.5: The results of the judges' evaluation using better-parsed data.

do a much better job of combining nodes under a common ancestor.

We also had human judges evaluate the new hierarchy. The results of the human evaluation are presented in Table 2.5.

The results in Table 2.5 illustrate that an improvement in the quality of parsing has a dramatic effect on the quality of the hierarchy constructed. For convenience, the results from the two different parsers are summarized in Table 2.6.

2.6 Refinements to the algorithm

Our initial algorithm can also be modified in a few simple ways to improve our results.

Parser	Hypernym 1		Any hypernym	
	majority	any	majority	any
original parser	6.6 / 33.0%	7.8 / 39.0%	9.5 / 47.5%	12.1 / 60.5%
improved parser	9.0 / 45.0%	11.6 / 58.0%	10.6 / 53.0%	13.9 / 69.5%

Table 2.6: A comparison of the hierarchies constructed using two different parsers.

Three best hypernyms	Hypernym 1		Any hypernym	
	majority	any	majority	any
incentive/program/purpose	7	10	7	13
executive/analyst/employee	17	18	18	18
agency/item/bank	7	7	9	10
analyst/professional/investor	13	18	13	18
official/executive/analyst	17	17	17	17
analyst/employee/executive	9	18	9	18
crop/product/asset	1	10	1	13
firm/airline/industry	13	16	13	16
country/nation/producer	12	17	12	17
group/investor/ally	14	15	15	16
AVERAGE	11.0 / 55.0%	14.6 / 73.0%	11.4 / 57.0%	15.6 / 78.0%
previous best	9.0 / 45.0%	11.6 / 58.0%	10.6 / 53.0%	13.9 / 69.5%

Table 2.7: The results of the judges' evaluation for the refined algorithm.

First, we include hypernyms as another type of co-occurrence; having conjuncts, appositives, and verbs as features, we also include the hypernyms which have appeared with each noun. Then each node of the tree has an associated distribution of hypernyms, and we can simply select the most frequent hypernyms as the label for that node. As an added benefit, the richer co-occurrence data should improve our clustering. We also extend our hypernym data to include Hearst's suggested constructions involving the phrase "such as," as in "Zs, such as X and Y," which implies that Z is a hypernym of X and Y.

We also note that clusters which are labeled with fewer than three candidate hypernyms tend to perform more poorly on evaluations, although this is not always the case. If only one or two candidate hypernyms are found for a cluster, the hypernym data for that cluster are likely very sparse, and probably insufficient to produce a meaningful label. This suggests that we should make our criteria for labeling clusters somewhat stricter. As a simple modification to screen out some sparse data problems, we now only label a node once we have at least three candidate hypernyms available.

2.6.1 Evaluation of the modified algorithm

Table 2.7 presents the results of a human evaluation of a noun hierarchy incorporating these refinements. Once again, we see a dramatic improvement in our results.

Chapter 3

A probability model for noun hierarchies

We can use a generative probability model to formalize the ideas behind the hierarchy-building process. The idea is that we assume there is a "true" underlying hierarchy from which the observed co-occurrence data were generated. We want the algorithm to build a hierarchy that is as close to this "true" hierarchy as possible. The assumption is that a better hierarchy will assign a higher probability to the observed data.

First we need to discuss how the hierarchy assigns probabilities. Assume that there is a process which randomly generates a noun n with probability $P(n)$, and then randomly generates a context word w to co-occur with it, with probability $P(w|n)$. So, for example, the process might generate the pair ("product", "sell") by generating the noun "product" and the word "sell" to co-occur with it (as a verb of which "product" is the direct object), or it might generate the pair ("president", "CEO") by first generating the noun "president" and then generating "CEO" as a word co-occurring in conjunction with it. The probability of a noun/context word pair is computed as

$$P(n, w) = P(n)P(w|n).$$

We assume that the $P(n)$ values are simply approximated by the empirical probabilities of the nouns; that is,

$$P(n) \approx \hat{P}(n).$$

These values are independent of the hierarchical structure. The $P(w|n)$ values are estimated according to a particular noun hierarchy as discussed below.

The idea behind clustering in linguistic applications is to overcome sparse data problems by providing statistics on clusters of related items. These statistics can be used in a backed-off model; something like

$$P(w|n) \approx \lambda\hat{P}(w|n) + (1-\lambda)\hat{P}(w|c)$$

for a noun n in a cluster c and some appropriate value for the coefficient λ.

With hierarchical clustering, we can have many levels of backing off. Assuming that there is nothing special about any particular level, at any level we can use the same coefficients for the contributions from that level and the levels above it:

$$P(w|n) \approx \lambda\hat{P}(w|n) + (1-\lambda)\left[\lambda\hat{P}(w|c_1) + (1-\lambda)\left[\lambda\hat{P}(w|c_2) + (1-\lambda)\left[\ldots\right]\right]\right]$$

where c_1 is the cluster immediately above n, c_2 is the cluster immediately above that, and so on up to the root of the tree.

The best tree would be the one that maximizes the probability of the data according to this model:

$$P(data|tree) = \prod_{item \in data} P(item|tree),$$

where $P(item|tree)$ is the probability $P(n, w)$ discussed earlier.

There is one further complication. Rather than using the same λ for all clusters, we want it to be dependent on cluster size. We do not want to assume that the probability distribution for a cluster is equally good whether it is based on 5 co-occurrences or 5,000. So rather than being a single variable, λ is actually a function that returns a particular value based on the size of a cluster. We then have:

$$P(w|n) \approx \lambda(n)\hat{P}(w|n) + (1-\lambda(n))\left[\lambda(c_1)\hat{P}(w|c_1) + (1-\lambda(c_1))\left[\lambda(c_2)\hat{P}(w|c_2) + (1-\lambda(c_2))\left[\ldots\right]\right]\right].$$

3.1 Hypernym probabilities

With the new idea of a hierarchy as a generative probability model, there is an obvious way to choose the best hypernym for a particular internal node of the tree. We can simply include the hypernym data the same way as any other co-occurrence data, so that we have probabilities for the hypernyms for each cluster just as for any other context words. The best hypernym for a node is just the hypernym that maximizes that probability. This is the same hypernym selection method used in the final section of the previous chapter.

3.2 Probabilistic clustering schemes

We can also modify our clustering scheme itself to try to produce a noun hierarchy which better models our co-occurrence data. We have examined how two probabilistically-based clustering schemes can be used to produce trees which model our data better.

3.2.1 Clustering based on probability distributions

Our probability model suggests an obvious clustering scheme to try. As usual, we start with each noun in its own cluster, and combine the two most "similar" clusters recursively until everything is in one cluster. But now instead of our similarity measure being based on cosine, we want a similarity measure which indicates which two clusters to combine to maximize the probability of the observed data.

As discussed earlier, the probability of the observed data according to a particular tree is $\prod P(n,w)$ over all (n,w) pairs in the data. We compute the item probabilities as

$$P(n,w) = P(n)P(w|n),$$

using the approximations

$$P(n) \approx \hat{P}(n)$$

and

$$P(w|n) \approx \lambda(n)\hat{P}(w|n)+(1-\lambda(n)) \left[\lambda(c_1)\hat{P}(w|c_1) + (1-\lambda(c_1)) \left[\lambda(c_2)\hat{P}(w|c_2) + (1-\lambda(c_2)) [\ldots]\right]\right].$$

$\hat{P}(w|n)$ is computed in the obvious way – the count of co-occurrences of w and n divided by the number of co-occurrences of anything and n. In other words, $\hat{P}(w|n)$ is the probability of w according to the distribution for n defined by the observed data. $\hat{P}(w|c)$ for a cluster c is then computed by counting the co-occurrences of w and any noun in c.

So, when we form cluster c from two other clusters, the probability of a data item including a context word w and a noun in c is updated to include a contribution from $\hat{P}(w|c)$.

Rather than recomputing the full probabilities at every step of the clustering algorithm, we can use a simpler scheme. At each step of the clustering, we choose to combine the two clusters for which the average probabilities of the data items given the new cluster are maximized. We do not need to look at the full probabilities for the items, just the probabilities given that particular cluster, since the rest of the terms in the item probabilities will be the same regardless of what the new parent cluster is. Let $D(x)$ represent the probability distribution of words co-occurring with words in cluster x. Then for two clusters A and B, the per-word probability p_{pw} of the cluster which would be formed by combining them is

$$p_{pw}(A,B) = \sqrt[|D(A)+D(B)|]{\prod_{w \in D(A)+D(B)} \hat{P}(w|A+B)}.$$

The p_{pw} measure takes the place of the similarity measure in our previous algorithm; at every step, we combine the A and B for which $p_{pw}(A,B)$ is maximized. This greedy strategy should build a reasonable approximation to the optimal tree.

In our original algorithm, the similarity of two clusters was defined as the average pairwise similarity of their members, so when two clusters were combined, the new cluster's similarity to each other cluster could be easily computed as a weighted average of the similarities for its children. However, the situation with this probabilistic measure is somewhat more complex. Assume nodes A and B have been combined under a new parent C, and the distributions $D(A)$ and $D(B)$ have been combined to create $D(C)$. We want to compute the per-word probability for a potential new cluster formed by combining C and each other node I. Here, the fact that we have $p_{pw}(A,I)$ and $p_{pw}(B,I)$ is of little help. Instead, we compute $p_{pw}(C,I)$ from scratch.

Rather than having to recompute the probability of each word in each distribution, we store the geometric mean of the total probability of all items for each cluster X; i.e., $p_{pw}(X)$. (In fact, these calculations are really done with logs, but in this discussion it is easier to proceed as if we are storing actual probabilities.) To compute $p_{pw}(C,I)$, we can take advantage of these numbers. For any word that appears only with nouns in C, its new probability is

$$\hat{P}(w|C+I) = \frac{|C|}{|C+I|}\hat{P}(w|C).$$

Similarly, for any word appearing only with I,

$$\hat{P}(w|C+I) = \frac{|I|}{|C+I|}\hat{P}(w|I).$$

If we assume that all words appear with only one cluster, then

$$p_{pw}(C,I) = \sqrt[|C+I|]{\left(\frac{|C|}{|C+I|}p_{pw}(C)\right)^{|C|}\left(\frac{|I|}{|C+I|}p_{pw}(I)\right)^{|I|}}.$$

(It should be clear at this point why using logs is necessary.)

Of course, some words do appear in co-occurrences with nouns from both clusters, but it is much simpler to compute the adjustments for just these words. These adjustments are made to the product of the probabilities, before taking the $|C+I|$th root. For any word w which co-occurs with nouns from both C and I, we divide out the separate contributions and multiply in the actual contribution $\hat{P}(w|C+I)$. However, in order to identify such words w, we do need to traverse the words co-occurring with either C or I and look them up to see if they appear in the other cluster, so this clustering algorithm will still be more complex than our original algorithm.

3.2.2 Clustering based on Kullback-Leibler divergence

For a simpler clustering scheme still based on probabilistic information, we used an algorithm much like our original clustering algorithm, differing only in the way item and cluster similarities are computed. Rather than using the cosine between the vectors representing the distribution of co-occurrences for each noun, we would like a measure that indicates how similar the co-occurrence probability distributions are for each noun.

Kullback-Leibler divergence (KL divergence), or relative entropy, is a measure of how different two probability distributions are. The KL divergence of two probability distributions $p(x)$ and $q(x)$ over X is computed as

$$D(p||q) = \sum_{x \in X} p(x) \log_2 \frac{p(x)}{q(x)}.$$

However, using KL divergence directly in our clustering would result in two problems, as described in Manning and Schütze ([22]). First, the measure is not symmetric; that is, usually

$$D(p||q) \neq D(q||p).$$

Second, when computing the KL divergence between two distributions p and q, $D(p||q)$ is undefined if there exists any x for which $p(x) > 0$ and $q(x) = 0$. One solution to these problems, and the one adopted here, is to instead use the information radius, defined as

$$IRad(p,q) = D(p||\frac{p+q}{2}) + D(q||\frac{p+q}{2}).$$

which is both symmetric and always defined. This measure compares both p and q to the distribution obtained by averaging p and q.

Of course, this measures the dissimilarity between two distributions, and we are interested in a similarity measure. Since $IRad$ varies between 0 and 2, in order to produce a similarity measure that, like cosine and probability, varies between 0 and 1 with 0 being least similar and 1 being most similar, we simply subtract the $IRad$ from 2 and divide by

2:

$$sim(n_1, n_2) = 2 - \frac{D(p_1||\frac{p_1+p_2}{2}) + D(p_2||\frac{p_1+p_2}{2})}{2},$$

where p_1 is the probability distribution for words co-occurring with noun n_1 and p_2 is the probability distribution for words co-occurring with noun n_2.

The rest of the algorithm operates exactly as in the cosine case. Cluster similarities are computed as average pairwise similarities of their members,

$$sim(A, B) = \frac{\sum_{a \in A, b \in B} sim(a, b)}{size(A)size(B)},$$

and when two clusters A and B are combined under a new parent C, we can easily compute the new cluster's similarity to each other cluster I as

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

3.3 Evaluating data probabilities assigned by each model

In order to judge how well the trees built by each clustering scheme model the data, we held out one-tenth of our co-occurrence data and used each clusterer to build a tree based on the other nine-tenths. We then evaluated how well each tree predicts the held-out data by looking at the probability it assigned to it.

3.3.1 λ training

In order to compute the actual probabilities assigned to the data, we need to find actual values for the λs. We can do this using the Baum-Welch algorithm for expectation maximization (see Baum et al ([2]) for the original algorithm, Manning and Schütze ([22]) for a description of using it for λ training).

We can imagine the probabilities as coming from a Hidden Markov Model. Figure 3.1 shows a portion of such an HMM. Each arc is labeled with a symbol emitted (or the empty string ϵ) and the probability of traversing that arc. The total probability of an item is the sum of all possible paths through the HMM emitting that item. Again, the diagram shown here is not complete; in the full HMM there are other arcs from the λ and *root* nodes labeled with other co-occurring words, and there is a start node with arcs labeled with each noun n and probability $\hat{P}(n)$ which lead to an n node such as shown here. Given such an HMM, the Baum-Welch algorithm provides a way to learn locally optimal values for the λs.

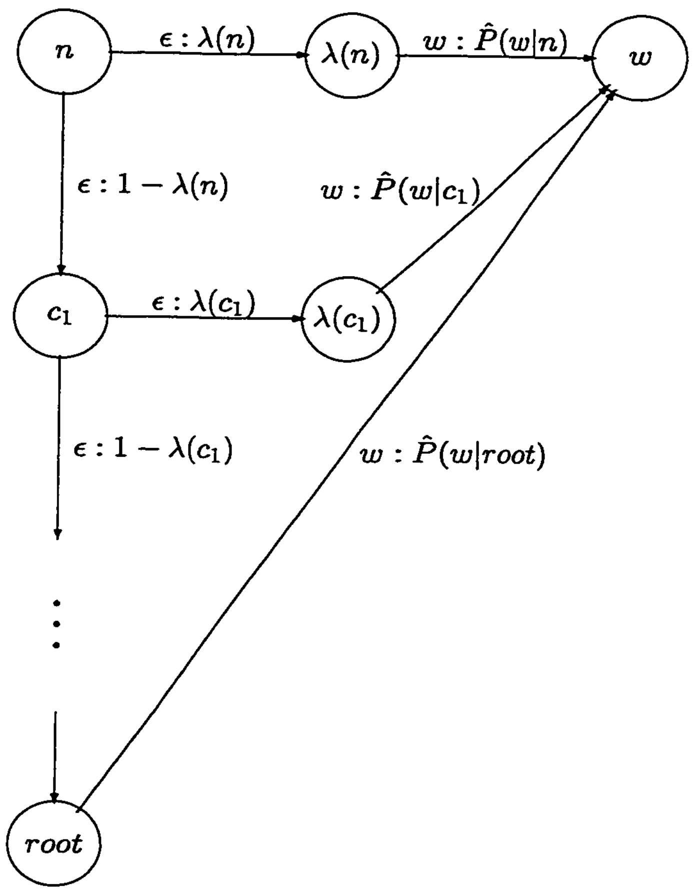

Figure 3.1: A section of an HMM for a probability model determined by a noun hierarchy.

Clustering basis	Per-item $\log_{10}$ prob	Per-item prob
cosine	-6.026	9.43×10^{-7}
cluster probability	-6.008	9.82×10^{-7}
KL divergence	-6.005	9.89×10^{-7}

Table 3.1: Probabilities assigned to held-out data by hierarchies based on three clustering schemes.

We use a separate λ value for each of four groups of clusters: those with up to 10 co-occurrences in our observed data, those with 11-100, those with 101-1000, and those with greater than 1000. We initially set all four λ values to 0.1.

Following the standard Baum-Welch algorithm, we compute the probability of each item in the held-out data according to the probability model defined by our noun hierarchy and the current λ values. For each λ and each data item, we find the probability-weighted counts of how many times an arc was traversed labeled with that λ, divided by the total probability for that data item. We do the same for each arc labeled with $1 - \lambda$ for that λ. We sum the values obtained for each pair of λ and its corresponding $1 - \lambda$. The total counts for each λ are then divided by the sum of the counts for that λ and $1 - \lambda$ to get the new λ value for the next iteration.

We repeat this process until the probabilities have converged. For this application, we chose to stop once the change in average probability per item between iterations was less than 0.1% of the previous value. (We use per-item probability rather than total probability as a basis for comparing models because depending on the model, some items might result in 0 probability and are therefore omitted from the λ calculations for that model.)

3.3.2 Results

We use the Baum-Welch algorithm to learn appropriate λ values for each hierarchy. The probabilities assigned to the held-out data by the probability models incorporating these λ values are shown in Table 3.1.

As we might expect, the two clusterers which treat the co-occurrence data as probability distributions perform better on this measure than the cosine-based version. It is interesting to note that these two models assign quite similar probabilities to the held-out data. Of course, the true test of the quality of the hierarchies is not how well they can predict the held-out data, but in how good they are determined to be by our human judges.

Three best hypernyms	Hypernym 1		Any hypernym	
	majority	any	majority	any
company/official/institution	6	12	7	13
company/firm/concern	7	8	11	11
maker/carrier/product	2	4	4	6
country/nation/debtor	13	16	15	16
problem/distortions/index	14	15	17	17
factor/measure/expense	0	1	8	10
carrier/company/insurer	2	12	2	15
city/country/center	7	8	7	9
industry/product/issue	13	15	15	16
product/hazard/industry	12	16	17	19
AVERAGE	7.6 / 38.0%	10.7 / 53.5%	10.3 / 51.5%	13.2 / 66.0%

Table 3.2: The results of the judges' evaluation for the tree-builder based on cluster probability.

3.4 Human evaluation

Tables 3.2 and 3.3 give detailed results of the human evaluations of the clustering schemes based on cluster probability and KL divergence, respectively. Table 3.4 summarizes the results for our three clustering schemes.

The most striking aspect of these results is certainly the very poor quality of the clusterer based on cluster probability. The reason for this can be understood by examining what the algorithm actually does. At any given iteration in the algorithm, we choose to combine the two clusters such that the probability of the co-occurrence data for the combined cluster is maximized. When a cluster has a very skewed distribution, the probability of its co-occurrence data is very high. If there is a great deal of co-occurrence data for that cluster, such as in the case of a cluster consisting of a single high-frequency noun with a skewed distribution, the cluster can combine with a cluster with a small amount of data with almost no reduction in probability. This is exactly what happens as the tree is built. The high-frequency clusters with skewed distribution tend to absorb the low-frequency clusters, regardless of what their distributions look like. Two low-frequency clusters with very similar distributions will not get the opportunity to combine; instead, they are first absorbed by (possibly different) high-frequency clusters. Although this produces a tree which assigns high probability to the held-out data, the internal structure of the tree is not at all desirable.

Our clustering scheme based on KL divergence produces a much nicer tree. The quality of this tree is virtually identical to that produced by the cosine-based clusterer. Since the KL-based clusterer also has a sound basis in probability, it is the clustering method we will

Three best hypernyms	Hypernym 1		Any hypernym	
	majority	any	majority	any
company/country/official	3	14	3	14
country/nation/market	7	19	8	19
crime/activity/charge	9	13	9	15
cost/charge/benefit	1	4	3	7
official/executive/analyst	15	15	15	15
product/compound/commodity	19	19	19	19
product/item/device	15	15	15	15
incentive/benefit/service	5	11	11	14
factor/issue/purpose	17	19	19	19
item/truck/good	15	15	15	15
AVERAGE	10.6 / 53.0%	14.4 / 72.0%	11.7 / 58.5%	15.2 / 76.0%

Table 3.3: The results of the judges' evaluation for the tree-builder based on KL divergence.

Three best hypernyms	Hypernym 1		Any hypernym	
	majority	any	majority	any
cosine	11.0 / 55.0%	14.6 / 73.0%	11.4 / 57.0%	15.6 / 78.0%
cluster probability	7.6 / 38.0%	10.7 / 53.5%	10.3 / 51.5%	13.2 / 66.0%
KL divergence	10.6 / 53.0%	14.4 / 72.0%	11.7 / 58.5%	15.2 / 76.0%

Table 3.4: Summary of the results of the judges' evaluation for the three clustering schemes.

prefer.

Chapter 4

Comparing the specificity of nouns

Some areas of the hierarchy might be improved if we were able to determine which of two nouns was more specific. In particular, some of the "hypernyms" in our co-occurrence data are in fact *hyponyms*, that is, the noun is actually a hypernym of the co-occurring word, rather than the reverse. We encountered this phenomenon numerous times in our experiments, generally because of dropped modifiers. Recall that we collect statistics on single words, rather than on multiple-word terms. When we encounter the phrase "pasta products, sauces and other specialty foods" in our corpus, it is treated as "products, sauces and other foods," leading to the conclusion that "food" is a hypernym of "product." If we knew that in fact "product" was a very general word compared to "food," we could filter this type of error out of our co-occurrence statistics. What we would like to have is a quantitative measure of specificity, to allow these comparisons to be made easily.

In [5], we presented and compared several methods for determining the specificity of nouns, again using only a text corpus. Each of these methods was trained on the text of the 1987 Wall Street Journal corpus discussed earlier.

4.1 The specificity measures

One possible indicator of specificity is how often the noun is modified. It seems reasonable to suppose that very specific nouns are rarely modified, while very general nouns would usually be modified. Using the parsed text, we collected statistics on the probability that a noun is modified by a prenominal adjective, verb, or other noun. (In all of these measures, when we say "noun" we are referring only to common nouns, tagged NN or NNS, not proper nouns tagged NNP or NNPS. Our results were consistently better when proper nouns were eliminated, probably since the proper nouns may conflict with identically-spelled common nouns.) We looked at both the probability that the noun is modified by any of these modifiers and the probability that the noun is modified by each specific category. The nouns, adjectives, and verbs are all stemmed before computing these statistics.

$$\begin{aligned}
P_{adj}(noun) &= \frac{count(noun \text{ with a prenominal adjective})}{count(noun)} \\
P_{vb}(noun) &= \frac{count(noun \text{ with a prenominal verb})}{count(noun)} \\
P_{nn}(noun) &= \frac{count(noun \text{ with a prenominal noun})}{count(noun)} \\
P_{mod}(noun) &= \frac{count(noun \text{ with prenom. adj, vb, or nn})}{count(noun)}
\end{aligned}$$

However, if a noun almost always appears with exactly the same modifiers, this may be an indication of an expression (e.g., "ugly duckling"), rather than a very general noun. For this reason, we also collected entropy-based statistics. For each noun, we computed the entropy of the rightmost prenominal modifier.

$$H_{mod}(noun) = -\sum_{modifier} [P(modifier|noun) * \log_2 P(modifier|noun)]$$

where $P(modifier|noun)$ is the probability that a (possibly null) $modifier$ is the rightmost modifier of $noun$. The higher the entropy, the more general we believe the noun to be. In other words, we are considering not just how often the noun is modified, but how much these modifiers vary. A great variety of modifiers suggests that the noun is quite general, while a noun that is rarely modified or modified in only a few different ways is probably fairly specific.

We also looked at a simpler measure which can be computed from raw text rather than parsed text. (For this experiment we used the part-of-speech tags determined by the parser, but that was only to produce the set of nouns for testing. If one wanted to compute this measure for all words, or for a specific list of words, tagging would be unnecessary.) We simply looked at all words appearing within an n-word window of any instance of the word being evaluated, and then computed the entropy measure:

$$H_n(noun) = -\sum_{word} [P(word|noun) * \log_2 P(word|noun)]$$

where $P(word|noun)$ is the probability that a word appearing within an n-word window of *noun* is *word*. Again, a higher entropy indicates a more general noun. In this measure, the nouns being evaluated are stemmed, but the words in its n-word window are not.

Finally, we computed the very simple measure of frequency $freq(noun)$). The higher the frequency, the more general we expect the noun to be. (Recall that we are using tagged text, so it is not merely the frequency of the word that is being measured, but the frequency of the word or its plural tagged as a common noun.)

This assumed inverse relationship between frequency and the semantic content of a word is used commonly in the literature, for example, to weight the importance of terms in the standard IDF measure used in information retrieval (see, e.g., Sparck Jones ([34])), and to weight the importance of context words to compare the semantic similarity of nouns in Grefenstette ([12]).

4.2 Evaluation of the specificity measures

To evaluate the performance of these measures, we used the hypernym data in WordNet ([10]) as our gold standard. We constructed three small sub-hierarchies of the nouns in WordNet and looked at how often our measures found the proper relationships between the hypernym/hyponym pairs in these hierarchies.

To build our three sub-hierarchies, we wanted to use sets of words for which there would be a reasonable amount of data in the Wall Street Journal corpus. We chose three noun clusters produced by a program similar to [29] except that it is based on a generative probability model and tries to classify all nouns rather than just those in pre-selected clusters. (All data sets are given in Appendix A.) The clusters we selected represented vehicles (car, truck, boat, ...), food (bread, pizza, wine, ...), and occupations (journalist, engineer, biochemist, ...). Since the clusterer was able to determine that these words were somehow related, we assume that our corpus contains sufficient data for them. From the clustered data we removed proper nouns and words that were not really in our target categories. We then looked up the remaining words in WordNet, and added their single-word hypernyms to the categories in the correct hierarchical structure. (Many WordNet categories are described by multiple words, e.g., "motorized vehicle," and these were omitted for obvious reasons.) We continued looking up hypernyms until each of our sets of words was built up into a tree structure under a common parent. This gave us three small sub-nets of WordNet to use for testing.

For each of these three hierarchies, we looked at each hypernym/hyponym pair within the

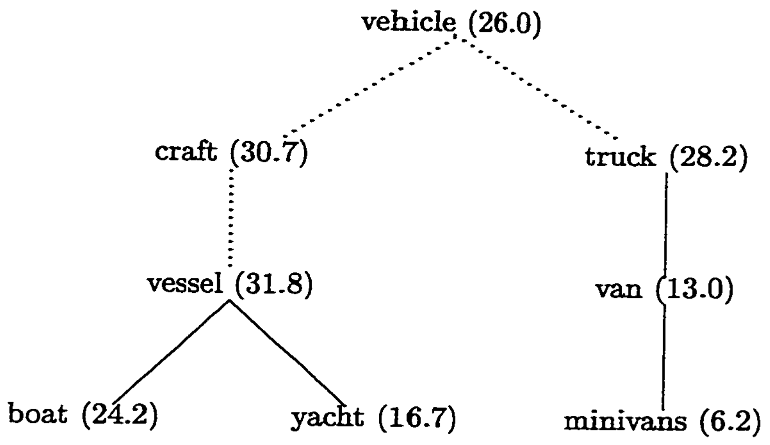

Figure 4.1: A piece of the "vehicles" hierarchy showing errors on the parent-child correctness measure.

Specificity measure	Vehicles	Food	Occupations	Average
P_{mod}	65.2	63.3	66.7	65.0
P_{adj}	65.2	67.3	69.7	67.4
P_{vb}	73.9	42.9	51.5	56.1
P_{nn}	65.2	57.1	51.5	58.0
H_{mod}	91.3	79.6	72.7	81.2
H_2	87.0	79.6	75.8	80.8
H_{10}	87.0	79.6	75.8	80.8
H_{50}	87.0	85.7	75.8	82.8
$Freq$	87.0	83.7	78.8	83.1

Table 4.1: Percentage of parent-child relationships which are ordered correctly by each specificity measure.

hierarchy and determined whether each specificity measure placed the words in the proper order. Figure 4.1 shows a piece of the "vehicles" hierarchy with values from an artificial specificity measure. The dotted lines represent hypernym/hyponym pairs for which the specificity relationship according to this measure is incorrect. In the figure, there are 7 pairwise comparisons of which 3 are incorrect, for a score of 57%.

The percentage of pairwise comparisons each specificity measure placed correctly is presented in Table 4.1.

Clearly the better measures are performing much better than a random-guess algorithm which would give 50% performance.

Among the measures based on the parsed text (P_{mod} and its components and H_{mod}),

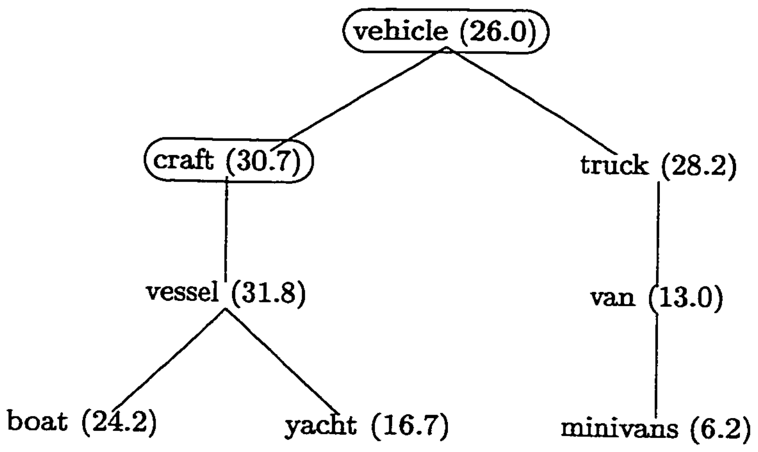

Figure 4.2: A piece of the "vehicles" hierarchy showing errors on the all-descendants correctness measure.

the entropy-based measure H_{mod} is clearly the best performer, as would be expected. However, it is interesting to note that the statistics based on adjectives alone (P_{adj}) somewhat outperform those based on all of our prenominal modifiers (P_{mod}). The reasons for this are not entirely clear.

Although H_{mod} is the best performer on the vehicles data, $freq$ and H_{50} do marginally better overall, with each of these three having the best results on one of the data sets. All three of these measures, as well as H_2 and H_{10}, get above 80% correct on average.

In these evaluations, it became clear that a single bad node high in the hierarchy could have a large effect on the results. For example, in the "occupations" hierarchy, the root node is "person," however, this is not a very frequent word in the Wall Street Journal corpus and rates as fairly specific across all of our measures. Since this node has eight children, a single bad value at this node can cause eight errors. We therefore considered another evaluation measure: for each internal node in the tree, we evaluated whether each specificity measure rated this word as more general than *all* of its descendants. (This is somewhat akin to the idea of edit distance. If we sufficiently increased the generality measure for each node marked incorrect in this system, all relationships in the hierarchy would be exactly correct.) Figure 4.2 shows the hierarchy from Figure 4.1 again, but in Figure 4.2, the nodes which are not in the correct relationship to all descendants are circled. In the figure, there are 5 internal nodes of which 2 have errors, for a score of 60%.

Although this is a harsher measure, it isolates the effect of individual difficult internal nodes. The results for this evaluation are presented in Table 4.2.

Although the numbers are lower in Table 4.2, the same measures as in Table 4.1 perform

Specificity measure	Vehicles	Food	Occupations	Average
P_{mod}	44.4	57.9	53.3	51.9
P_{adj}	33.3	52.6	60.0	48.7
P_{vb}	33.3	21.1	40.0	31.5
P_{nn}	55.6	21.1	33.3	36.6
H_{mod}	77.8	63.2	66.7	69.2
H_2	66.7	57.9	60.0	61.5
H_{10}	66.7	63.2	60.0	63.3
H_{50}	66.7	73.7	60.0	66.8
$Freq$	66.7	63.2	60.0	63.3

Table 4.2: Percentage of internal nodes having the correct relationship to all of their descendants.

relatively well. However, here H_{mod} has the best performance both on average and on two of three data sets, while the $freq$ measure does a bit less well, now performing at about the level of H_{10} rather than H_{50}. The fact that some of the numbers in Table 4.2 are below 50% should not be alarming, as the average number of descendants of an internal node is over 5, implying that random chance would give performance well below the 50% level on this measure.

Some of these results are negatively affected by word-sense problems. Some of the words added from the WordNet data are much more common in the Wall Street Journal data for a different word sense than the one we are trying to evaluate. For example, the word "performer" is in the occupations hierarchy, but in the Wall Street Journal this word generally refers to stocks or funds (as "good performers," for example) rather than to people. Since it was our goal to avoid using any outside sources of semantic knowledge, these words were included in the evaluations. However, if we eliminate those words, the results are as shown in Tables 4.3 and 4.4.

It is possible that using some kind of automatic word-sense disambiguation while gathering the statistics would help reduce this problem. This is also an area for future work. However, it should be noted that on the evaluation measures in Tables 4.3 and 4.4, as in the first two tables, the best results are obtained with H_{mod}, H_{50} and $freq$.

The above results are primarily for nouns at "basic level" and below (see, e.g., Lakoff ([17]) for a discussion of basic level nouns), a group which includes the vast majority of nouns. We also considered a data set at basic level and above, with "entity" at its root. Table 4.5 presents the results of testing on this data set and each specificity measure, for the evaluation measures described above, percentage of correct parent-child relationships

Specificity measure	Vehicles	Food	Occupations	Average
P_{mod}	65.0	62.5	67.7	65.1
P_{adj}	70.0	66.7	71.0	69.2
P_{vb}	80.0	43.8	48.4	57.4
P_{nn}	70.0	56.3	51.6	59.3
H_{mod}	100.0	81.3	74.2	85.1
H_2	95.0	79.2	77.4	83.9
H_{10}	95.0	79.2	77.4	83.9
H_{50}	95.0	85.4	77.4	85.9
$Freq$	95.0	83.3	80.6	86.3

Table 4.3: Percentage of correct parent-child relationships when words with the wrong predominant sense are removed.

Specificity measure	Vehicles	Food	Occupations	Average
P_{mod}	50.0	55.6	61.5	55.7
P_{adj}	33.3	50.0	61.5	48.3
P_{vb}	33.3	16.7	38.5	29.5
P_{nn}	66.7	22.2	30.8	39.9
H_{mod}	100.0	55.6	76.9	77.5
H_2	83.3	55.6	69.2	69.4
H_{10}	83.3	61.1	61.5	68.7
H_{50}	83.3	72.2	69.2	74.9
$Freq$	83.3	61.1	69.2	71.2

Table 4.4: Percentage of internal nodes with the correct relationship to all descendants when words with the wrong predominant sense are removed.

Specificity measure	Parent-child	All descendants
P_{mod}	59.1	46.4
P_{adj}	60.2	46.4
P_{vb}	50.0	35.7
P_{nn}	50.0	28.6
H_{mod}	59.1	39.3
H_2	53.4	25.0
H_{10}	45.5	32.1
H_{50}	46.6	32.1
$Freq$	45.5	32.1

Table 4.5: Evaluation of the various specificity measures on a test set of more general nouns.

Specificity measure	WordNet subtrees	WordNet subtrees with correct sense only	Very general WordNet subtree
H_{mod}	81.2	85.1	59.1
H_{50}	82.8	85.9	46.6
$Freq$	83.1	86.3	45.5

Table 4.6: Summary of results for the three best noun specificity measures.

and percentage of nodes in the correct relationship to all of their descendants.

Note that on these nouns, $freq$ and H_{50} are among the worst performers; in fact, by looking at the parent-child results, we can see that these measures actually do worse than chance. As nouns start to get extremely general, their frequency appears to actually decrease, so these are no longer useful measures. On the other hand, H_{mod} is still one of the best performers; although it does perform worse here than on very specific nouns, it still assigns the correct relationship to a pair of nouns about 59% of the time.

Table 4.6 summarizes the results on the parent-child evaluations for our three best specificity measures. Although H_{50} and $freq$ perform as well or better on words below basic level, and $freq$ in particular is a much simpler measure to compute, H_{mod} is also useful near the root of the tree where the other measures perform poorly. The H_{mod} measure, then, is the one we should use in the construction of a noun hierarchy.

Chapter 5

Incorporating the specificity data

Our co-occurrence data contain pairs of nouns and hypernyms (n, h). In fact, h is not always a hypernym of n. Aside from simple parsing or data collection errors, which also affect our other co-occurrence data, there is a potential source of error unique to the case of hypernyms. As discussed in the previous chapter, some of the apparent "hypernyms" we collect are actually hyponyms, so that we may have "food" as a hypernym of "product." (Since we are only using head nouns, dropping a modifier, such as the "pasta" in "pasta products," may turn a very specific noun phrase into a much more general one, reversing the relationship between h and n.)

Previously, we assumed that all hypernym data items represented a true hypernym relationship; that is, the probability that h was a hypernym of n was 1. Now, we assume that h is either a hypernym or a hyponym of n. For (n, h) pairs where we have seen data items, we want to multiply the counts by the probability that the relationship actually holds in the correct direction. From the previous chapter, the H_{mod} specificity measure can identify the more specific of two nouns with about 85% accuracy (assuming the predominant word senses in the specificity data are the word senses we are interested in, a reasonable assumption in this case since our hypernym data come from the same corpus). Therefore, we multiply our hypernym counts by 0.85 if the specificity measure indicates that h is likely more general than n and by 0.15 if the reverse is true.

5.1 Evaluation

We constructed a tree using the KL divergence-based clusterer and these weighted numbers and compared it to our previous KL divergence-based tree. The results are presented in Table 5.1.

Three best hypernyms	Hypernym 1		Any hypernym	
	majority	any	majority	any
investor/ally/bank	17	17	17	17
country/nation/producer	16	19	16	19
chain/company/airline	2	14	3	14
official/professional/investor	10	18	15	19
executive/officer/analyst	17	17	17	17
spirits/product/brand	14	17	15	18
area/factor/issue	2	18	2	20
charge/cost/benefit	1	3	1	3
product/compound/fiber	20	20	20	20
product/equipment/industry	13	20	14	20
AVERAGE	11.2 / 56.0%	16.3 / 81.5%	12.0 / 60.0%	16.7 / 83.5%
previous results	10.6 / 53.0%	14.4 / 72.0%	11.7 / 58.5%	15.2 / 76.0%

Table 5.1: The results of the judges' evaluation for the tree-builder including specificity statistics.

As the table illustrates, incorporating the specificity data improves the quality of our noun hierarchy. We can see improvements in all areas, although the largest improvements are in the columns indicating that any judge considered a hypernym correct. This suggests that rather than adding new good hypernyms, the specificity statistics are primarily helping us by screening out bad candidates; the improvement is the result of replacing unquestionably bad candidates with somewhat better ones, requiring more of a judgment call. It may be the case that given the sparseness of our hypernym data, we have already found most of the unquestionably good hypernym labels available. The issue of hypernym data sparseness is addressed in the next chapter as an area for future research.

It is also interesting at this point to go back and compare these results to those from our original algorithm, presented in Table 2.2. Besides the obvious quantitative differences, we can see a qualitative difference in the hierarchies produced. In our original experiment, the "Hypernym 1" and "Any hypernym" were extremely different. However, in our current results, they are virtually identical. We are now apparently doing a very good job in ranking the candidate hypernyms, and our performance is likely being hindered primarily by the lack of better hypernyms in our data.

Chapter 6

Discussion and future directions

The hierarchy presented here performs reasonably well compared to previous algorithms for constructing semantic lexicons, but clearly more work is needed in order to produce truly useful semantic hierarchies without requiring a great deal of manual post-editing.

6.1 Improving the hypernym labels

As discussed in the previous chapter, our technique seems to be quite successful in choosing among the available candidate hypernyms. Many of our erroneous hypernym labels in the hierarchy are simply a result of not having better choices available in the hypernym data. A major factor in this is the sparseness of the text patterns we are using to collect our hypernym statistics; out of the 3,467 nouns for which we have hypernym statistics, only 48 occur with any hypernym more than once in our corpus. An obvious help would be to look for other patterns in the text as suggested by Hearst ([14] and [15]). However, our technique will actually allow for richer sources of data than Hearst's work can utilize, because she considers a single instance of a text pattern as definitive evidence of a hypernym relationship, and therefore she only considers patterns that almost always indicate hypernyms. Our statistical framework will permit noisier sources of data to be incorporated, using ideas similar to those used to incorporate the specificity statistics. We can also use a larger corpus to simply collect a larger amount of data.

Occasionally, a cluster is badly mislabeled based on a very small fraction of the data in the cluster. For example, our technique often constructs very nice clusters of cities, but "city" is not a very common hypernym. Since some cities share their names with companies, a city cluster is often mislabeled "company" based on just a few cluster members. It might be useful to have some stricter criterion for hypernyms used to label a node, say, that they

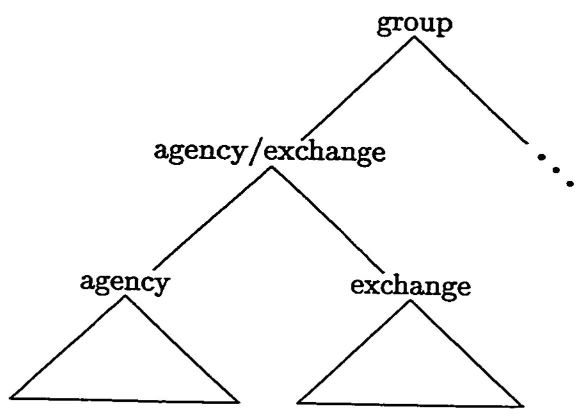

Figure 6.1: A tree with an artificial intermediate node.

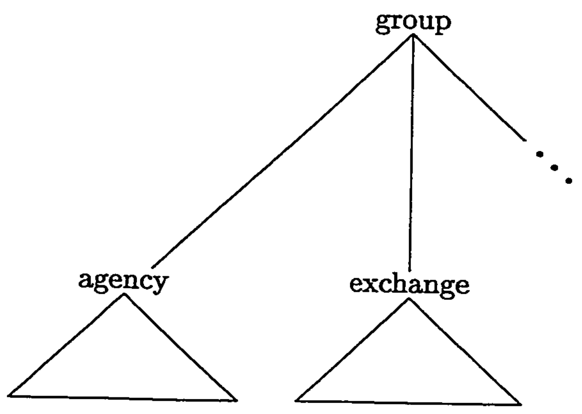

Figure 6.2: The tree of Figure 6.1 with the intermediate node removed.

occur with a certain percentage of the nouns below them in the tree. Additional hypernym data would be particularly helpful in this case.

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, a cluster with best hypernym "agency" and one with best hypernym "exchange" (as in "stock exchange") can have a parent with the two best hypernyms "agency/exchange" as shown in Figure 6.1, rather than both of these nodes simply being attached to the next level up with best hypernym "group," as shown in Figure 6.2. It might be possible to correct for this situation by changing how the hypernym distributions of a subtree are combined to form the distribution for an internal node; for example, if there is little overlap between the distributions of the subtrees, more general words could be weighted more heavily in the parent distribution.

Another interesting idea which may result in a somewhat cleaner hierarchy is to require a hypernym to appear in only one internal node of the tree. Currently, a high-frequency, very general word like "product" may appear as a hypernym in an arbitrary number of places in the hierarchy, barring only those duplications that are caught by the compression algorithm. Requiring this word to appear in only one place could allow lower-frequency but more specific hypernyms to be used instead.

6.2 Multiple-word phrases

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 ([14]) 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, because we would have data on other hypernyms of "bone" or other words in its cluster and 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. Comparison of the H_{mod} measure discussed here with something like the P_{mod} measure may help in the identification of multiple word phrases, by identifying nouns which are frequently modified using the same modifiers (rather than a variety of modifiers). A discussion of the difficulties in deciding how much of a noun phrase to use can be found in Hearst. Roark and Charniak ([29]) present one possible solution; Daille ([8]) evaluates several techniques to identify multiple-word terms and recommends the use of Dunning's Loglike statistic ([9]).

6.3 Word-sense disambiguation

Ideally, a useful hierarchy should allow for multiple senses of a word, and this is an area which may 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.

Word-sense disambiguation could be done either before the noun clustering takes place or by the clustering algorithm itself. In a preprocessing method, unsupervised methods could be used to identify words having multiple senses, and to cluster the occurrences of these words according to senses. Various methods for word-sense disambiguation are discussed in the following chapter. The distinct senses of a word would then be treated as entirely separate words in the clustering algorithm.

Alternatively, multiple senses of a word could be identified by a modified version of the clustering algorithm. The clustering algorithm used now places each word deterministically in a single cluster. A probabilistic algorithm could allow the word instead to belong to multiple classes simultaneously.

6.4 Using unparsed text

We used parsed text for these experiments because we believed we would get better results and the parsed corpus 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 ([1]), particularly since parsing is such a time-consuming process. Both Hearst ([14] and [15]) and Riloff and Shepherd ([28]) use unparsed text. However, the text processing would have to be done very precisely, as we have seen the difference that parse quality makes in the constructed hierarchy.

Chapter 7

Related work

Pereira et al ([26]) use 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 alone rather than also on the noun-noun relations that we use. 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 strictly on the verb-object relations.

Brown et al ([3]) use a bottom-up clustering method to build a decision-tree based language model for speech recognition. Their system includes the idea of running the algorithm on a subset of size k, clustering to $k/2$ clusters, and repeating, which was developed independently in this project. Jelinek ([16]) devotes a chapter to language models for speech recognition that are based on decision trees, including [3]. The discussion of smoothing these language models in section 10.14.1 discusses "bottom-up smoothing," which is the same smoothing technique developed independently for the probability model presented here, although he suggests tying the smoothing parameters based on tree depth rather than on data frequency as done here.

Riloff and Shepherd ([28]) suggest using conjunction and appositive data to cluster nouns; however, they approximate these co-occurrence counts by just looking at the nearest NP on each side of a particular NP. Roark and Charniak ([29]) build on that work by actually using conjunction and appositive data for noun clustering, as we include 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.

In addition to the papers cited above, various statistical techniques for clustering nouns have been investigated. Some recent results are presented by Lin([20] and [21]) and Li and Abe ([18]).

Hearst ([14] and [15]) introduces 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.

The sparseness of Hearst's patterns prevents that technique from being an effective approach to the problem of determining the specificity of nouns. To the best of our knowledge, the work described here and in [5] is the first research to address that problem.

Sanderson and Croft ([30]) automatically construct concept hierarchies for information retrieval using statistical techniques based on "subsumption" rather than clustering. The terms for the hierarchy are selected in advance, and then hierarchical relationships between them are found based on co-occurrences among document sets.

There is a great deal of literature describing statistical techniques for word-sense disambiguation. Gale et al ([11]) use a corpus of parallel English and French text as training data for this problem. For each occurrence of a particular ambiguous word in the English corpus, they create a vector of the words appearing within a 100-word window of that word, along with the correct sense according to the French translation. Using these statistics, they are able to identify the correct sense of new occurrences of these words about 90% of the time. Yarowsky ([35]) uses a similar technique to assign occurrences of words to one of the 1042 classes of words in Roget's International Thesaurus. He uses the Roget lists of words in each class and a text corpus to gather 100-word windows for each class.

Schütze ([32] and [31]) reports on a related experiment which uses unsupervised learning methods. He creates vectors of not the words appearing within a 100-word context window of the target word, but words occurring within a 100-word window of those context words, to help overcome sparse data problems. He then clusters these vectors using a cosine-based similarity metric (as we do in Chapter 2) to identify multiple senses of the word, and these clusters are used to classify new occurrences of the word, with 89% to 95% accuracy on the ten test words he evaluates. Yarowsky ([36]) gives an unsupervised method that uses heuristics to identify a small number of training examples and then uses bootstrapping to create more training data, from which a decision list is learned. Yarowsky's method rivals the performance of supervised methods, performing at about 96% accuracy on twelve tested words.

Resnick ([27]) uses verb-object data from parsed text to identify the correct WordNet sense for a given occurrence of a noun. Pereira et al ([26]) also show how verb-object data could be used to create a hierarchy of word senses for a particular word rather than a hierarchy of different words.

Statistical word-sense disambiguation is an active area of research. Recent work includes Ng and Lee ([24]) and Ng([23]), Dagan et al ([7]), and Pedersen and Bruce ([25]).

Various other authors have used statistical techniques to learn lexical semantic information. Light ([19]) uses morphological data, specifically, statistics on the affixes a word can take, to identify semantic features of that word. Hatzivassiloglou and McKeown ([13]) use statistical methods to identify whether adjectives have positive or negative orientation, starting from a small hand-labeled list of adjectives and using the fact that conjoined adjectives typically have the same orientation. Siegel ([33]) uses various machine learning techniques to combine evidence for classifying verbs according to a group of semantic features.

Chapter 8

Conclusions

We have shown that hypernym hierarchies of nouns can be constructed automatically from co-occurrence statistics on text. The quality of the hierarchies we can construct exceeds that of previous work in which semantic lexicons are built automatically for hand-selected hypernyms.

Since hypernyms are defined in terms of judgments made by native speakers, human judgments are the most appropriate method of evaluating these hierarchies. Our current best method builds a hierarchy which has been evaluated by human judges to produce clusters whose "best" hypernym label is correct for at least 56% of the cluster members. In addition, one of the top three hypernym labels is judged correct by at least one judge for 83.5%.

The quality of the parser used to parse the corpus from which the co-occurrence statistics are gathered is quite important. A state-of-the-art parser is currently necessary to achieve good results.

We have also shown how our hierarchy can be used as a probability model to predict co-occurrence data; however, it is important to note that predicting held-out data well is not a sufficient test of the hierarchy's quality. There is currently no substitute for human judges in evaluating the quality of noun hierarchies.

Statistics on a text corpus can also be used to compare nouns in terms of their specificity. We have identified several measures which do this task reasonably well, including H_{mod}, the entropy of the noun's rightmost modifier, which works reasonably well even for very general nouns, unlike heavily frequency-dependent measures. This specificity measure can be used to improve the quality of automatically constructed noun hierarchies.

Our techniques appear to be highly successful at choosing hypernyms from among the available candidates. When we address the hypernym data sparseness issue in future work, we expect a substantial improvement in our results.

The semantic noun hierarchy constructed by the methods presented in this thesis is a dramatic improvement over our previously published results. We previously reported a correctness measure of 33% ([4]); with the improvements described here, we have achieved 56%, and expect to be able to build even further on these results in future work.

Appendix A

Specificity data sets

The following pages show the data sets used in the specificity experiments. Words shown in *italics* are omitted from the results in Tables 4.3 and 4.4 because the predominant sense in the Wall Street Journal text is not the one represented by the word's position in this hierarchy.

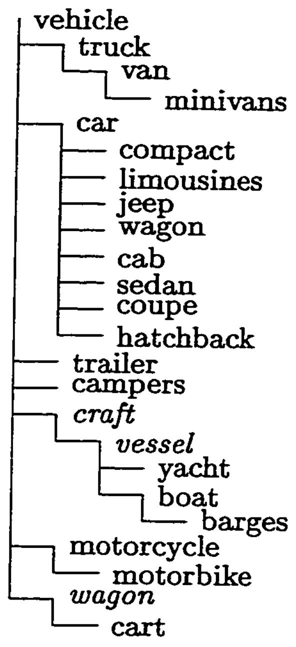

Figure A.1: The “vehicles” data set.

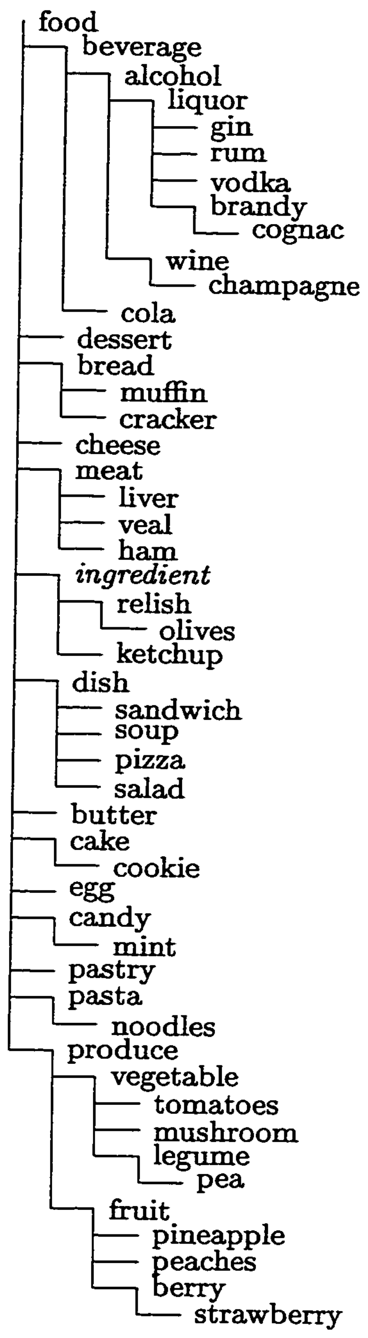

Figure A.2: The "food" data set.

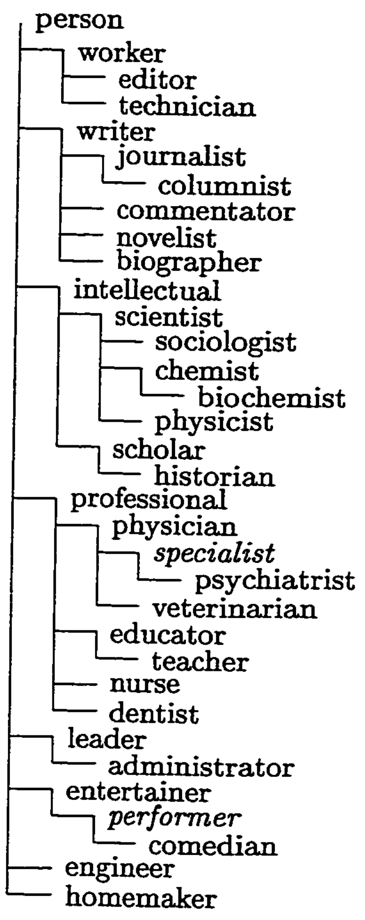

Figure A.3: The "occupations" data set.

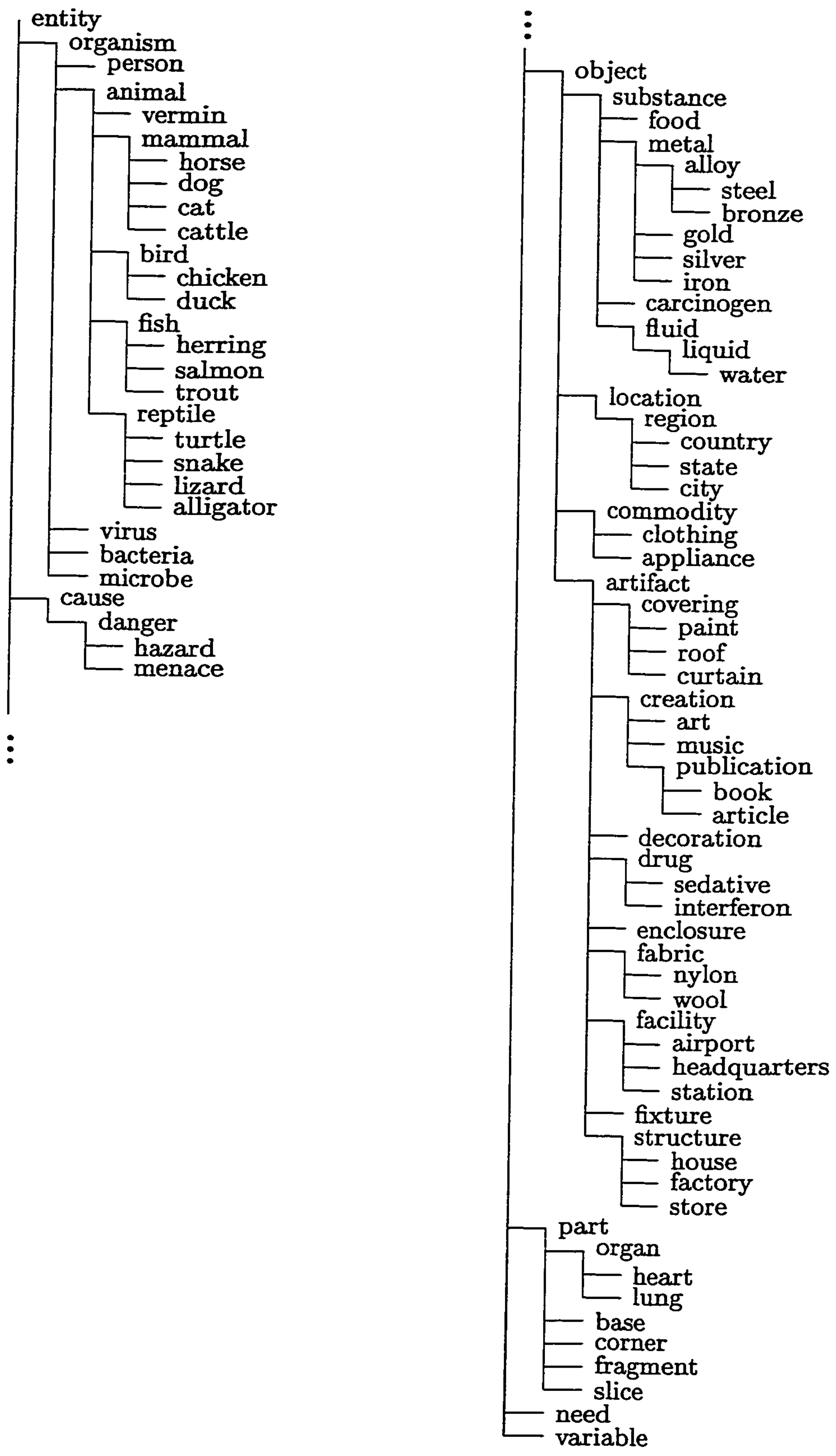

Figure A.4: The high-level data set.

Bibliography

[1] Thomas Ahlswede and Martha Evens. 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, 1988.

[2] L.E. Baum, T. Petrie, G. Soules, and N. Weiss. A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains. *Annals of Mathematical Statistics*, 41:164–171, 1970.

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

[4] Sharon A. Caraballo. Automatic construction of a hypernym-labeled noun hierarchy from text. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, 1999.

[5] Sharon A. Caraballo and Eugene Charniak. Determining the specificity of nouns from text. In *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, 1999.

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

[7] Ido Dagan, Lillian Lee, and Fernando Pereira. Similarity-based methods for word sense disambiguation. In *35th Annual Meeting of the Association for Computational Linguistics and 8th Conference of the European Chapter of the Association for Computational Linguistics: Proceedings of the Conference*, pages 56–63, 1997.

[8] Béatrice Daille. Study and implementation of combined techniques for automatic extraction of terminology. In *The Balancing Act: Combining Symbolic and Statistical Approaches to Language*, pages 49–66. The MIT Press, 1996.

[9] Ted Dunning. Accurate methods for the statistics of surprise and coincidence. *Computational Linguistics*, 19:61–76, 1993.

[10] Christiane Fellbaum, editor. *WordNet: An Electronic Lexical Database.* The MIT Press, 1998.

[11] William A. Gale, Kenneth W. Church, and David Yarowsky. A method for disambiguating word senses in a large corpus. *Computers and the Humanities*, 26:415–439, 1993.

[12] Gregory Grefenstette. SEXTANT: Extracting semantics from raw text implementation details. *Heuristics: The Journal of Knowledge Engineering*, 1993.

[13] Vasileios Hatzivassiloglou and Kathleen R. McKeown. Predicting the semantic orientation of adjectives. In *35th Annual Meeting of the Association for Computational Linguistics and 8th Conference of the European Chapter of the Association for Computational Linguistics: Proceedings of the Conference*, pages 174–181, 1997.

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

[15] Marti A. Hearst. Automated discovery of WordNet relations. In Fellbaum [10], pages 131–151.

[16] Frederick Jelinek. *Statistical Methods for Speech Recognition.* The MIT Press, 1997.

[17] George Lakoff. *Women, Fire, and Dangerous Things: What Categories Reveal about the Mind.* University of Chicago Press, 1987.

[18] Hang Li and Naoki Abe. Word clustering and disambiguation based on co-occurrence data. 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 749–755, 1998.

[19] Marc Light. Morphological cues for lexical semantics. In *Annual Meeting of the Association for Computational Linguistics: Proceedings of the Conference*, pages 25–31, 1996.

[20] Dekang Lin. Using syntactic dependency as local context to resolve word sense ambiguity. In *35th Annual Meeting of the Association for Computational Linguistics and 8th Conference of the European Chapter of the Association for Computational Linguistics: Proceedings of the Conference*, pages 64–71, 1997.

[21] Dekang Lin. Automatic retrieval and clustering of similar words. 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 768–774, 1998.

[22] Christopher D. Manning and Hinrich Schütze. *Foundations of Statistical Natural Language Processing.* The MIT Press, 1999.

[23] Hwee Tou Ng. Exemplar-based word sense disambiguation: Some recent improvements. In *Proceedings of the Second Conference on Empirical Methods in Natural Language Processing*, pages 208–213, 1997.

[24] Hwee Tou Ng and Hian Beng Lee. Integrating multiple knowledge sources to disambiguate word sense: An exemplar-based approach. In *Annual Meeting of the Association for Computational Linguistics: Proceedings of the Conference*, pages 40–47, 1996.

[25] Ted Pedersen and Rebecca Bruce. Distinguishing word senses in untagged text. In *Proceedings of the Second Conference on Empirical Methods in Natural Language Processing*, pages 197–207, 1997.

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

[27] Philip Resnick. Wordnet and distributional analysis: a class-based approach to lexical discovery. In *AAAI Workshop on Statistically-Based Natural Language Processing Techniques*, pages 54–64, 1992.

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

[29] Brian Roark and Eugene Charniak. 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, 1998.

[30] Mark Sanderson and Bruce Croft. Deriving concept hierarchies from text. In *Proceedings of the 22nd ACM SIGIR Conference*, pages 206–213, 1999.

[31] Hinrich Schütze. Context space. In *Working Notes, AAAI Fall Symposium Series, Probabilistic Approaches to Natural Language*, pages 113–120, 1992.

[32] Hinrich Schütze. Word sense disambiguation with sublexical representations. In *AAAI Workshop on Statistically-Based Natural Language Processing Techniques*, pages 109–113, 1992.

[33] Eric V. Siegel. Learning methods for combining linguistic indicators to classify verbs. In *Proceedings of the Second Conference on Empirical Methods in Natural Language Processing*, pages 156–162, 1997.

[34] Karen Sparck Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 28:11–21, 1972.

[35] David Yarowsky. Word-sense disambiguation using statistical models of Roget's categories trained on large corpora. In *Proceedings of the 14th International Conference on Computational Linguistics (COLING-92)*, pages 454–460, 1992.

[36] David Yarowsky. Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, pages 189–196, 1995.