SEMANTIC TAXONOMY INDUCTION

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Rion Langley Snow
June 2009

Abstract

Understanding natural language has been a longstanding dream of artificial intelligence, and machine learning offers a new perspective on this old problem. This work addresses four key problems in automatically reading and understanding text: extracting the knowledge expressed in a body of text in the form of structured relations, reconciling and formalizing that knowledge in a fully consistent, sense-disambiguated hierarchy of knowledge, fluidly transitioning from fine-grained to coarse-grained distinctions between word senses, and applying extracted structured knowledge in applications that depend on deep textual understanding.

Textual patterns have frequently been devised to identify specific instances of world knowledge in text. For example, from the text "such fruits as apples and oranges" one might infer the knowledge that "apples and oranges are kinds of fruit". In this work we discuss the use of *distant supervision* for relation extraction, which applies machine learning techniques to a set of example relation instances and a large body of unannotated text in order to rediscover many of the textual patterns formerly proposed in the information extraction literature, along with hundreds of thousands of previously unconsidered patterns. Further, we apply these automatically discovered patterns to extract structured knowledge from newswire articles and other text, significantly outperforming hand-designed patterns and discovering hundreds of thousands of novel examples of world knowledge not previously encoded in manually-created knowledge bases.

Many proposed methods for extracting structured knowledge suffer from a critical inability to deal with redundancy or contradictory extractions. While modern algorithms can often suggest millions of possible facts extracted from a large body of text, they are unable to reconcile this extracted knowledge into a set of consistent, sense-disambiguated assertions. We propose a probabilistic framework for taxonomy induction that solves each of these problems, taking advantage of the full set of predicted facts and any knowledge already known in an existing taxonomy. This work has resulted in one of the largest automatically-constructed augmentations of the WordNet knowledge base currently in existence.

In addition to the automatic augmentation of knowledge resources, we explore the task of automatically creating coarse-grained taxonomies. It has been widely observed that different natural language applications require different sense granularities in order to best exploit word sense distinctions, and that for many applications WordNet senses are too fine-grained. In contrast to previously proposed automatic methods for sense clustering, we formulate sense merging as a supervised learning problem, exploiting human-labeled sense clusterings as training data. Our learned similarity measure outperforms previously proposed automatic methods for sense clustering on the task of predicting human sense merging judgments. Finally, we propose a model for clustering sense taxonomies using the outputs of this classifier, and we make available several automatically sense-clustered WordNets of various sense granularities. These resources offer the capability of tailoring a knowledge resource to the sense granularity most suited to a particular application.

Our framework for taxonomy induction lays the groundwork for new semantic applications, including inferring domain-specific hierarchies of knowledge and augmenting foreign-language Wordnets. Finally, we demonstrate that our automatically augmented taxonomies significantly outperform manually-constructed resources across several natural language tasks, including relation prediction, question answering, and text categorization.

Acknowledgements

I dedicate this work to my parents, Kathy and Robert, for giving me life. To my advisors, Andrew Ng and Dan Jurafsky, for giving me sunlight, nourishment, and room to grow. To Tom Bartol, Ted O'Connor, and Balaji Srinivasan, for inspiring me to create.

To my scientific and philosophical mentors: to John Frank, for showing me the deeply intertwingled nature of wisdom and fun. To Nancy Johnson, for encouraging me to experiment with dangerous ideas. To Bob Severson, for convincing me that the physical world might just be weird enough to be understood. To Linda Petrait, for pouring fuel on the fire. And to Rachel Kalmar, for, well, everything.

To my community, my *sangha*, my mind tribe: you are my heroes. I'm smiling right now, and it's because I'm thinking of you. You have opened my mind, and my heart, and I owe you the world for it. Thank you.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The acquisition, organization, and application of machine-readable representations of knowledge have been fundamental challenges since the inception of the field of artificial intelligence. These goals have been the motivating force underlying numerous projects in lexical acquisition, information extraction, and the construction of semantic taxonomies. Broad-coverage semantic taxonomies such as WordNet [50] and Cyc [78] have been manually constructed at great cost to address these challenges. These lexical resources have become crucial sources of structured information about relations between words and concepts, and are used in a wide variety of applied tasks, from reasoning in expert systems to computing measures of word similarity. Despite their wide use, these taxonomies suffer from sparse coverage and limited usefulness in a variety of domains. Further, as new textual information continues to be produced at an ever increasing rate, the problem of manually annotating each new expression and resolving it with existing knowledge becomes overwhelming. While this explosion in information production presents an unmanageable obstacle to manual annotation, it represents an exciting opportunity for automated data mining methods. Taking inspiration from the rapidly increasing amount of publicly-available textual data and the decreasing cost of computation, this work focuses on developing machine learning algorithms for automatically acquiring and organizing structured knowledge that can be most effective when confronted with millions or even billions of sentences of text.

Representing knowledge as part of a semantic taxonomy confers numerous advantages for applications in natural language processing; however, the problem of incorporating new concepts and relations into a semantic taxonomy comes with an array of challenges not adequately addressed by modern information extraction and relationship extraction algorithms. These challenges include incorporating multiple sources of evidence for several different relations, disambiguating the correct senses of the words between which new relationships are added, and maintaining the consistency of the taxonomy as a whole. Previous work has typically either focused on only inferring small taxonomies over a single relation, or has used evidence for multiple relations independently from one another. Additionally, previous techniques for taxonomy induction have often sidestepped the issue of lexical ambiguity by making the assumption of only a single sense (or single most frequent sense) per word, thus inferring taxonomies over words rather than the larger space of concepts to which those words refer.

In this work we propose a probabilistic framework for taxonomy induction that solves each of these problems using a unified formulation of taxonomy likelihood. This framework is not restricted by the existence of multiple senses and multiple relations, but rather is capable of taking advantage of the information already present in an existing taxonomy to make better decisions about where to add new concepts and how to structure new links between concepts. The approach we propose simultaneously provides a solution to the problems of both lexical ambiguity and the integration of heterogenous sources of evidence within a single flexible, probabilistic framework.

We show that this framework may be applied not only to acquiring novel concepts from unstructured text, but also to other modifications of taxonomic information. We describe several operations for restructuring taxonomies, including adding and removing relations, and adding, removing, and merging concepts. We demonstrate the application of these operations through several experiments. In these experiments we have automatically created modified versions of the WordNet taxonomy, including several augmented and sense-merged taxonomies. Finally, we demonstrate the application of our framework to broader problems within natural language processing, including question answering, and text categorization.

1.1 Relation extraction

A major goal of natural language processing and computational semantics in particular is the creation of an automated method for capturing a structured representation encompassing the meaning of a chosen segment of natural text. The most ambitious of these attempts have aimed at representing the complete meaning and full implication of arbitrary expressions of natural text, often in the form of logical statements. One introduction to a formal approach of representing natural language sentences in first-order logic is expressed by Blackburn and Bos [13, 14]. Because even representing the full meaning of an arbitrary piece of text (much less automatically extracting it) is extremely difficult, much work in this area has concentrated on extracting only a limited form of the knowledge implied in a given text. For example, much work in named entity recognition focuses specifically on only labeling the specific words or phrases in a text that refer to people, places, or organizations [12].

In this section we discuss previous work on the problem of extracting relational facts about text. For example, we might want to learn that a *chair* is a kind of *furniture*, or that a person is employed by a particular organization, or that a geographic entity is located in a particular region. Specifically we will discuss methods for extracting *binary* relations in text—that is, relations between exactly two entities. While some work has been extended to include *n-ary* or *complex* relations (relations between an arbitrary number of entities) [90], most work in relation extraction has focused on the case of binary relations. In this section we discuss four general approaches to relation extraction: the application of manually specified patterns, semi-supervised classification, fully-supervised classification, and unsupervised approaches.

1.1.1 Manually-specified patterns

The application of manually-specified patterns or heuristics has been among the most wide-spread methods for the automatic extraction of binary semantic relations from text. Here we discuss several pattern-based approaches that have been proposed for the *hypernym* (IS-A) relation, the *meronym* (part-whole) relation, and several others.

Hearst pattern	**Example occurrences**
NP_X and other NP_Y:	...temples, treasuries, and other important civic buildings.
NP_X or other NP_Y:	Bruises, wounds, broken bones or other injuries...
NP_Y such as NP_X:	The bow lute, such as the Bambara ndang...
Such NP_Y as NP_X:	...such authors as Herrick, Goldsmith, and Shakespeare.
NP_Y including NP_X:	...common-law countries, including Canada and England...
NP_Y, especially NP_X:	European countries, especially France, England, and Spain...

Table 1.1: Hearst's patterns and example occurrences

The *hypernym* relation

The IS-A or *hypernym* relation is among the most well-studied binary relations in lexical semantics. We state that a noun X is a *hyponym* of a noun Y (and, respectively, Y is a *hypernym* of X) if X is a subtype or instance of Y, that is, if X is a *kind of* Y. This is the relation indicated in the common sense knowledge that a *dog* IS-A *animal*, or that a *rose* IS-A *flower*, or that *water* IS-A *liquid*. Ordered pairs such as "*dog/animal*" are referred to as *hyponym/hypernym* pairs; *dog* is the *hyponym* of *animal*, and *animal* is the *hypernym* of *dog*. The hypernym relations between concepts make up much of the information contained in lexical ontologies such as WordNet[1] [50].

Much of the previous work in the automatic semantic classification of words and relations has been based on a key insight first articulated by Hearst [60]—that the presence of certain "lexico-syntactic patterns" can indicate a particular semantic relationship between two nouns. Hearst noticed that, for example, linking two noun phrases (NPs) via the constructions "Such NP_Y as NP_X", or "NP_X and other NP_Y", often implies that NP_X is a hyponym of NP_Y, i.e., that NP_X is a kind of NP_Y. The full set of patterns proposed by Hearst along with examples of their occurrence in natural text are given in Table 1.1.

In [60] Hearst searched for instances of the "NP_Y *such as* NP_X" pattern in Grolier's *Academic American Encyclopedia* [65], a digital encyclopedia. By extracting the corresponding noun phrases that occurred in this pattern, Hearst found several correct hypernym relations that were not already in WordNet. In a later work, Hearst

[1]Refer to Section 2.2 for a more detailed discussion of WordNet.

[61] applied the “NP_X *or other* NP_Y” pattern to a corpus of text from the *New York Times*, extracting 200 candidate sentences. The precision of the predicted hypernym pairs was then evaluated by hand. Of the 166 “eligible” sentences containing unique relations and conforming to the required syntax, the manual evaluation determined that 104 (or 63%) contained a hyponym/hypernym pair that was either already present in WordNet, or else was a “strong candidate for inclusion”.

Since Hearst’s original work, several researchers have used a small number (typically less than ten) of similar hand-crafted patterns to try to automatically label semantic relations [23, 32, 26, 48, 55, 108, 129]. Caraballo [23] makes use of two of the Hearst Patterns (“NP_X *and other* NP_Y” and “NP_X *or other* NP_Y”) as a key part of automatically inferring a WordNet-like noun hypernym hierarchy. First, a bottom-up clustering was performed over words with high distributional similarity[2]; then the Hearst patterns were used in an attempt to discover the potential hypernyms of the members of those clusters. In this way a hierarchy of over 20,000 nodes was created. In these experiments both the distributional similarity model and the hypernym extraction was performed on a corpus of text from the *Wall Street Journal*. A manual evaluation was then applied to 200 of the automatically extracted hyponym/hypernym pairs to determine the precision of the inferred hypernym hierarchy. The evaluation determined that, depending on how critical the criteria for hypernym correctness was, between 33% and 60.5% of the extracted hypernyms were correct.

An approach to labeling semantic classes using dependency path features is detailed by Pantel and Ravichandran [108]. In this work a 3GB text corpus was parsed using MINIPAR, a broad-coverage dependency parser[3] [81]. A distributional similarity model was built using the syntactic dependencies produced b MINIPAR as context, and group-average agglomerative clustering was then performed to create *committees* [105]. From the members of these committees a grammatical template or *signature* was created, aggregating the dependency paths across all members of that semantic class. These templates were then examined in order to determine which dependency

[2]Refer to Section 2.7 for a more detailed discussion of distributional similarity.

[3]Refer to Section 2.3.1 for a more detailed discussion of MINIPAR and dependency parsing.

Pattern type	**Dependency link**	**Example occurrences**
Apposition	NP_X:appo:NP_Y	... Oracle, a company known for...
Nominal subject	NP_Y:subj:NP_X	Apple was a hot young company...
"NP_Y such as NP_X"	NP_Y:such as:NP_X	...companies such as IBM...
"NP_Y like NP_X"	NP_Y:like:NP_X	...companies like Sun Microsystems...

Table 1.2: Pantel's patterns and example occurrences

paths had the highest mutual information[4] with the actual labels of the classes. Four features were found to have the highest mutual information: the apposition, nominal subject, *such as*, and *like* dependencies; these patterns are summarized along with examples in Table 1.2.

From the 3GB of newswire text, 1,432 total noun clusters were extracted, comprising 18,000 unique words. An attempt was then made to name each of these clusters, using the four dependency links above. The accuracy of the algorithm's hypernym prediction was then manually evaluated, resulting in an estimated accuracy in the range of 62.0% to 76.5%. An evaluation restricted to proper noun clusters significantly increased the judged accuracy, to the range of 79.0% to 85.5%. A further accomplishment of this work was the application of the automatically inferred hypernyms to applications in question answering and information retrieval; the applications demonstrated that the inferred hypernyms could significantly increase recall for both tasks.

In the work of Cimiano et al. [32], the Hearst patterns were used to identify potential hypernym pairs in two domain-specific text collections The extracted potential hypernym pairs were then filtered by using the Hearst patterns again, this time by searching for their occurrence on the World Wide Web using the Google search engine[5]. The total page count was requested via Google query for each Hearst pattern, using the predicted pair to fill in the NP slots of the pattern. This retrieved page count was then used as part of a calculation of the system's confidence for that hypernym relation. Using the Hearst patterns alone in tourism-specific corpora resulted in an

[4]Refer to Section 2.7.1 for a more detailed discussion of mutual information.
[5]Available at `http://www.google.com`.

Berland pattern	**Example occurrences**
NP_Y's NP_X:	...building's basement...
NP_X of {the|a} NP_Y:	...basement of a building...
NP_X in {the|a} NP_X:	...basements in a building...
NP_X of NP_Y:	...basements of buildings...
NP_X in NP_Y:	...basements in buildings...

Table 1.3: Berland's patterns and example occurrences

F-score[6] of more of 6.92%, with a precision of 29.1% and recall of 3.93%. These numbers were much improved by filtering through the Google API with the same Hearst patterns; this raised the results to an F-score of 18.85% at a precision of 15.77% and recall of 23.43%. Finally, by combining additional information the results were increased a small amount more, to an F-score of 21.81%, with a precision of 17.38% and recall of 23.43%.

Other relations

Other work has taken a similar pattern-based approach in attempts to identify the *meronym* or *part-whole* relation (e.g., a *wheel* is PART-OF a *car*) [11, 55, 54], biomedical relations [102, 34, 126, 53], and others.

As an example of work addressing the part-whole relation, Berland and Charniak [11] apply the five manually-specified patterns listed in Table 1.3 to a news corpus of approximately 100 million words. In a manual evaluation of the predicted part-whole relations, approximately 55% of the predicted relations were found to be correct.

Manually-specified patterns have also been proposed for extracting biomedical relations [102, 34, 126, 53]; for example, Fundel et al. [53] use heuristically-specified patterns over dependency parses to recognize relations between genes and proteins. Many approaches have also been applied to extracting a variety of basic semantic relations from machine-readable dictionaries (MRDs) [122, 5].

Other proposals have been made to use a manually-defined set of features for general purpose relation extraction. For example, Turney and Littman [147] propose

[6]Refer to Section 2.5 for a more detailed discussion of F-score, precision and recall.

NP_X NP_Y	NP_X become NP_Y
NP_X * not NP_Y	NP_X but not NP_Y
NP_X * very NP_Y	NP_X contain NP_Y
NP_X after NP_Y	NP_X for NP_Y
NP_X and not NP_Y	NP_X for example NP_Y
NP_X are NP_Y	NP_X for the NP_Y
NP_X at NP_Y	NP_X from NP_Y
NP_X at the NP_Y	NP_X from the NP_Y

Table 1.4: 16 of Turney's 128 patterns

64 "short phrases" between pairs of entities (creating 128 order-dependent patterns), and use these patterns to represent arbitrary binary relations in a high-dimensional vector space. 16 of Turney's 128 patterns are given in Table 1.4. These patterns contain many previously-discussed of the textual patterns [60, 11, 108], along with many others. This set of patterns was used to predict general binary semantic relations in an analogy-solving problem. By using these 128 patterns as a way of expressing the relation holding between two entities as evidenced in a large body of text, a similarity metric between arbitrary relations could be efficiently calculated.

1.1.2 Semi-supervised approaches

In addition to proposing six manually-specified patterns for extracting hyponym/hypernym pairs, Hearst [60] further suggested a powerful automatic algorithm for automatically determining patterns to detect arbitrary relations. The proposed algorithm is summarized below:

1. Decide on the lexical relation R of interest.
2. Gather a list of terms for which this relation is known to hold.
3. Find places in the corpus where these expressions occur syntactically near one another and record the environment.
4. Find the commonalities among these environments and hypothesize that common ones yield patterns that indicate the relation of interest.

```
Data: seed set of instances $S$, text corpus $D$
Result: a set of predicted instances $S$ and discovered features $F$
while stop-criterion-is-unmet do
    $F \leftarrow$ Extract-Features($S$, $D$)
    $S' \leftarrow$ Extract-Instances($F$, $D$)
    $S \leftarrow$ Union($S$, Conf($S'$))
end
```

Algorithm 1: Bootstrapping framework for relation extraction

5. Once a new pattern has been positively identified, use it to gather more instances of the target relation and go to Step 2.

This algorithm is implemented only by hand by Hearst [60]; the author states that she did not implement an automatic version of this algorithm, "primarily because Step 4 is underdetermined". Many researchers since then have provided ways for accomplishing Step 4—that is, ways to determine the aspects of the lexico-syntactic environments that are common and to choose how to use those aspects to identify new patterns. Many of these approaches fall under the domain of automatic feature discovery; we summarize several of these approaches in this section. Another survey of similar techniques may be found in [89].

The pseudocode in Algorithm 1 presents the underlying bootstrapping framework widely used in many applications of semi-supervised classification. Here the algorithm is initialized with some seed set of targets S; in the case of binary relation extraction, these seeds typically correspond to entity pairs known to be in a particular relation. The second required input is some existing text corpus D in which the seeds can be found. The algorithm then proceeds to extract some set of features F from appearances of the seed set S in the data set D, then re-use these features to extract an additional set of seeds S'. This new set of seeds is then typically pruned in some way, so that only the high-confidence seeds Conf(S') remain. These surviving seeds are combined with the previous set, and the procedure is repeated until some stop criterion is met.

Early uses of a bootstrapping framework in natural language processing include applications to word sense disambiguation [156], text classification [69], and lexicon induction [125]. Several algorithms have adapted this framework for extracting patterns for binary relation extraction, including Dipre, Snowball, and Espresso [15, 2, 108, 24].

A precursor to the use of pattern extraction for binary relation extraction may be found in the AutoSlog [123] and AutoSlog-TS systems [124]. These systems present some of the earliest examples of automatic pattern extraction. In these systems the goal is semantic lexicon induction. Rather than extracting pairs of entities that participate in some binary relation, the goal of these systems is to identify members of a particular semantic class. These systems may be seen as having the goal of *unary* relation extraction. The AutoSlog system was first applied to recognizing entities within the domain of terrorist event descriptions in the MUC-4 corpora [144], and was extended to general web text in [111]. Here the goal of the system is to extract such semantic classes as the *victim* and the *perpetrator* of attacks, and it is given a certain set of natural text with corresponding entities tagged along with their class. Thus, given an annotated sentence fragment like "terrorists bombed the U.S. embassy", AutoSlog would be able to extract the pattern "<perpetrator> bombed the" and "bombed the <victim>". In this way the system can then apply these automatically extracted patterns to discover new instances of victims and perpetrators, in a sense performing a unary relation extraction for the predicates victim(x) and perpetrator(x).

Dipre

Dipre, or Dual Iterative Pattern Relation Extraction, was an early technique proposed for general binary relation extraction by Brin [15]. The basic premise of Dipre follows the general bootstrapping pattern induction algorithm presented by Hearst [60], using a bootstrapping procedure for discovering features. Here the application of the pattern extraction algorithm is to extract (*Author*, *Book*) pairs from a web corpus of 24 million documents.

Dipre begins with some seed set of pairs of entities (e_1, e_2) known to be in the desired relation, and discovers a large number of occurrences of those pairs in a text corpus. The five (*Author*, *Book*) pairs used to initiate the algorithm are given in

Author	**Book**
Isaac Asimov	The Robots of Dawn
David Brin	Startide Rising
James Gleick	Chaos: Making a New Science
Charles Dickens	Great Expectations
William Shakespeare	The Comedy of Errors

Table 1.5: Example seed pairs for extracting the AUTHOR-BOOK relation with DIPRE

URL Prefix	**Text Pattern**
`www.sff.net/locus/c.*`	<LI><B>*title*</B> by *author* (
`dns.city-net.com/~lmann/awards/hugos/1984.html`	<i>*title*</i> by *author* (
`dolphin.upenn.edu/~dcummins/texts/sf-award.htm`	*author* || *title* || (

Table 1.6: Example extracted URL and text patterns using DIPRE

Table 1.5. First, the algorithm identifies documents in the corpus that contain at least one of the seed entity pairs, and extracts the string context around the pair within those documents. From these extracted contexts the system extracts a tuple of the form {*order*, *urlprefix*, *prefix*, *middle*, *suffix*}. Here *order* contains the single bit of whether the author or the book came first in the string; the *urlprefix* records the URL at which the author/book pair is found; and the *prefix*, *middle*, and *suffix* fields each record a particular aspect of the context around the entity pair: up to 10 characters of context to the left of the first entity, all the characters in between, and up to 10 characters of context to the right of the second entity, respectively.

In the application of DIPRE to a web corpus of 24 million pages, more than ten thousand (*Author*, *Book*) pairs were extracted starting with only five seed pairs. In the first iteration of the algorithm the five initial (*Author*, *Book*) seed pairs were found in 199 occurrences across the web corpus. From these occurrences three patterns were extracted satisfying the imposed requirements of redundancy and specificity; these three patterns are listed in Table 1.6. In the next stage of the iterative extraction, these 3 patterns were identified in other locations within the web corpus in order to discover new potential author-book pairs, yielding an additional 4047 author-book pairs. This process was then repeated, yielding the quantity of entity pairs and

Iter. #	# Entity pairs	#Corpus frequency	# Extracted patterns
1	5	199	3
2	4047	3972	105
3	9369	9938	346
4	15257	——	——

Table 1.7: Total number of entity pairs, pair occurrences, and extracted patterns at each iteration of DIPRE

extracted patterns detailed in Table 1.7: the columns here correspond to the iteration number, the total discovered unique entity pairs, the total counts observed in the web corpus of all extracted entity pairs, and the total count of extracted patterns from the set of entity pairs, respectively. Note that after the first iteration the patterns were extracted from successively smaller portions of the full text corpus; otherwise, the corpus frequency of the entity pairs would be a great deal higher. Brin [15] states that these extractions were of very high accuracy, and that the final extraction contained "very little bogus data". There was, however, a manual intervention in the extraction of the 9369 entity pairs in iteration number 3; in this case, 242 of the entity pairs were observed to contain the string *Conclusion* as the author. These were manually thrown out and the algorithm allowed to progress, but it is not reported whether leaving these mistakes in would have been disastrous for performance of future iterations of the algorithm.

Snowball

SNOWBALL [2] is an algorithm with a similar architecture to DIPRE [15], but with some improvements. Similar to DIPRE, SNOWBALL is initialized with a seed set of pairs representing the desired relation. Agichtein and Gravano [2] apply SNOWBALL to the task of extracting organizations and locations in the LOCATED-IN relation. The example seed set used is given in Table 1.8. SNOWBALL then searches over a corpus of text to identify sentences that contain the occurrences of at least one of the seed pairs. From these sentences SNOWBALL and extracts the [$left$, $middle$, $right$] contexts around the seed pair. Rather than limiting the context by a character limit,

Organization	**Location of Headquarters**
Microsoft	Redmond
Exxon	Irving
IBM	Armonk
Boeing	Seattle
Intel	Santa Clara

Table 1.8: Example seed pairs for extracting the LOCATED-IN relation with SNOWBALL

Example text	**Extracted patterns**
...Microsoft's headquarters in Redmond...	<*Organization*>'s headquarters in <*Location*>
...the Irving-based Exxon Corporation...	<*Location*>-based< *Organization*>
...Intel, Santa Clara, announced...	<*Organization*>, <*Location*>

Table 1.9: Example extracted patterns using SNOWBALL

SNOWBALL instead extracts a fixed number of terms on the left and right of the two entities, and the full context in the middle.

Since SNOWBALL focuses on the LOCATED-IN relation, it may take advantage of the semantic classes of the entities it is extracting, that is it may require that the entities must be locations and organizations. SNOWBALL makes use of this information by applying a named entity recognizer to the text it extracts as a preprocessing step, and including the named entity type as part of the patterns that it extracts. This allows SNOWBALL to require that the entities in the appropriate slots of the induced patterns are recognized as the appropriate named entity type by the named entity classifier; while this is not a generally-applicable technique for arbitrary binary semantic relations, this allows the extracted patterns for the LOCATED-IN relation to be a great deal more precise. SNOWBALL uses the MITRE Alembic Workbench [41] in order to perform named entity classification; the Alembic Workbench itself provides a semi-automatic bootstrapping framework to quickly learn new training examples for named entity recognition. Examples of some of the patterns extracted by SNOWBALL are given in Table 1.9.

Because of its iterative bootstrapping architecture, SNOWBALL is capable of extracting patterns and predicting tuples of new relations with very small amounts of seed data. However, training named entity classifiers like the one used by Snowball is typically more expensive, requiring a large training set of examples of organizations and locations in context. Nonetheless, this approach may still be feasible even for non-standard named entity classes. Since training data containing at least a single entity in a sentence is much more prevalent than data containing both entities in a relation pair, the prospect of collecting a larger amount of data to train an entity classifier over singletons may make an initial entity prediction step plausible for many classes of entities.

Agichtein and Gravano [2] further provide a performance comparison of Snowball, Dipre, and a baseline algorithm on the task of extracting the located-in relation using the five seed patterns in Table 1.8. Each algorithm is run for three iterations. The evaluation finds that Dipre diverges after the first iteration; both the precision and recall drop in the two successive iterations. In contrast, the Snowball and the baseline algorithm essentially converge after the first iteration. While this is a point in their favor (in that their performance does not degrade on successive iterations), it raises the question of whether the bootstrapping paradigm is actually helpful for this task. Snowball's stronger performance is attributable to its more selective method for weighting patterns and extracted entity pairs.

Espresso

Pantel and Pennacchiotti [107] propose Espresso, an algorithm that also follows the basic bootstrapping framework proposed by Hearst [60]. However, unlike Dipre and Snowball, Espresso does not impose any specific regular expression or named entity restrictions on the entities it extracts, and thus is more easily generalizable to arbitrary binary semantic relations. The primary difference between Espresso and previously discussed algorithms is in the weighting scheme it uses for for assessing the confidence of extracted entities and patterns. Espresso extracts features using the lexico-syntactic environment representation proposed in Ravichandran and Hovy [118]. Further, whereas Dipre and Snowball were evaluated on only a single relation, Pantel and Pennacchiotti [107] provide an evaluation of their system across five different relation types, as shown in Table 1.10.

Relation name	**Example seeds**
IS-A	[wheat / crop], [George Wendt / star]
PART-OF	[leader / panel], [city / region]
SUCCESSION	[Khrushchev / Stalin], [Carla Hills / Yeutter]
REACTION	[magnesium / oxygen], [hydrazine, water]
PRODUCTION	[bright flame / flares], [hydrogen / metal hydrides]

Table 1.10: Example relations and seed pairs used for the evaluation of ESPRESSO

Predicate	**class label**	**relation label**
City	“city”, “town”	——
Country	“country”, “nation”	——
capitalOf(City,Country)	——	“capital of”

Table 1.11: Example input for the CAPITAL-OF relation in KNOWITALL

KnowItAll

At one extreme on the spectrum of minimal supervision is KNOWITALL [48], a minimally-supervised system for relation extraction that uses a bootstrapping algorithm as part of its architecture. While it uses bootstrapping in a similar way to DIPRE and SNOWBALL, unlike the previously discussed systems KNOWITALL does not require a previously specified seed set of entity pairs. Instead, KNOWITALL requires as input only the class labels and relation labels, which are then used to create initial extraction patterns. For example, in extracting the LOCATED-IN relation between *City* and *Country*, KNOWITALL uses the input class labels and relation labels given in Table 1.11. Here multiple synonyms may be used as class labels.

From the input class labels and relation labels corresponding to the initial predicates, KNOWITALL formulates an initial set of unary patterns for discovering instances of the classes involved in each unary predicate. These patterns are deterministically formulated from a set of manually-specified patterns for discovering the IS-A relation. The patterns used are a superset of the Hearst patterns [60], listed below:

- <entity> “and other” <class>
- <entity> “or other” <class>

- <class> "especially" <entity>
- <class> "including" <entity>
- <class> "such as" <entity>
- "such" <entity> "as" <entity>
- <entity> "is a" <class>
- <entity> "is the" <class>

Having extracted entities belonging to each of the classes in a given relation, examples of pairs of related entities are extracted. These pairs are extracted using two additional manually-specified pattern templates, each with three slots (one for each of the two classes, along with the relation label). These relation extraction patterns are listed below:

- <class1> "is the" <relation> <class 2>
- <class1>, <relation> <class 2>

For example, having extracted *Paris* and *Berlin* as cities, and *France* and *Germany* as countries using the unary IS-A patterns for the City and Country predicates, the binary relation patterns could be filled in as "Paris is the capital of France", "Berlin, capital of Germany", etc. KnowItAll uses an extension of the PMI-IR algorithm from Turney [146] to evaluate the extracted entities and novel patterns. KnowItAll also uses a variety of other sources of evidence beyond extracted textual patterns, including subclass extraction and list extraction.

The semi-supervised approach presented in the previous algorithms offers a method of learning new patterns and entities from only a small set of seed entities. The heuristic weighting used allows the classifiers to avoid overtraining despite having only a very small training set. Nonetheless, several flaws exist with these methods; for one, these algorithms typically have a large set of input parameters that have been tuned by experiment. The dependence of the stability of these algorithms on the specific input parameters must be considered; it is possible that in many circumstances the performance of these algorithms is very sensitive to their input parameters.

An appealing alternative to the heuristic weighting of features is the use of a fully supervised classifier with a probabilistic interpretation; this would reduce the reliance on tuning parameters, and simultaneously allow an interpretation that could be used in an iterative framework for re-estimation with theoretical guarantees such as expectation-maximization. Abney [1] gives a theoretical analysis of the general bootstrapping framework proposed by Yarowsky [156], showing that the use of EM within this framework can guarantee a monotonic increase in data likelihood with successive iterations. Further, as suggested by McDonald [90], concerns about overtraining using a supervised method with a small training set can be addressed with regularization or feature selection. In the following section we consider fully supervised algorithms for relation extraction.

1.1.3 Supervised approaches

Many supervised approaches to relation extraction have been widely explored in such venues as the Message Understanding Conferences (MUC) [27], the Automatic Content Extraction (ACE) datasets [45], and various work in biomedical relation extraction [56, 90, 130]. Unfortunately these experiments have typically relied on relatively small datasets. For example, the NIST Automatic Content Extraction (ACE) RDC 2003 and 2004 corpora include more than 1,000 documents in which pairs of entities have been labeled with up to 7 major relation types and up to 24 subrelations, totaling 16,771 positive relation instances. Approaches based on the ACE program have been restricted in their evaluation to a small number of relation instances and corpora of less than a million words.

Many approaches to ACE have explored a wide variety of features for extraction in combination with support vector machine (SVM) or logistic regression classifiers [67, 162, 70]. These studies have compared the use of many features extracted from string, constituency parse, and dependency parse representations of the text. Considered features have included entity attributes, bag-of-words features, bigrams, grammar productions, dependency relations, dependency paths, and many others.

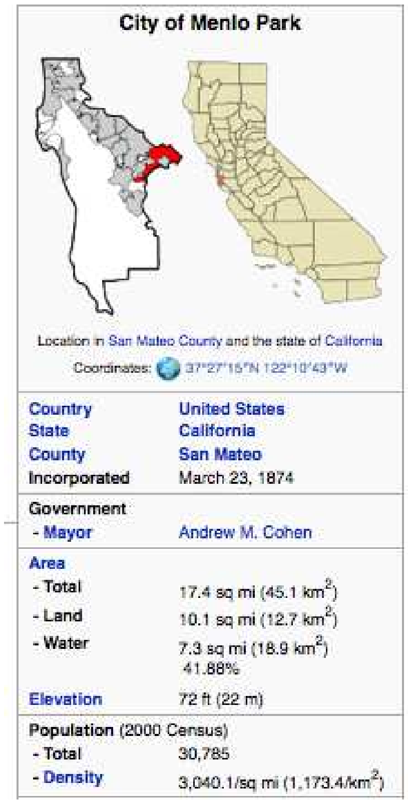
Further lines of research has explored the use of *kernel methods* [35, 148, 37] for exploring a much larger space of features. Zelenko et al. [160] propose a kernel-based approach for relation extraction applied to the output of a shallow parser, demonstrating improved performance over a simpler feature-based approach. Several other kernels have been proposed over a variety of underlying feature spaces for relation extraction, including a subsequence kernel [21], a shortest path dependency kernel [20], a dependency tree kernel [40], a convolution dependency path kernel [151], and the composition of a convolution tree kernel and an entity kernel [161].

Kylin

Kylin [153] is a supervised classifier that has been applied to Wikipedia *infobox* relations. In Wikipedia, a large amount of semi-structured information exists in the form of *infoboxes*, which list attribute-value pairs for specific named classes. An example of an infobox for the city *Menlo Park* is presented in Figure 1.1. This infobox lists such city-specific relations as located-in-country, located-in-state, located-in-county, has-population, and others.

Kylin trains individual classifiers for predicting each class-attribute pair, as well as individual classifiers for predicting members of each class. Thus, if there are C classes and an average of A attributes per class, Kylin trains $C+CA$ total individual classifiers.

Consider a particular classifier with the objective of extracting a specific class/attribute pair: for each such classifier Kylin automatically creates a labeled training set from the documents labeled by the corresponding infoboxes with the target class and attribute. Positive training examples in this set are created by extracting the sentences that contain the infobox-labeled attribute value for the specified class, while negative examples are created by extracting other sentences not containing the infobox-labeled term. For example, consider the case of training a classifier to predict the located-in-state relation for cities. The infobox for *Menlo Park* states that *Menlo Park* is located in *California*; further, the first sentence in the Wikipedia article states that “Menlo Park is an affluent city in...California”. This sentence would



City of Menlo Park

Location in San Mateo County and the state of California

Coordinates: 37°27′15″N 122°10′43″W

Country	United States
State	California
County	San Mateo
Incorporated	March 23, 1874
Government	
- Mayor	Andrew M. Cohen
Area	
- Total	17.4 sq mi (45.1 km^2)
- Land	10.1 sq mi (12.7 km^2)
- Water	7.3 sq mi (18.9 km^2) 41.88%
Elevation	72 ft (22 m)
Population (2000 Census)	
- Total	30,785
- Density	3,040.1/sq mi (1,173.4/km^2)

Figure 1.1: Example Wikipedia infobox for the city of Menlo Park

be added as a positive example to the training set for predicting LOCATED-IN-STATE, whereas the following sentence ("It is located at latitude...") would be added as a negative example to the training set.

This model differs significantly from previously discussed approaches in this section, in that it does not require that both the class and the attribute appear in the same sentence. Instead, KYLIN requires that the document from which an attribute is extracted has been determined to be relevant to the specified class in some way. Since the relation classifier is now only attempting to classify a single contiguous entity within a sentence (rather than entity pairs, as in previous algorithms), the task of entity extraction can be framed as a sequence labeling task, akin to many approaches in named entity recognition. This allows the classifier to consider a wide range of features extracted from the sentences in the training set, including not only the preceding and following tokens, but also a variety of string features over the extracted target token (including capitalization, numeric characters, etc.), whether the target is anchored in a hypertext link, and many others. These extracted features are used to train a conditional random field (CRF) classifier [75].

To extract novel relations, the trained classifiers are applied to a set of documents predicted to be relevant to the specific target relation by an independent document classifier. In a manual evaluation of the predictions of the classifiers on attributes for four major classes (Actor, Airline, U.S. County, and University), the performance of the KYLIN classifiers was found to be comparable to the performance of human editors.

While these classifiers demonstrate high performance for certain relations, this form of fully-supervised relation extraction still suffers from a number of problems. Labeled training data is expensive to produce and thus limited in quantity. Further, because the training data is typically drawn from a small corpus, the resulting classifiers are likely to be biased toward the particular domain represented by the training corpus. The problem of data sparsity in fully-supervised approaches has been addressed in some work by modeling the commonality among related classes [163]; nonetheless, the fully-supervised approach remains fundamentally constrained by the requirement for labeled data.

1.1.4 Unsupervised approaches

An alternative to the supervised or semi-supervised approaches is purely unsupervised information extraction, where the "relations" are discovered via extracting, clustering, and simplifying the words connecting entities in large amounts of text [136, 8, 9]. Here we consider one such approach, TextRunner, in detail.

TextRunner

Whereas KnowItAll [48] is a minimally-supervised relation learner, requiring as a minimum only a single pattern for each class name and relation, TextRunner [8, 9] uses no class-specific or relation-specific supervision whatsoever. Instead, TextRunner attempts to identify tuples of the form $(e_i, r_{i,j}, e_j)$ within each individual sentence. In order to identify entities and relations, the system applies a supervised statistical parser [73] to a small corpus, and applies a set of heuristics to the resulting output, such as:

- There exists a dependency path between e_i and e_j consisting of no more than a threshold t individual dependency links.
- The path from e_i to e_j in the syntax tree does not cross a sentence-like boundary.
- Neither e_i nor e_j consists solely of a pronoun.

If all of the set of heuristics are satisfied, the extracted tuples and sentence are labeled as positive examples; if one or more of these heuristics are unsatisfied, the extracted tuples and sentence are labeled as negative examples. A supervised classifier is then trained using this parser-labeled corpus, with the goal of extracting the positive-labeled $(e_i, r_{i,j}, e_j)$ tuples from each sentence, using domain-independent features extracted without the aid of a syntactic parser. Various supervised classifiers have been applied to the extraction of the these tuples, including Naive Bayes [8] and Conditional Random Fields [9].

TextRunner applies this light-weight classifier for tuple extraction to a large web corpus. Potential relation synonyms are found and resolved using the Resolver algorithm [157, 158], an unsupervised method for determining if two strings refer to the same item. Applied to a corpus of 9 million Web pages, TextRunner extracts more than 7 million tuples with well-formed arguments, with an estimated 1 million of those tuples grounded in concrete, real-world entities, and with an estimated precision of over 80%.

Further work in this direction has resulted in the AuContraire system [127] for detecting contradictions within relations inferred in this unsupervised manner. These experiments include a study of detecting whether certain automatically-inferred relations are *functional*, i.e., whether there is only one or if there are many possible second arguments for a relation given a particular first argument. Further, the relations extracted by TextRunner have been used by HypernymFinder [128], a system for hypernym discovery using the distant supervision of a set of proposed hyponym/hypernym pairs automatically extracted using Hearst patterns.

The unsupervised approach has the benefit of being able to use very large amounts of data to extract very large numbers of relations; however, since the resulting relations have not been standardized by any particular predicate, it may not be easy to map them to the relations needed for a specific knowledge base or application.

1.1.5 Discussion

Several learning paradigms have been applied to the task of extracting relational facts from text. Here we have discussed four basic paradigms: using manually-specified patterns, and semi-supervised, supervised, and unsupervised classification. Each approach has corresponding advantages and disadvantages. Manual patterns are easy to specify, but frequently result in poor precision. Semi-supervised classification only requires a small number of examples and can be applied to an arbitrarily large corpus, but can be subject to semantic drift and may require a large number of experimentally-tuned initial parameters. Fully-supervised classifiers can offer high performance on certain relations, but typically requires a large amount of expensive annotation for each desired relation. Finally, unsupervised approaches can extract a very large number of relations from arbitrarily large corpora, but the extracted relations may be difficult to resolve with existing predicates in taxonomies or applications.

In this work we discuss an alternative approach to training relation extraction classifiers, *distant supervision*, that combines some of the advantages of each of these approaches. Distant supervision involves using an existing semantic database to automatically label an unannotated corpus for relation extraction. The intuition of distant supervision is that any sentence that contains a pair of entities that participate in a known relation in an existing source of structured knowledge is likely to contain some evidence for that particular relation. Since this technique does not require natural text to be specially labeled in any way, distant supervision may be applied to an arbitrarily large corpus, with the goal of extracting a very large number of (potentially noisy) features that can be used in a supervised classifier. Thus whereas the supervised training paradigm is typically restricted to only a small labeled corpus of relation instances as training data, our algorithm can make use of much larger amounts of text. Further, because our algorithm is supervised by a database, rather than by labeled text, it does not suffer from the problems of overfitting and domain-dependence that plague supervised systems.

1.2 Taxonomy induction

While many relation extraction algorithms have been proposed, they frequently lack the global knowledge necessary to integrate their predictions into a complex taxonomy with multiple relations. Several challenges arise when moving from simple relation extraction to the problem of creating a self-consistent structured representation of relations between concepts, including the problems of combining evidence across multiple predictions and resolving issues of lexical ambiguity.

Past work on semantic taxonomy induction includes the noun hypernym hierarchy created by Caraballo [23], the part-whole taxonomies by Girju et al. [55], and a great deal of related work described by Buitelaar et al. [18]. Such work has typically either focused on only inferring small taxonomies over a single relation, or as in [23], has used evidence for multiple relations independently from one another, by for example first focusing strictly on inferring clusters of coordinate terms, and then inferring hypernyms over those clusters.

Another major shortfall in previous techniques for taxonomy induction has been the inability to handle lexical ambiguity. Previous approaches have typically sidestepped the issue of polysemy altogether by making the assumption of only a single sense per word, and inferring taxonomies explicitly over words and not senses. Enforcing a false monosemy has the downside of allowing potentially erroneous inferences; for example, collapsing the polysemous term *Bush* into a single sense might lead one to infer by transitivity that a *rose bush* is a kind of *U.S. president*.

Our approach provides a solution to these problems. The key contribution of this work is to offer a solution to two crucial problems in taxonomy induction and hyponym acquisition: the problem of combining heterogenous sources of evidence in a flexible way, and the problem of correctly identifying the appropriate word sense of each new word added to the taxonomy.

1.3 Overview of experimental results

In addition to presenting our approach to *distant supervision* for relation extraction and our probabilistic framework for taxonomy induction, we present a number of experimental results showing the use of these techniques to address several previously unsolved problems. Our experiments demonstrate that a distantly supervised classification algorithm can outperform previously proposed automatic methods for identifying hypernym pairs in newswire corpora, and even has better performance than using WordNet itself. Using our probabilistic framework for taxonomy induction we have constructed multiple augmented taxonomies using WordNet as a base, the largest of which are more than four times the size of the existing WordNet noun taxonomy. We have shown that the augmented taxonomies constructed in this manner may be used directly as substitutes for WordNet in many natural language applications, and have resulted in improvements in the performance of such applied tasks as question answering and text classification. We also describe a method for measuring the similarity between two concepts within a taxonomy like WordNet, and show that our method has better performance than many previously proposed automatic techniques as evaluated on a manually-created gold standard testset. We then demonstrate the application of this method for determining the similarity of two concepts to the creation of arbitrarily coarse-grained taxonomies. Next we demonstrate the application of these sense-clustered taxonomies to the problem of coarse-grained evaluation of word sense disambiguation algorithms. The augmented and sense-clustered taxonomies discussed in this thesis have been made publicly available, and may be downloaded at the following link:

`http://ai.stanford.edu/∼rion/swn`

1.4 Major contributions of each chapter

Here the contributions of this dissertation are summarized by chapter:

- **Chapter 2**

  In Chapter 2 we discuss the problem of relation extraction and our approach of using *distant supervision* for creating classifiers to predict taxonomic relations. We first discuss the structure and relations within WordNet, the primary taxonomy that we consider throughout this thesis. We then describe the general-purpose framework of distant supervision for training classifiers to recognize relations between pairs of entities in a large unannotated corpus. We describe specific experiments using this approach to learn the hypernym (IS-A) relation from lexico-syntactic patterns, demonstrating how we can use examples of known hypernym pairs from WordNet to automatically identify large numbers of useful lexico-syntactic patterns, and then combine these patterns using a supervised learning algorithm to obtain a high accuracy hypernym classifier. We further describe experiments to learn hypernym and coordinate term relations from other sources of information, including distributional similarity and semi-structured list information.

- **Chapter 3**

In Chapter 3 we discuss the problem of formulating a unified framework for joint inference over predictions from multiple relation classification algorithms. We discuss a framework that is capable of using multiple predictions in order to both consistently improve relation prediction accuracy and to robustly disambiguate the senses of the predicted entities with respect to an existing sense hierarchy. We discuss related work in addressing the challenges of joint inference and lexical ambiguity. We outline our basic definition of taxonomies, relations, and concepts, and give a probabilistic means for calculating taxonomy likelihood. Next, we show how our model is capable of being extended to calculate evidence among word senses rather than simply word tokens. Finally, we discuss the primary basic operations we use to modify taxonomies, along with each operation's respective effect on the estimate of taxonomy likelihood.

- **Chapter 4**

  In Chapter 4 we discuss the application of our model for taxonomy induction to the problem of automatically augmenting the WordNet taxonomy. We discuss a set of experiments with the goal of inferring novel noun hyponyms within WordNet, using a combination of hypernym and coordinate term classifiers. We discuss the construction and evaluation of several augmented taxonomies created by this approach. We compare the performance of our model for taxonomy induction against a baseline algorithm using the same underlying relation classifiers. We also compare our performance against several "oracle" named entity recognition baselines, showing that our method has significantly higher fine-grained link precision, coarse-grained link precision, and sense disambiguation precision. We also show significant improvement over WordNet on the task of predicting hypernym relations in newswire text.

- **Chapter 5**

  In Chapter 5 we consider the problem of creating arbitrarily coarse-grained sense hierarchies by use of an operator for merging concepts within a taxonomy. We present an algorithm for clustering large-scale sense hierarchies like

WordNet that makes use of a supervised classifier capable of making graduated judgments corresponding to the estimated probability that each particular sense pair should be merged. This classifier is trained on gold standard sense clustering judgments using a diverse feature space. We are able to use the outputs of our classifier to produce a list of sense merge judgments ranked by merge probability, and from this create sense-clustered inventories of arbitrary sense granularity. We evaluate our classifier for merging senses against a gold-standard human-labeled dataset of merged senses, and we evaluate our sense-clustered hierarchies with respect to the task of coarse-grained word sense disambiguation.

- **Chapter 6**

  In Chapter 6 we discuss the topic of applying our techniques and automatically inferred taxonomic relations to applied tasks in natural language processing. We discuss the applications of our model to the problems of domain-specific taxonomy induction, foreign-language taxonomy augmentation, question answering, and text categorization.

- **Chapter 7**

  In Chapter 7 we summarize and discuss the contributions and results of this thesis.

1.5 First published appearances of contributions

Most of the contributions described here have first appeared in other publications. Here we cite the publications where some portion of the contributions in each chapter have appeared:

- Chapter 2: Snow, Jurafsky, and Ng (2005) [137] and Mintz, Bills, Snow, and Jurafsky (2009) [96].

- Chapter 3: Snow, Jurafsky, and Ng (2006) [138].

- Chapter 4: Snow, Jurafsky, and Ng (2006) [138].
- Chapter 5: Snow, Prakash, Jurafsky, and Ng (2007) [139].
- Chapter 6: Srinivasan, Snow, Branson, Pande, Ng, Altman, and Batzoglou (2007) [142] and McNamee, Snow, Mayfield, and Schone (2008) [91].

Chapter 2

Predicting Taxonomic Relations

2.1 Introduction

In this chapter we consider the application of *distant supervision* to the creation of a classifier for predicting whether a factual relation exists between a pair of entities. In particular we will focus on the *hypernym* relation. We state that a noun X is a hyponym of a noun Y (and, respectively, Y is a *hypernym* of X) if X is a subtype or instance of Y, that is, if X is a *kind of* Y. Thus "Shakespeare" is a hyponym of "author" (and conversely "author" is a hypernym of "Shakespeare"), "dog" is a hyponym of "canine", "desk" is a hyponym of "furniture", and so on. The hypernym relation is considered to be transitive—thus if X *is-a* Y and Y IS-A Z, then we can infer that X IS-A Z. Section **??** gives a detailed discussion of the hypernym relation in the context of WordNet.

As discussed in Section 1.1, many supervised and bootstrapping approaches to relation extraction have been previously proposed, but have typically relied on relatively small datasets or initial seed sets. In this work we discuss an alternative approach to training relation extraction classifiers, *distant supervision*, that combines some of the advantages of each of the previously proposed approaches. Distant supervision involves using an existing semantic database to automatically label an unannotated corpus for relation extraction. The intuition of distant supervision is that any sentence that contains a pair of entities that participate in a known relation in an existing source of structured knowledge is likely to contain some evidence for that particular relation. Since this technique does not require natural text to be specially labeled in any way, distant supervision may be applied to an arbitrarily large corpus, with the goal of extracting a very large number of (potentially noisy) features that can be used in a supervised classifier. Thus whereas the supervised training paradigm is typically restricted to only a small labeled corpus of relation mentions as training data, our algorithm can make use of much larger amounts of text. Further, because our algorithm is supervised by a database, rather than by labeled text, it does not suffer from the problems of overfitting and domain-dependence that plague supervised systems.

Supervision by a database also means that, unlike unsupervised approaches, our classifier uses canonical names for the relations it predicts. In addition, combining vast numbers of features in a large classifier helps reduce the problem of noisy patterns. This approach offers a natural way of integrating data from multiple sentences to decide if a relation holds between two entities. Because our algorithm can use large amounts of unlabeled data, the same pair of entities may occur many times in the test set. For each pair of entities, we aggregate the features from the many different sentences in which that pair appeared and treat these as a single instance, allowing us to provide our classifier with more information, resulting in more accurate labels.

A form of this training paradigm has been used effectively by Wu and Weld [153] in order to extract relations from Wikipedia articles using relations in Wikipedia infoboxes for supervision. Unlike the corpus-specific method proposed by Wu and Weld, which typically only infers features from the Wikipedia page containing the target infobox, our method allows us to extract evidence for a particular relation across many different articles. Similar algorithms have also been proposed in the analysis of biomedical text, for example the use of *weakly labeled data* [36, 98].

This chapter is structured as follows: in Section 2.2 we discuss WordNet, the structured knowledge base that we primarily focus on for relation extraction; in Section 2.3 we introduce the approach of distant supervision for training relation classifiers, and we describe our representation for automatically discovering patterns indicative of hypernymy. Section 2.4 describes the setup of our experiments, Section 2.5 provides an analysis of our feature space, and in Section 2.6 we discuss a classifier using these features that achieves high accuracy on the task of hypernym identification. In Section 2.7 we show how we can use lexico-syntactic patterns and distributional similarity in order to classify coordinate terms, and in Section 2.8 we demonstrate one way in which we can combine our hypernym and coordinate classifiers to improve hypernym classification, and further discuss the results of comparing our hypernym classifier to WordNet in the applied task of discovering hypernym pairs in newswire text Section 2.9 considers semi-structured lists an additional source of knowledge for inferring hypernym and coordinate relations. Finally, in Section 2.10 we discuss and summarize the results of this chapter.

2.2 WordNet

In order to examine the kinds of relations we would like to learn to extract, we first consider WordNet, the structured knowledge representation that the majority of our work has focused on. WordNet began as an exploration of several theories in psycholinguistics by one of the fathers of cognitive psychology, George Miller. The first electronic version started as a semantic network of 45 nouns in 1984. This semantic network differed from standard dictionaries in that it contained structured representations of the semantic relations between concepts in the network. In particular, each node in the network was connected to at least one other node by use of the IS-A, or *hypernym* relation, thus forming a graph of concepts.

This initial small semantic network was greatly enlarged in the following years and was first publicly released as a network of nearly 40,000 concepts in 1989. It has since grown to more than 100,000 words and concepts in English; the growth of WordNet from its first public release in 1989 until the most recent release in 2006 is shown in Figure 2.1, with data drawn from [50]. One primary motivation behind creating WordNet was to explore the possibility of scaling up theories of relational lexical semantics beyond simple toy problems; as Miller wrote in the Foreword to [50],

> "[M]uch otherwise excellent work along these lines runs aground on the magnitude of the problem. An author might propose a semantic theory

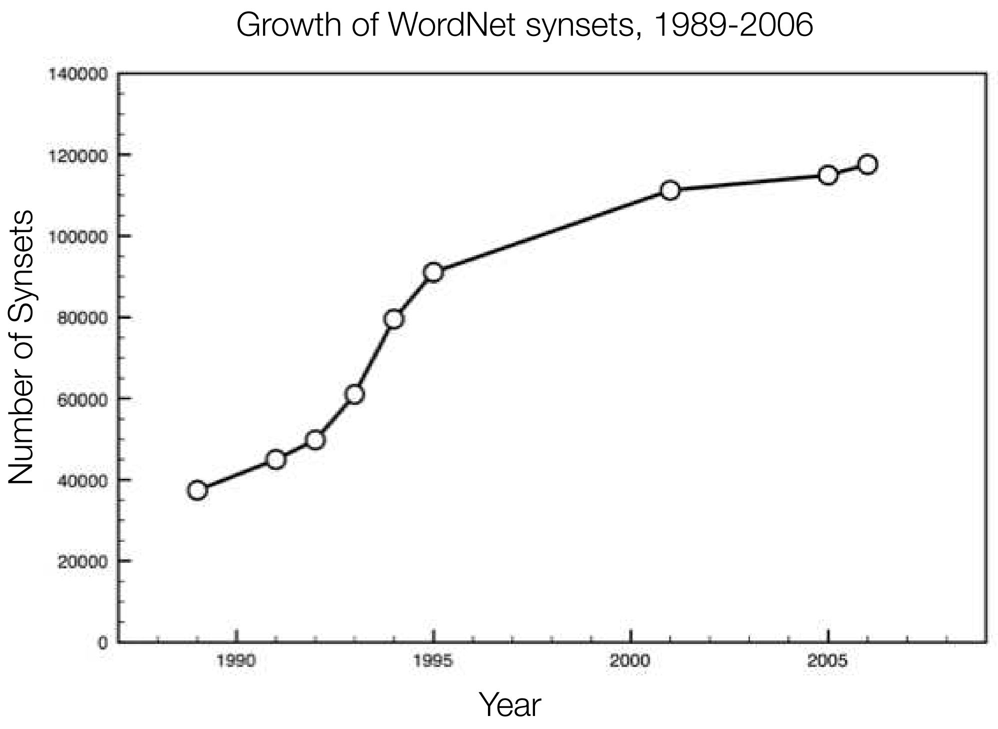


Figure 2.1: Growth of synsets in WordNet, 1989 to 2006 [50]

> and illustrate it with some 20 or 50 English words (usually nouns), leaving the other 100,000 words of English as an exercise for the reader."

However, Miller also recognized the potential importance of this resource for applications in natural language processing. He wrote:

> "[C]omputational linguistics, if it were ever to process natural languages as people do, would need to have available a store of lexical knowledge as extensive as people have... There seemed to be a need for a comprehensive lexical database that would include word meanings as well as word forms and that could be used under computer control."

As outlined in [50], WordNet has been used for purposes far beyond its initial use as an electronic dictionary browser; it has been used as the primary sense inventory for

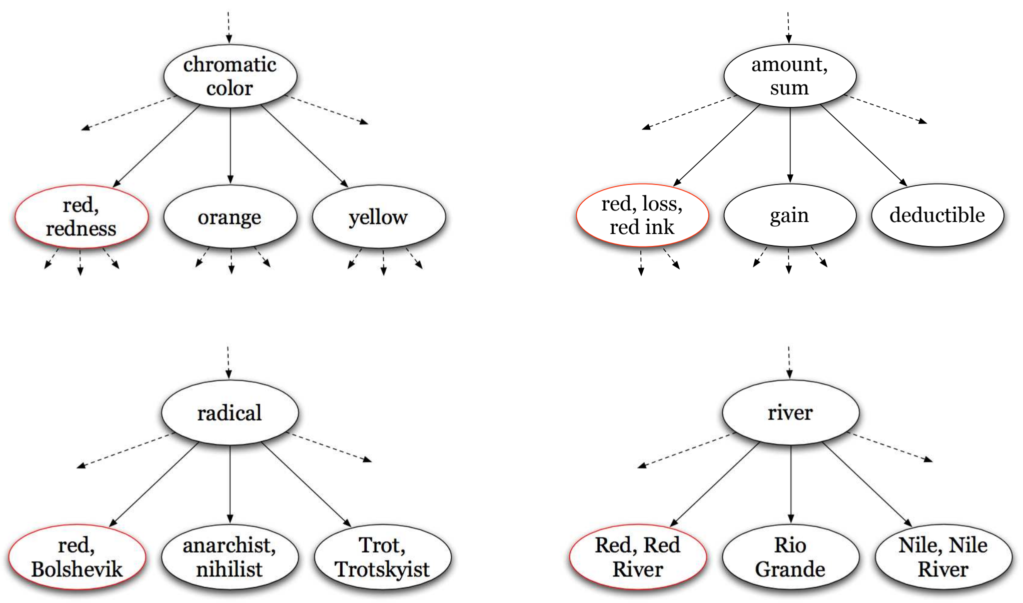


Figure 2.2: Four senses of *red* in WordNet

several word sense disambiguation evaluations [72, 140, 121, 104], and in applications to information retrieval [57], query expansion [97], and question answering [91].

A fundamental feature within the WordNet hierarchy is that of *lexical ambiguity*; that is, there is a many-to-many mapping between words and concepts. Thus an individual word can refer to several possible concepts (or *senses*, or *synsets*), and an individual concept can be described by many different words (or *synonyms*). An example of how this ambiguity is represented within WordNet is shown in Figure 2.2. Here, four different senses are given for the word *red*; the four senses listed within WordNet include the familiar notion of a *color*, but also include *red* as an amount or debt (as in, "in the *red*"), as a slang term for a person with radical ideas, and as the name of a river. Thus there are four different concepts that the word *red* maps to; further, each individual concept in WordNet can have several synonyms that refer to it. For example, the concept of *red* as a slang term for a radical has a total of five synonyms that it maps to, including *Bolshevik*, *Marxist*, *bolshie*, and *bolshy*.

Figure 2.3 gives an overview of the structure of the noun hypernym hierarchy in WordNet. Here direct hypernym relations are indicated with a solid line, and

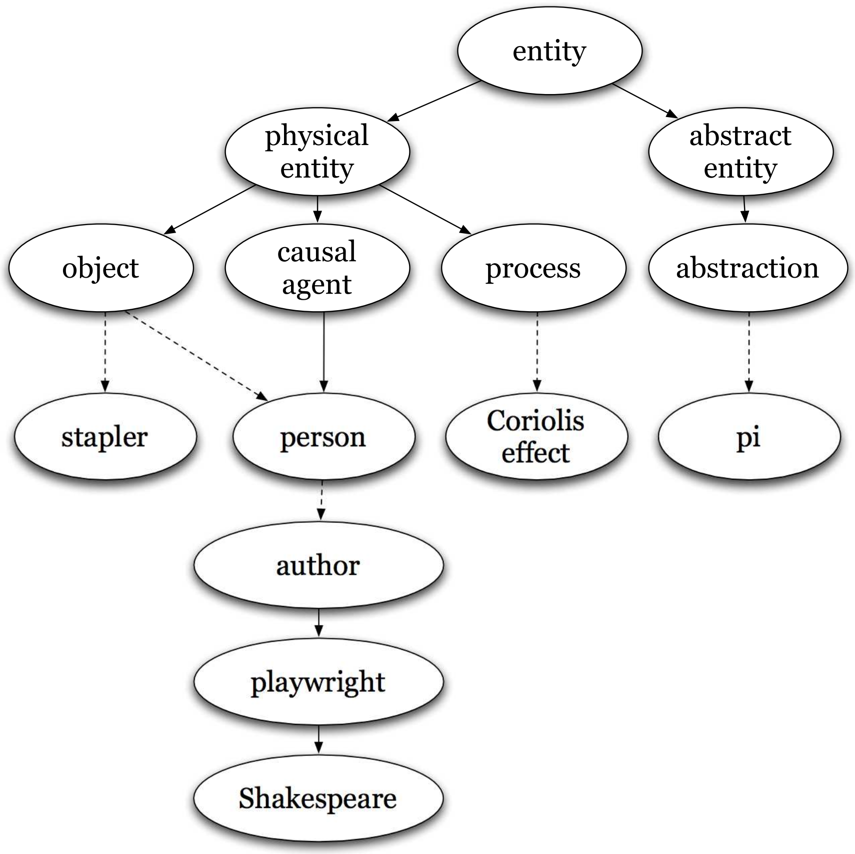


Figure 2.3: Example fragment of noun hypernym hierarchy in WordNet

dotted lines represent a hypernym chain of two or more nodes. At the root of the hierarchy is the most general concept *entity*, and all other nouns descend from this node[1]. Beneath *entity*, nouns are divided into the categories of *physical entity* and *abstract entity*; continuing down the hypernym chain leads to increasingly specific concepts, with terminal nodes such as *stapler*, the *Coriolis effect*, the number *pi*, and the playwright *Shakespeare*.

Proper names (e.g., *Shakespeare*) typically refer to unique entities that have no hyponyms of their own; in many structured representations of knowledge these entities are represented as *instances* and differentiated from the more general category of *classes*. Prior to version 2.1, WordNet made no distinction between noun *classes* and *instances*; both the *is-a-hyponym* and *is-an-instance* relations were subsumed under the single *is-a* relation.

In version 2.1 WordNet added this distinction, tagging 7,671 noun synsets as *instances* [94]. In discussing the addition, Miller and Hristea [94] described instances as sharing three characteristics: first, they are nouns; second, they are proper nouns; and third, they are unique entities and have no hyponyms, that is, they are leaf nodes in the hypernym hierarchy. In the experiments we present in this work we make no distinction between classes and instances, and we treat the class and instance hypernym relations identically, referring to them both as hypernym relations.

To see the full extent of one hypernym ancestry, consider Figure 2.4. This is a representation of the full hypernym ancestry of the leaf node *burgundy*, which has nine hypernym ancestors. This hypernym information can enable a deeper understanding of textual information; for example, by using this information, a question answering system could answer a question such as: "Which couches are red?", even if in a particular corpus a particular couch is only described as "burgundy".

Similarly, the noun meronym hierarchy may be observed in Figure 2.5. Here, *telomere* is a leaf node in the meronym hierarchy – i.e., there are no listed parts of a *telomere*. However, the *telomere* has several holonyms, or things it is a part of: a *chromosome*, which is a part of a *nucleus*, which is a part of *cell*, which is a part of

[1]Prior to WordNet version 2.1, the noun hypernym graph consisted of several unconnected taxonomies, with several roots; the verb hypernym hierarchy in the current version of WordNet still consists of multiple, unconnected components.

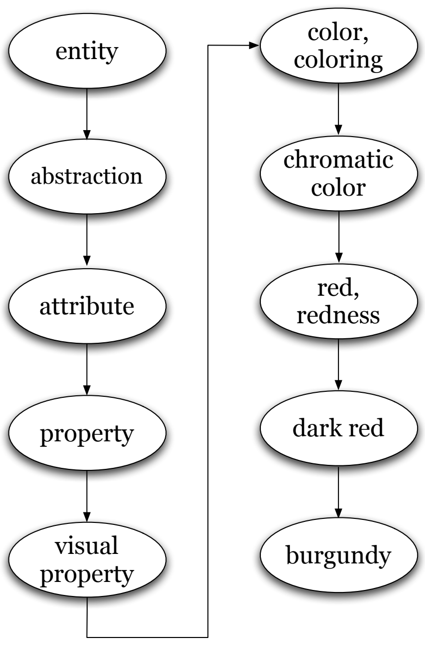


Figure 2.4: Full hypernym ancestry of *burgundy* in WordNet

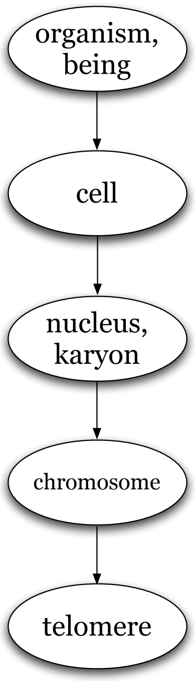


Figure 2.5: Full meronym ancestry of *telomere* in WordNet

an *organism*.

2.3 Distant supervision

Our goal is to apply machine learning to natural language text in order to automatically learn the kind of relational knowledge that has been hand-built in WordNet. In order to accomplish this task without using a labeled corpus we apply the technique of *distant supervision*. This technique allows us to automatically, though noisily, identify sentences within a large unannotated corpus as being positive or negative examples of our target relation. In order to accomplish this we require a knowledge base containing a large number of examples of desired relation; in this chapter we will consider the use of WordNet to automatically label a large unannotated corpus with positive and negative examples of the hypernym relation. In this example, our general approach is to use examples of known hypernym pairs from WordNet to automatically identify large numbers of useful lexico-syntactic patterns, and then combine these patterns using a supervised learning algorithm to obtain a high accuracy hypernym classifier.

More precisely, our approach is as follows:

1. Training:

    (a) Collect noun pairs from corpora, identifying examples of hypernym pairs (pairs of nouns in a hypernym/hyponym relation) using WordNet.

    (b) For each noun pair, collect sentences in which both nouns occur.

    (c) Parse the sentences, and automatically extract patterns from the parse tree.

    (d) Train a hypernym classifier based on these features.

2. Test:

    (a) Given a pair of nouns in the test set, extract features and use the classifier to determine if the noun pair is in the hypernym/hyponym relation or not.

The first goal of our work is to automatically identify lexico-syntactic patterns indicative of hypernymy. In order to do so, we need a general way of representing these patterns. We propose the use of *dependency paths* as a general-purpose formalization of the space of lexico-syntactic patterns. Dependency paths have been used successfully to represent lexico-syntactic relations suitable for semantic processing [83, 20, 40, 151].

2.3.1 Dependency parsing

Dependency parsing is a method for creating a representation of the syntactic structure of a sentence. The origins of dependency parsing may be found in the tradition of dependency grammar in theoretical linguistics, which theorizes that the syntactic structure of a sentence may be captured entirely by a set of dependency relations between individual words [100]. For example, consider the tree formed by the syntactic relations indicated in Figure 2.6. The development of automatic methods for dependency parsing has been an active area of natural language processing research [47, 25, 101, 132].

A dependency parser produces a dependency tree that represents the directed syntactic dependencies between the words or phrases within a sentence. The constituency parse representation of a sentence typically forms a sentence with the sentence symbol S or other start symbol as the root, syntactic constituent symbols as the intermediate nodes, and the actual words of the sentence as leaf nodes. In contrast, each node in a dependency parse representations typically corresponds directly to a word or phrase in the sentence. Dependency parse trees can be constructed in a deterministic fashion from existing constituency trees; examples of 'head-finding' rules for creating dependency parse representations may be found in [33] and [43]. Kübler et al. present a detailed review of dependency parsing in [132].

In this work we use MINIPAR, a broad-coverage dependency parser [81]. We consider each dependency parse of a sentence to be a list of edge tuples of the form: ($word_1$,CATEGORY$_1$:RELATION:CATEGORY$_2$, $word_2$). In this formulation each *word* is the stemmed form of the word or multi-word phrase (so that "*authors*" becomes

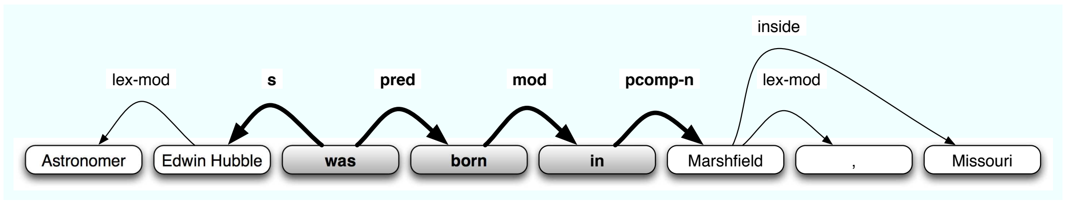


Figure 2.6: Example Minipar dependency tree

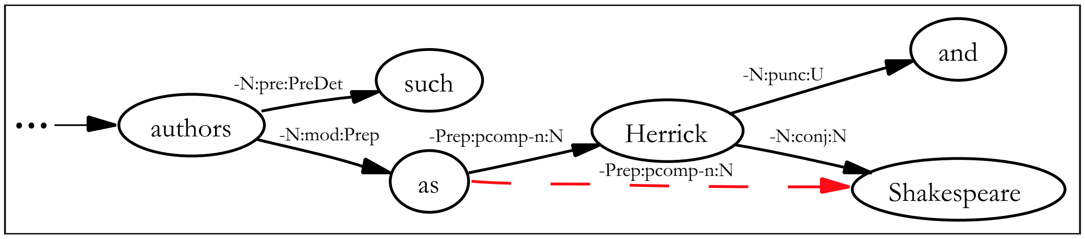


Figure 2.7: Minipar dependency tree example with transform

"*author*"), and corresponds to a specific node in the dependency tree; each *category* is the part of speech label of the corresponding word (e.g., N for noun or Prep for preposition); and the *relation* is the directed syntactic relationship exhibited between $word_1$ and $word_2$ (e.g., obj for object, mod for modifier, or conj for conjunction), and corresponds to a specific link in the tree.

Figure 2.6 shows a representation of the syntactic dependencies in the sentence "Astronomer Edwin Hubble was born in Marshfield, Missouri", generated by Minipar [81]. Note that the root of the tree is the verb "was"; the direct syntactic dependents are the subject "Edwin Hubble" and the predicate "born", and all other nodes in the sentence are further descendants of these nodes.

We define our space of lexico-syntactic patterns to be the space of all shortest paths of four links or less between any two nouns in a dependency tree. Figure 2.7 shows the partial dependency tree for the sentence fragment *"...such authors as Herrick and Shakespeare"* generated by Minipar [81]. Here the extracted shortest dependency path between *authors* and *Shakespeare* in the explicit dependency parse generated by Minipar is: (*authors*, -N:mod:Prep, *as*, -Prep:pcomp-n:N, *Herrick*, -N:conj:N, *Shakespeare*).

2.3.2 Representing syntactic context

Consider the dependency parse of the sentence fragment represented in Figure 2.7. Here the directly extracted dependency path between *authors* and *Shakespeare* is unsatisfying (*authors*, -N:mod:Prep, *as*, -Prep:pcomp-n:N, *Herrick*, -N:conj:N, *Shakespeare*). First, in order to generalize this pattern so that it may be used for the extraction of new entity pairs, we must first remove the lexical nodes *authors* and *Shakespeare* at the terminal points of the path. Second, this shortest dependency path is missing the important *such* node, a dependent of *authors*. Clearly the addition of the *such* node would increase the precision of the extracted pattern. Finally, the explicit shortest dependency path traverses through the *Herrick* node before passing to *Shakespeare*; for the purposes of generalization we would like to bypass the *Herrick* node and distribute the syntactic path across the conjunction relation. In this section we discuss our methods for solving these problems.

First, in order to represent the particular syntactic context between the entities in a sentence, we remove the original nouns in the noun pair to create a more general pattern. Each dependency path may then be presented as an ordered list of dependency tuples; for example, this generalizes the path between *authors* and *Shakespeare* to: (-N:mod:Prep, *as*, -Prep:pcomp-n:N, *Herrick*, -N:conj:N).

We then extend this basic MINIPAR representation in two ways: first, we wish to capture the fact that certain external function words like "such" (in "such NP as NP") or "other" (in "NP and other NP") are important parts of lexico-syntactic patterns. We capture this additional context by adding optional "satellite links" to each shortest path representation. We define a satellite link as any single link not already contained in the shortest dependency path but that is a dependent of either one of the nouns participating in the noun pair. We add redundant feature representations for every satellite dependent within a sentence like this. Thus, for example, we would capture the *such* dependent of *authors* in the dependency path above by augmenting the initial path with the satellite link, yielding the new path ((*such*, PreDet:pre:N), -N:mod:Prep, *as*, -Prep:pcomp-n:N, *Herrick*, -N:conj:N).

Next we address the problem of syntactic paths extended over conjunctions, i.e., over *and*, *or*, and comma-separated lists. In Hearst's original proposed patterns [60],

NP_X and other NP_Y:	(*and*,U:punc:N),-N:conj:N, (*other*,A:mod:N)
NP_X or other NP_Y:	(*or*,U:punc:N),-N:conj:N, (*other*,A:mod:N)
NP_Y such as NP_X:	N:pcomp-n:Prep,*such_as,such_as*,Prep:mod:N
Such NP_Y as NP_X:	N:pcomp-n:Prep,*as,as*,Prep:mod:N,(*such*,PreDet:pre:N)
NP_Y including NP_X:	N:obj:V,*include,include*,V:i:C,*dummy_node,dummy_node*,C:rel:N
NP_Y, especially NP_X:	-N:appo:N,(*especially*,A:appo-mod:N)

Table 2.1: Dependency path representations of Hearst's patterns

comma-separated lists and conjunct-separated lists of noun phrases were explicitly captured in order to collect larger amounts of data. We can capitalize on the distributive nature of the syntactic *conjunction* relation by distributing dependency links across such conjunctions. What this effectively means is that we add additional links to each dependency parse whenever we find a conjunction. For example, in the simple 2-member conjunction chain of "Herrick" and "Shakespeare" in Figure 2.7, we add the entrance link "*as*, -Prep:pcomp-n:N" to the single element "Shakespeare" (as a dotted line in the figure). Distributing dependencies across conjuncts allows us to represent the dependency path between *authors* and *Shakespeare* in the above parse with the following two simple dependency paths: ((*such*, PreDet:pre:N), -N:mod:Prep, *as*) and (-N:mod:Prep, *as*).

This distribution of syntactic dependencies over the conjunction relation has the effect of adding the shortest external dependency path to every member of a conjunct-separated list of noun phrases. Our extended dependency notation is able to capture the power of the hand-engineered patterns described in previous literature; for example, Table 2.1 shows the six patterns used in [23, 26, 60] and their corresponding dependency path formalizations.

2.4 Experimental design

Our goal is to build a classifier which, when given an ordered pair of nouns, makes the binary decision of whether the nouns are related by hypernymy.

All of our experiments are based on a corpus of over 6 million newswire sentences. This corpus contains articles from the Associated Press, Wall Street Journal, and

Los Angeles Times, drawn from the Tipster 1, 2, 3, and Trec 5 corpora [59]. Our experiments also consider articles from Wikipedia (a popular web encyclopedia), extracted with the help of Tero Karvinen's Tero-dump software. We first parsed each of the sentences in the corpus using Minipar. We extract every pair of nouns from each sentence.

We apply the distant supervision of WordNet in order to automatically label the noun pairs in our text corpus as either positive (hypernym) or negative (non-hypernym) training examples. In our experiments we label 752,311 of the resulting noun pairs extracted from our corpus; we label them as either Known Hypernym or Known Non-Hypernym using WordNet.[2] Here we discuss the criteria that we require in order to label a noun pair as either a positive or negative example.

We label a noun pair (n_i, n_j) as a Known Hypernym if one of the nouns n_j is an ancestor of the first sense of the other noun n_i in the WordNet hypernym taxonomy. In addition, we require that the word n_i is the only "frequently-used" expression of that noun out of the possible senses for that word listed in WordNet. We determine a noun sense to be "frequently-used" if it occurs at least once in the sense-tagged Brown Corpus Semantic Concordance files (as reported in the `cntlist` file distributed as part of WordNet 2.0).

Our motivation in determining whether a sense of a noun is "frequently-used" comes from our desire to reduce as much as possible the number of false hypernym/hyponym labels within our automatically constructed training corpus. The problem of lexical ambiguity can greatly reduce the accuracy of an automatic labeling process of this kind; many more potentially false hypernym/hyponym labels come up due to highly polysemous nouns (nouns which have multiple meanings). Note that we consider n_j to be a hypernym of n_i regardless of how many steps away in the noun hierarchy it is with respect to n_i.

A noun pair may be labeled as a negative training example (to the set of Known Non-Hypernym pairs) if both nouns in the noun pair are contained within WordNet, but neither noun is an ancestor of the other in the WordNet hypernym taxonomy for any senses of either noun. Of our total collected noun pairs in our corpus, we find

[2]We access WordNet 2.0 via Jason Rennie's WordNet::QueryData interface.

that 14,387 pairs satisfy our definition for Known Hypernym pairs, and we choose the 737,924 most frequently occurring noun pairs that satisfy our criteria for Known Non-Hypernym pairs. We choose this number of non-hypernym pairs in order to satisfy the roughly 1:50 ratio of hypernym-to-non-hypernym pairs that we observe in our hand-labeled test set (discussed below).

We evaluated our binary classifiers in two ways. For both sets of evaluations, our classifier was given a pair of nouns from an unseen sentence and had to make a hypernym vs. non-hypernym decision. In the first style of evaluation, we compared the performance of our classifiers against the Known Hypernym versus Known Non-Hypernym labels assigned by WordNet. This provides a metric for how well our classifiers do at “recreating” WordNet’s judgments.

For the second set of evaluations we hand-labeled a test set of 5,387 noun pairs from randomly-selected paragraphs within our corpus (with part-of-speech labels assigned by MINIPAR). The annotators were instructed to label each ordered noun pair as one of “hyponym-to-hypernym”, “hypernym-to-hyponym”, “coordinate”, or “unrelated” (the coordinate relation will be addressed in Section 2.7). As expected, the vast majority of pairs (5,122) were found to be unrelated by these measures; the rest were split evenly between hypernym and coordinate pairs (134 and 131, respectively).

Interannotator agreement was obtained between four labelers (all native speakers of English) on a set of 511 noun pairs, and determined for each task according to the averaged F-Score across all pairs of the four labelers. Agreement was 83% and 64% for the hypernym and coordinate term classification tasks, respectively.

2.5 Feature analysis

Our first study focused on discovering which dependency paths might prove useful features for our classifiers. Our representation of the lexico-syntactic space consists of a feature lexicon of 69,592 dependency path. This feature lexicon consists of every extracted dependency path that occurred with at least five unique noun pairs in our corpus. That is, we consider a feature only if it occurs in at least five different feature vectors.

We evaluate these features independently on the task of hypernym classification. For each feature, we constructed a simple binary classifier from only that feature. This classifier predicts that a noun pair is an example of the hypernym relation if and only if the specific feature occurs at least once in the corresponding feature vector.

We may represent these classifiers in mathematical notation in the following way: suppose we represent the n-dimensional entity pair feature vector as $\mathbf{x} = x_0, x_1, x_2, ..., x_n$, where each entry x_i corresponds to the count of the i-th feature. Then, in this step we construct n individual classifiers. Each classifier is a function that considers as input only the corresponding feature; thus a classifier f_i for the i-th feature only depends on the element x_i of the feature vector $\mathbf{x}$. In particular, the classifier will predict a positive instance of the hypernym/hyponym relation if and if the count x_i is greater than zero, that is:

$$f_i(\mathbf{x}) = \begin{cases} 1 & \text{if } x_i > 0 \\ 0 & \text{otherwise} \end{cases}$$

We evaluate each of these classifiers on the WordNet-labeled training set discussed in Section 2.4, consisting of 14,387 positive Known Hypernym pairs, and 737,924 negative Known Non-hypernym Pairs. Figure 2.8 depicts the precision and recall of the simple classifiers we automatically construct in this manner. Precision, represented along the y-axis, gives the fraction of the examples classified as positive examples by the classifier that are actually labeled as true in the testset. Recall, represented along the x-axis, gives the fraction of the total positive instances in the testset that are classified as true by the classifier. That is, assuming that the classifier outputs some classification for every example in the test set, if a given classifier correctly identifies t_p *true positive* examples, and correctly identifies t_n *true negative examples*, but falsely identifies f_p negative instances as positive, and falsely identifies f_n positive examples as negative, then Precision is defined as:

$$\text{Precision} = \frac{t_p}{t_p + f_p}$$

and Recall is defined as:

$$\text{Recall} = \frac{t_p}{t_p + f_n}$$

Finally, we will also refer to the "F-score" as a summary of the performance of a classifier. The F-score is simply the harmonic mean of the Precision and Recall, or:

$$\text{F-score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Thus, each of Precision, Recall and F-score scale between 0 and 1. In a precision-recall plot, the ideal classifier would have a precision and recall of 1, residing at the upper right corner of the plot. In our task the ideal classifier would classify all 14,387 of the actual positive examples as true (thus having a recall of 1), and no more (thus having a precision of 1). In Figure 2.8 we show the precision and recall of all classifiers with recall at least .0015 on the WordNet-labeled data set. Redundant features consisting of an identical base path to an identified pattern but differing only by an additional "satellite link" are marked in Figure 2.8 by smaller versions of the same symbol. For example, the large X glyph in the figure represents the dependency path corresponding to the "X and/or other Y" lexico-syntactic pattern. Additional features also exist as variation of this pattern; for example, the pattern might have a satellite link with a generic article such as "a" or "the" before X, expressing the lexico-syntactic pattern"a X and/or other Y" or "the X and/or other Y". Such patterns along the high precision/recall boundary differing only by a satellite link are shown with the smaller version of the glyph.

Using this formalism we have been able to capture a wide variety of repeatable patterns between hypernym/hyponym noun pairs. in particular, we have been able to rediscover the hand-designed patterns originally proposed by Hearst [60]. These patterns are the first five features listed in Figure 2.8[3].

This analysis serves as a quantitative justification to Hearst's initial intuition of the power of these particular hand-selected patterns; nearly all of Hearst's patterns

[3]There are five basic patterns instead of the six that Hearst proposed because the "and other" and "or other" patterns are collapsed into a single pattern. This pattern can be thought of as a generalized "*conjunction* other" pattern, and is represented within the MINIPAR dependency scheme as -N:CONJ:N, (*other*,A:MOD:N).

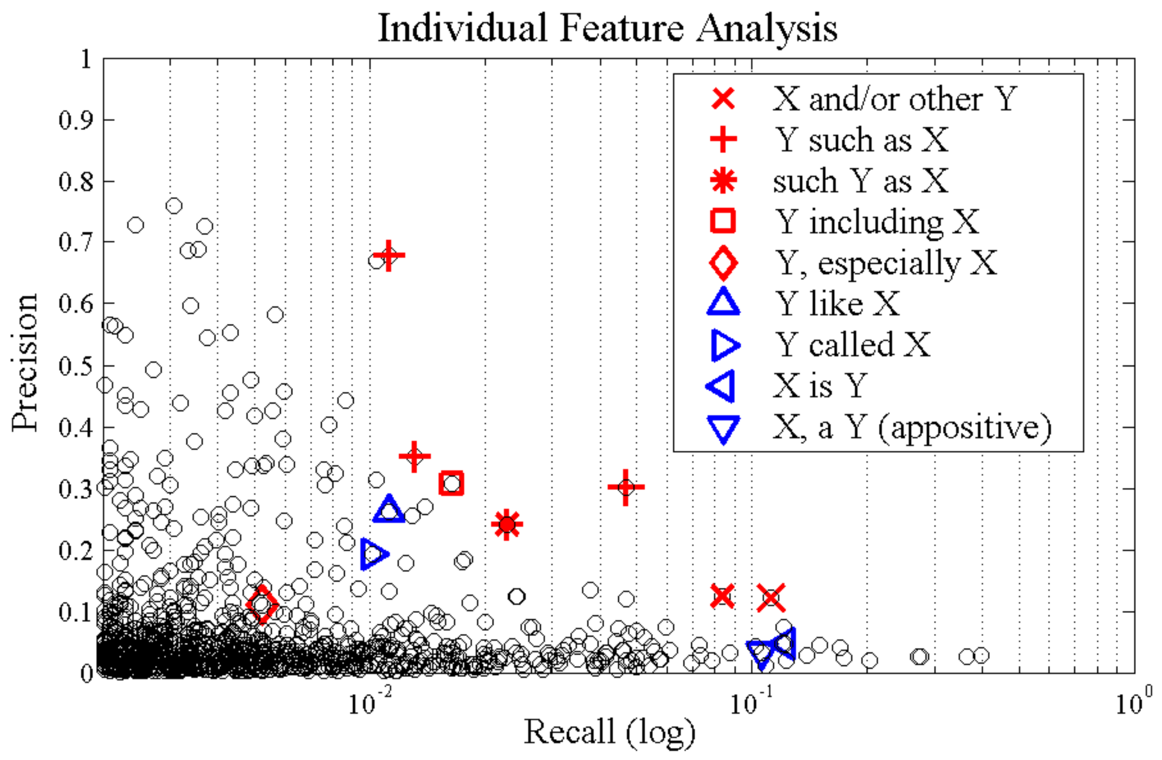


Figure 2.8: Hypernym Figure/Recall for all features

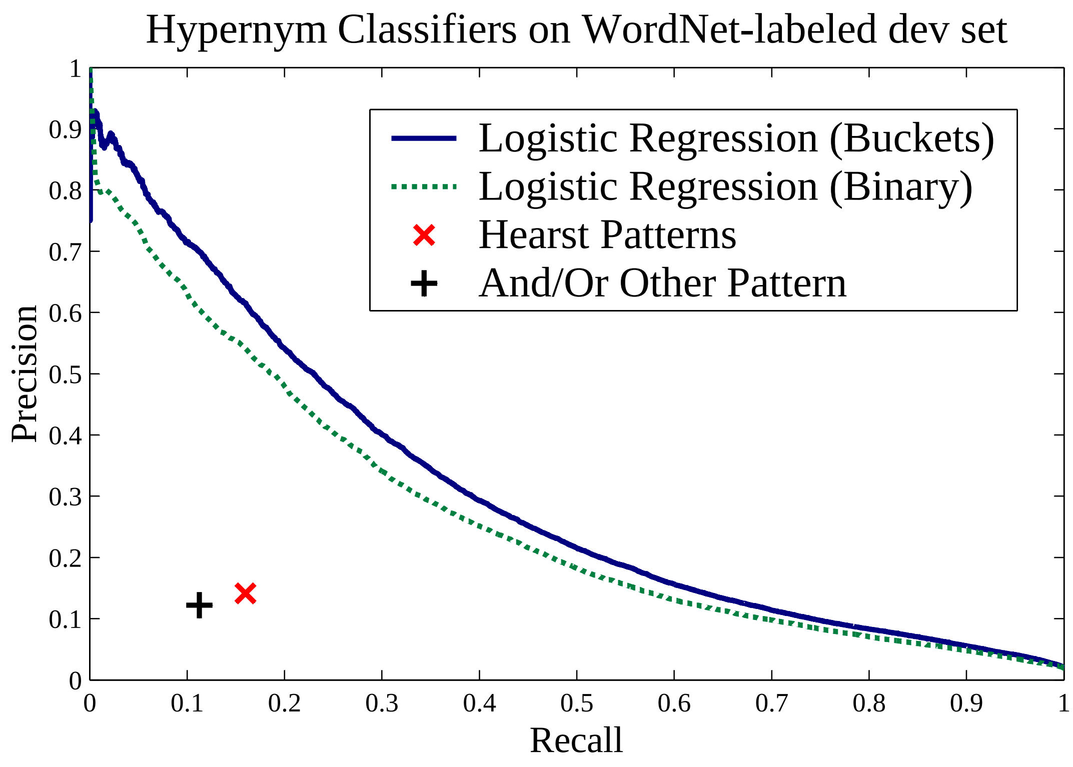


Figure 2.9: Hypernym classifier precision/recall on WordNet-labeled testset

Text representation	**Dependency Path Representation**
NP_Y like NP_X:	N:PCOMP-N:PREP,*like,like*,PREP:MOD:N
NP_Y called NP_X:	N:DESC:V,*call,call*,V:VREL:N
NP_X is a NP_Y:	N:S:VBE,*be,be*,-VBE:PRED:N
NP_X, a NP_Y (appositive):	N:APPO:N

Table 2.2: Dependency path representations of other high-scoring patterns

are at the high-performance boundary of precision and recall for individual features. Nonetheless there are tens of thousands of additional patterns represented here. In addition to Hearst's patterns, there are a large number of new patterns not previously discussed; we mark four as blue triangles in Figure 2.8 and listed in Table 2.2. We can expect that there is a great deal of signal in the wealth of patterns represented here. Some patterns, like "X is a Y", give weak evidence for the hypernym relation; while many hypernym pairs are seen with this pattern, many non-hypernyms are also seen with this pattern. Other patterns, like "Y called X" are more precise. In the next section we consider algorithms for effectively using the full power of this feature space using supervised classification.

2.6 Hypernym classification

Our first hypernym classifier is based on the intuition that a given noun pair is more likely to be in the hypernym relation if that pair occurs in a large corpus of text with one or more of the lexico-syntactic patterns found to be indicative of hypernymy in our labeled training set. We will consider every noun pair that occurs with at least five unique paths from our feature lexicon discussed in the previous section. Then, for each noun pair, we create a feature vector, where each element of the feature vector represents the count of one of the 69,592 features in our lexicon. That is, each element corresponds to a specific dependency path, and the count corresponds to the total number of times that the dependency path was found as the shortest path connecting the corresponding noun pair in some dependency tree in our corpus. In order to perform hypernym classification we define our task as the binary classification of a noun pair as a hypernym pair based on its feature vector of dependency path counts.

For our initial experiments we use the WordNet-labeled Known Hypernym / Known Non-Hypernym training set defined in Section 2.4. We experimented training several supervised classifiers on this data set, including multinomial Naive Bayes, and logistic regression.

2.6.1 Multinomial Naive Bayes

We formulate the task of hypernym prediction as a *binary classification* task. That is, in this task we state that a noun pair may be classified into one of two classes: the hypernym class (i.e., a positive example of the hypernym class), or the non-hypernym class (i.e., a negative example of the hypernym class). Here we consider using the multinomial Naive Bayes model for binary classification. The Multinomial Naive Bayes model is a *generative* model that assumes that each individual occurrence of a feature f_i in a feature vector $\mathbf{x}$ is generated with a fixed conditional probability $P(f \in \mathbf{x} = f_i | \mathbf{x} \in \mathbf{X}_c)$ given the class c of the feature vector $\mathbf{x}$ to which the feature belongs.

Given that we have n initial syntactic dependency path features, in our Naive Bayes model we will have $2n$ model parameters; one for each class-feature pair. Here the likelihood of a feature vector corresponding to a positive hypernym relation is calculated as the product of the individual parameters for each feature that occurs in that feature vector, Each of these parameters θ_{ci} correspond to an estimate of the conditional probability of any particular element within the feature vector $\mathbf{x}$ of class c corresponding to the feature f_i. That is, if we imagine each feature in an instance $\mathbf{x}$ of class c as being generated sequentially, θ_{ci} corresponds to the probability that the next feature encountered will be f_i. If we denote the set of feature vectors corresponding to class c as $\mathbf{X}_c$, and the frequency of feature f_i within a given feature vector $\mathbf{x}$ as x_i, we may then directly calculate each parameter as:

$$\theta_{ci} = \hat{P}(f \in \mathbf{x} = f_i | \mathbf{x} \in \mathbf{X}_c) = \sum_{\mathbf{x} \in \mathbf{X}_c} \frac{x_i}{\sum x_j \in \mathbf{x}}$$

We may also estimate the prior probability $P(c)$ of a feature vector belonging to the class c as the simple fraction of all labeled feature vectors that are labeled as belonging to c:

$$\theta_c = \hat{P}(c) = \frac{\sum_{\mathbf{x} \in \mathbf{X}_c} 1}{\sum_{\mathbf{x} \in \mathbf{X}_c} 1 + \sum_{\mathbf{x} \notin \mathbf{X}_c} 1}$$

Having calculated these parameters, we may estimate the conditional probability of a feature vector $\mathbf{x}$ belonging to the positive class c_1 (as opposed to the negative class c_2) given the set of features as:

$$\hat{P}(c_1|\mathbf{x}) = \frac{\hat{P}(c_1) \prod_{x_i \in \mathbf{x}} \theta_{c_1 i}^{x_i}}{\hat{P}(c_1) \prod_{x_i \in \mathbf{x}} \theta_{c_1 i}^{x_i} + \hat{P}(c_2) \prod_{x_i \in \mathbf{x}} \theta_{c_2 i}^{x_i}}$$

In this way we can calculate the estimated conditional probabilities $\hat{P}(c_1|\mathbf{x})$ and $\hat{P}(c_2|\mathbf{x})$ and assign the label of the class with the larger conditional probability. In practice, since the denominator in the calculation is the same for both $\hat{P}(c_1|\mathbf{x})$ and $\hat{P}(c_1|\mathbf{x})$, we typically only calculate the numerators and assign the label corresponding to the the larger of the two, i.e., we choose the estimated class $\hat{c}$ as:

$$\hat{c} = \text{argmax}_c \hat{P}(c) \prod_{x_i \in \mathbf{x}} \theta_c^{x_i}$$

2.6.2 Logistic regression

Next we consider a logistic regression classifier. Logistic regression is a statistical model that estimates the conditional probability of a class $P(c|\mathbf{x})$ given a feature vector $\mathbf{x}$ as a nonlinear function:

$$h_\theta(\mathbf{x}) = \hat{P}(c|\mathbf{x}) = \frac{1}{1 + e^{-\theta^T \mathbf{x}}}$$

We fit the parameters of the classifier using maximum likelihood. Defining the likelihood of our parameters as:

$$L(\theta) = \prod_{\mathbf{x} \in \mathbf{X}} p(c|\mathbf{x};\theta) \tag{2.1}$$

$$\tag{2.2}$$

We may maximize this likelihood using the stochastic gradient ascent rule. Assuming that the class c takes the value of 1 for a the positive class, and 0 for the negative class, θ we update the parameter θ_j corresponding to the weight for the j-th feature using the $i-th$ feature vector $\mathbf{x}^{(i)}$ using the following update:

$$\theta_j := \theta_j + \alpha(c^{(i)} - h_\theta(\mathbf{x}^{(i)})x_j^{(i)}$$

We train logistic regression classifiers using multiple feature representations, as discussed below.

Feature representations

In our logistic regression experiments we explored multiple representations of our features. First we considered a simple binary representation, where we convert each feature count into a single binary feature. Thus if the original feature vector is $\mathbf{x}$, then each element x'_i of our modified binary feature vector $\mathbf{x}'$ is constructed as:

$$x'_i = \begin{cases} 1 & \text{if } x_i > 0 \\ 0 & \text{otherwise} \end{cases}$$

Second we consider a "bucketed" representation where we convert each integer feature count into a set of binary representations. We expand our initial feature set of 69,592 dependency paths to include multiple binary features per dependency path. These binary features are set to 1 if the feature count is greater than a particular threshold, and are set to zero otherwise. Specifically we use 14 redundant thresholds spaced at the exponentially increasing powers of 2: $\{2^0, 2^1, 2^2, ..., 2^{13}\}$, or $\{1, 2, 4, ...8192\}$. This increases the size of our possible feature set to $69,592 \times 14 =$

$974,288$, or nearly a million binary features. However, the vast majority of features in any particular feature vector are zero, and so we can represent the vectors efficiently in a sparse representation.

Since each element of the original feature vector $\mathbf{x}$ is expanded into k different binary features in the represented 'bucketed' binary feature vector $\mathbf{x}'$, we can write that each original element x_i determines the values of the k elements in the new vector from $x_{k\times i}$ to $x_{k\times i+k-1}$. Specifically, we can write that each element x_i determines the element $x_{k\times i+j}$ of the new vector in the following way, for $0 \leq j < k$:

$$x'_{k\times i+j} = \begin{cases} 1 & \text{if } x_i \geq 2^j \\ 0 & \text{otherwise} \end{cases}$$

We evaluated the performance of each classifier by using 10-fold cross validation on the training set of positive and negative WordNet-labeled noun pairs. We perform 10-fold cross validation by randomizing the order of the training set and setting up 10 experiments, each using a different 10% of the training set as the test set for that experiment. Then, we train a classifier using the remaining 90% of the training set, and measure the performance of that classifier on the test set.

We evaluate each model based on its precision, recall, and maximum F-Score averaged across all folds. The summary of maximum F-scores averaged across all folds is presented in Table 2.3, and the precision/recall plot of our best models is presented in Figure 2.9. Due to the continuous output of our probabilistic classifiers for predicting hypernymy, we can produce a full precision-recall curve for hypernym prediction by varying the prediction threshold for the classifier. That is, each point in the precision-recall plot corresponds to a specific probability threshold T such that the classifier classifies instances as members of the positive class if they have probability greater than or equal T, and otherwise as members of the negative class.

For comparison, we also evaluate two baseline algorithms based on simple manually-specified patterns only a handful of hand-engineered features; the first simply detects the presence of at least one of Hearst's patterns as implemented for hypernym discovery in [26]. The second classifier consists of only the "NP and/or other NP" subset of Hearst's patterns, as used in the automatic construction of a noun-labeled hypernym

Classifier	**F-score**
Best Logistic Regression (Buckets):	0.3480
Best Logistic Regression (Binary):	0.3200
Best Multinomial Naive Bayes:	0.3175
Hearst Patterns:	0.1500
“And/Or Other” Pattern:	0.1170

Table 2.3: Average maximum F-scores for cross validation on WordNet-labeled training set

taxonomy in [23]. In our tests we found that the classifier with the greatest performance was the binary logistic regression model using the “bucketed” representation of binary features.

Our classifier shows a dramatic improvement over the approaches using manually-specified patterns; in particular, applying our best logistic regression classifier trained on newswire corpora, we observe a 132% relative improvement of average maximum F-score over the classifier based on Hearst’s patterns. Our results show that the logistic regression classifier is able to combine a rich selection of features, allowing the classification of hypernym pairs with an F-score of 0.348, far above classifiers based on small numbers of patterns with F-scores of 0.150 and 0.117.

2.7 Coordinate term classification

While our hypernym-only classifier performed better than previous classifiers based on hand-built patterns, there is still much room for improvement. As Cederberg and Widdows [26] point out, one problem with pattern-based hypernym classifiers in general is that within-sentence hypernym pattern information is quite sparse. Patterns are useful only for classifying noun pairs which happen to occur in the same sentence; many hypernym/hyponym pairs may simply not occur within the same sentence in a given corpus. For this reason Cederberg and Widdows [26], following Caraballo [23] suggest relying on a second source of knowledge: “coordinate” or similarity relations between nouns.

The WordNet glossary defines *coordinate terms* as “nouns or verbs that have the same hypernym". Here we treat the coordinate relation as a symmetric relation that exists between two nouns that share at least one common ancestor in the hypernym taxonomy, and are therefore "the same kind of thing" at some level. Many methods exist for inferring that two nouns are coordinate terms; inferring that two nouns are coordinates is a common subtask in automatic thesaurus induction.

We expect that by using coordinate information we will be able to increase the recall of our hypernym classifier. For example, if through coordinate term classification we become confident that two nouns n_i, n_j are coordinate terms, and through hypernym classification we become confident that n_j is a hyponym of n_k, we may then infer with higher probability that n_i is also a a hyponym of n_k, even if we have never encountered the pair (n_i, n_k) within a single sentence, and thus we cannot make a confident prediction directly using the hypernym classifier.

Prior work for identifying coordinate terms includes automatic word sense clustering methods based on *distributional similarity* [105, 113] or on pattern-based techniques, specifically using the coordination pattern "X, Y, and Z" [26]. We consider both kinds of classifier.

2.7.1 Vector space model for distributional similarity

First we construct a vector-space model similar to that constructed by Pantel [105], using single MINIPAR dependency links as our distributional features. For every noun in our corpus we construct a feature vector, where the elements of the feature vector correspond to ordered dependency links observed to occur with that noun. For example, in the dependency parse for the sentence "Astronomer Edwin Hubble was born in Marshfield, Missouri", as represented in Figure 2.6, we would add the two immediately adjacent dependency links to the feature vector representation, namely: (-N:lex-mod:N, *Astronomer*) and (N:s:V, *be*).

We construct a feature vector in this way for every noun we observe in the six million sentences in our hypernym training set. In this case our feature lexicon consists of the 30,000 most frequent noun-connected dependency edges, and we construct feature count vectors for each of the most frequently occurring 163,198 individual nouns.

After constructing these feature vectors, similarly to Pantel [105] we normalize these feature counts with pointwise mutual information. The pointwise mutual information [86] between two events x and y is defined as the logarithm of the ratio of the joint probability of x and y to the product of the prior probabilities of each of x and y:

$$pmi(x, y) = \log\left(\frac{P(x, y)}{P(x)P(y)}\right)$$

This quantity has the property of being equal to 0 when x and y are independent (i.e., when $P(x, y) = P(x)P(y)$), positive when x and y are positively correlated, and negative when the two events are negatively correlated. If we denote the count of times that a word x occurs with a syntactic feature f as c_{xf}, and if we denote the frequency of all features of all words as:

$$N = \sum_i \sum_j c_{ij}$$

Then we can calculate our estimate of the pointwise mutual information of x and f as:

$$pmi(x, f) = \log\left(\frac{\frac{c_{xf}}{N}}{\frac{\sum_i c_{if}}{N} \times \frac{\sum_j c_{xj}}{N}}\right)$$

Thus, if the initial raw feature vector for a word is $\mathbf{x}$, where a given element x_i contains the frequency of dependency path f_i with that word, then we create a modified representation $\mathbf{x}'$ of the feature vector where each entry corresponds to the pointwise mutual information between the word and the feature. Having created our representation of the pointwise mutual information of the words and features in $\mathbf{x}'$, we then compute the cosine coefficient [131] between these normalized vectors in order to obtain our measure of distributional similarity. That is, for two normalized feature vectors $\mathbf{x}'$ and $\mathbf{y}'$, we compute:

$$sim(\mathbf{x}', \mathbf{y}') = \frac{\sum_j \left(x'_j \times y'_j\right)}{\sqrt{\sum_j \left(x'^2_j\right) \times \sum_j \left(y'^2_j\right)}}$$

We use this normalized similarity score for coordinate term classification.

2.7.2 Conjunction pattern classifier

Next we consider a simple pattern-based classifier based on the conjunction pattern, which simply ranks a pair of entities based on the total number of conjunction dependency paths found by Minipar between that entity pair. That is, if we consider the feature vector $\mathbf{x}$ consisting of the frequency counts of all dependency paths observed between two words w_i and w_j, then our similarity score is proportional to the element x_{coord} corresponding to the frequency count of the"conjunction pattern" (X, -N:conj:N, Y). Examples of natural text from which the conjunction pattern is inferred are the expressions "X and Y" and "X or Y". That is, our conjunction classifier simply uses the similarity score:

$$sim(w_i, w_j) = x_{coord}$$

These two classifiers, respectively built from distributional similarity and the conjunction pattern, give us two independent methods for determining the likelihood that two words are linked through a coordinate relation.

2.7.3 WordNet classifiers

We would like to be able to compare the performance of our automatic classifiers for the coordinate term relation to the existing coordinate terms in WordNet. For purposes of performance comparison for labeling the coordinate terms from our hand-labeled testset we construct a series of classifiers using WordNet. These classifiers determine a word pair to be in a coordinate relation if they share a common hypernym ancestor within the WordNet hierarchy. We create multiple classifiers by varying the requirement of how close the hypernym must be to both nouns; for our evaluation we create six classifiers, where the k-th classifier determines that two nouns are coordinates if they share a hypernym of a distance no more than k from either noun in the hypernym taxonomy, for all k from 1 to 6.

Classifier	**F-score**
Interannotator Agreement:	0.6405
Distributional Similarity Vector Space Model:	0.3327
Thresholded Conjunction Pattern Classifier:	0.2857
Best WordNet Classifier:	0.2630

Table 2.4: Summary of maximum F-scores on hand-labeled coordinate pairs

2.7.4 Evaluation

We evaluate each of the proposed classifiers on the task of recognizing the coordinate terms in our hand-labeled testset described in Section 2.4. We display the precision/recall of each of the above classifiers in Figure 2.10, and give the maximum F-scores of each algorithm in Table 2.4. Since each of our coordinate classifiers produces continuous output scores for predicting the coordinate class, we can produce a full precision-recall curve for hypernym prediction by varying the prediction threshold for the classifier. That is, each point in the precision-recall plot corresponds to a specific score threshold T such that the classifier classifies instances as members of the positive class if they have probability greater than or equal T, and otherwise as members of the negative class.

We find that WordNet has higher precision than the automatic coordinate classifiers at low levels of recall, but that both the coordinate classifiers based on distributional similarity and the simple conjunction pattern perform as well or better than WordNet for recall greater than 20%, and both have a higher maximum F-score.

The strong performance of the simple conjunction pattern model suggests that it may be worth pursuing an extended pattern-based coordinate classifier along the lines of our full weighted hypernym classifier (rather than only using a single feature). In the remaining experiments we will continue using our distributional similarity classifier (with a 16% relative F-score improvement over the conjunction model) in the construction of a combined hypernym-coordinate hybrid classifier.

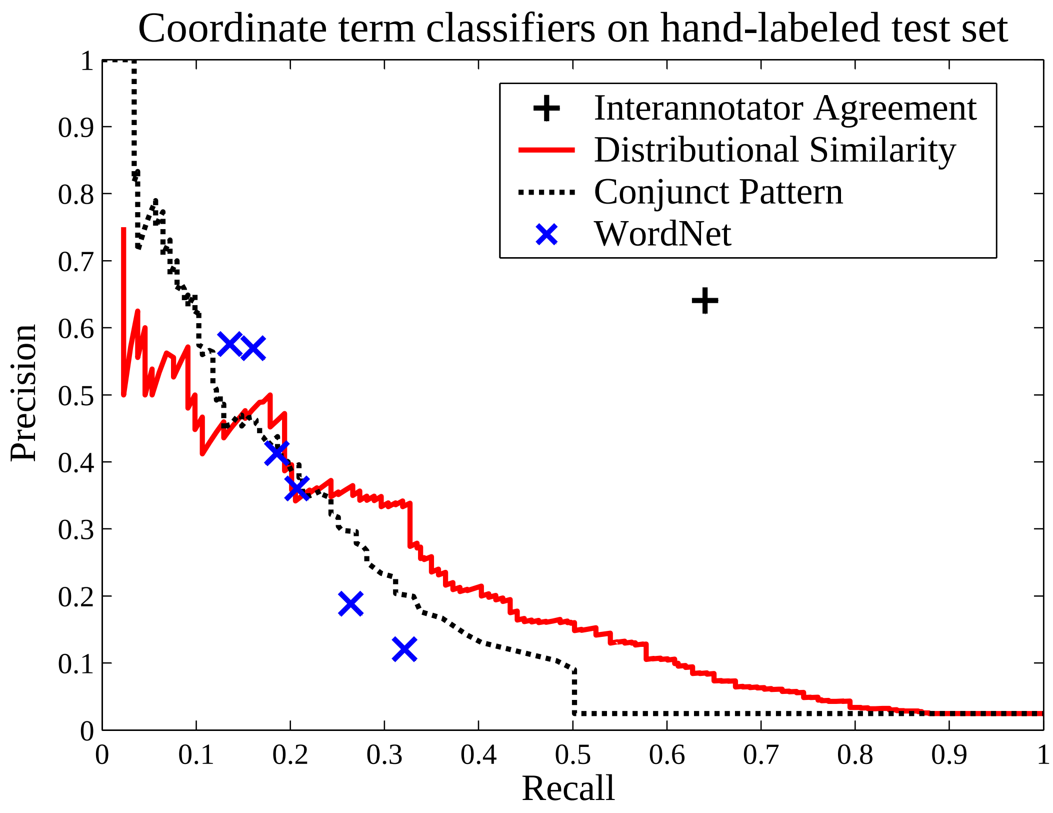


Figure 2.10: Coordinate classifiers on hand-labeled test set

2.8 Hybrid hypernym-coordinate classification

Next we experiment with combining our hypernym and coordinate models in order to improve hypernym classification. We define the outputs of our hypernym and coordinate classifiers as follows: first, we denote the probability that a noun n_i has n_j as an ancestor in its hypernym hierarchy as $P(n_i \underset{H}{<} n_j)$.

Second, we denote the probability that the nouns n_i and n_j are *coordinate terms*, i.e., that they share a common hypernym ancestor at some level, as $P(n_i \underset{C}{\sim} n_j)$. We estimate this probability using the similarity calculation of our distributional similarity vector space model from Section 2.7.1.

If we consider the initial probability estimate produced by our best hypernym-only classifier as $P_{old}(n_i \underset{H}{<} n_j)$, and the probability obtained by normalizing the similarity score from our coordinate classifier as $P(n_i \underset{C}{\sim} n_j)$, we then apply a simple linear interpolation scheme to compute a new estimate of the hypernym probability. Specifically, for each pair of nouns (n_i, n_k), we recompute the probability that n_k is a hypernym of n_i as:

$$P_{new}(n_i \underset{H}{<} n_k) \propto \lambda_1 P_{old}(n_i \underset{H}{<} n_k) + \lambda_2 \sum_j P(n_i \underset{C}{\sim} n_j) P_{old}(n_j \underset{H}{<} n_k)$$

We constrain our parameters λ_1, λ_2 such that $\lambda_1 + \lambda_2 = 1$; we set these parameters using 10-fold cross-validation on our hand-labeled test set. For our evaluation in Section 2.8.1 we use $\lambda_1 = 0.7$.

2.8.1 Evaluation

We would now like to evaluate how well our hypernym and hybrid hypernym-coordinate classifiers perform for recognizing hypernym pairs in newswire corpora. Our hand-labeled dataset allows us to compare the classifiers we have proposed to classifiers constructed using WordNet, as well as the previously discussed methods relying on manually-specified patterns. As in our evaluation of coordinate terms in Section 2.7.4, we construct several classifiers based on the WordNet hypernym taxonomy. We constructed many different classifiers by taking combinations of the following parameters:

- The maximum number of senses of a hyponym word for which to find hypernyms
- The maximum distance between the hyponym and its hypernym in the WordNet taxonomy

We display the six best WordNet classifiers we found by varying these parameters in Figure 2.11. Figure 2.11 shows the performance of each of the discussed method in a precision/recall plot, using the manually-constructed testset discussed in Section 2.4 as ground truth. We found that the WordNet model achieving the maximum F-score uses only the first sense of a hyponym and allows a maximum distance of 4 links between a hyponym and hypernym. We find that our logistic regression hypernym-only model trained on the newswire corpora has a 16% relative F-score improvement over the best WordNet classifier, while the combined hypernym/coordinate model has a 40% relative F-score improvement. Our best-performing classifier is a hypernym-only model additionally trained on the Wikipedia corpus, with an expanded feature lexicon of 200,000 dependency paths; this classifier shows a 54% improvement over WordNet. In Table 2.5 we list the maximum F-scores of each method.

In Table 2.6 we analyze the disagreements between the highest F-score WordNet classifier and our combined hypernym/coordinate classifier. There are 31 such disagreements, with WordNet agreeing with the human labels on 5 and our hybrid model agreeing on the other 26. We additionally inspect the types of noun pairs where our model improves upon WordNet, and find that at least 30% of our model's improvements are not restricted to Named Entities; given that the distribution of Named Entities among the labeled hypernyms in our test set is over 60%, this gives us hope that our classifier will perform well at the task of hypernym induction even in more general, non-newswire domains.

This evaluation demonstrates that automatic methods for hypernym classification can significantly outperform WordNet on the task of hypernym pair classification in newswire text. Further, our analysis of the disagreements between our classifier and WordNet shows that our automatic methods are not limited to only inferring a small class of hypernym relations, but rather infer general-purpose hypernym relations. These results are encouraging for the overall goal of being able to

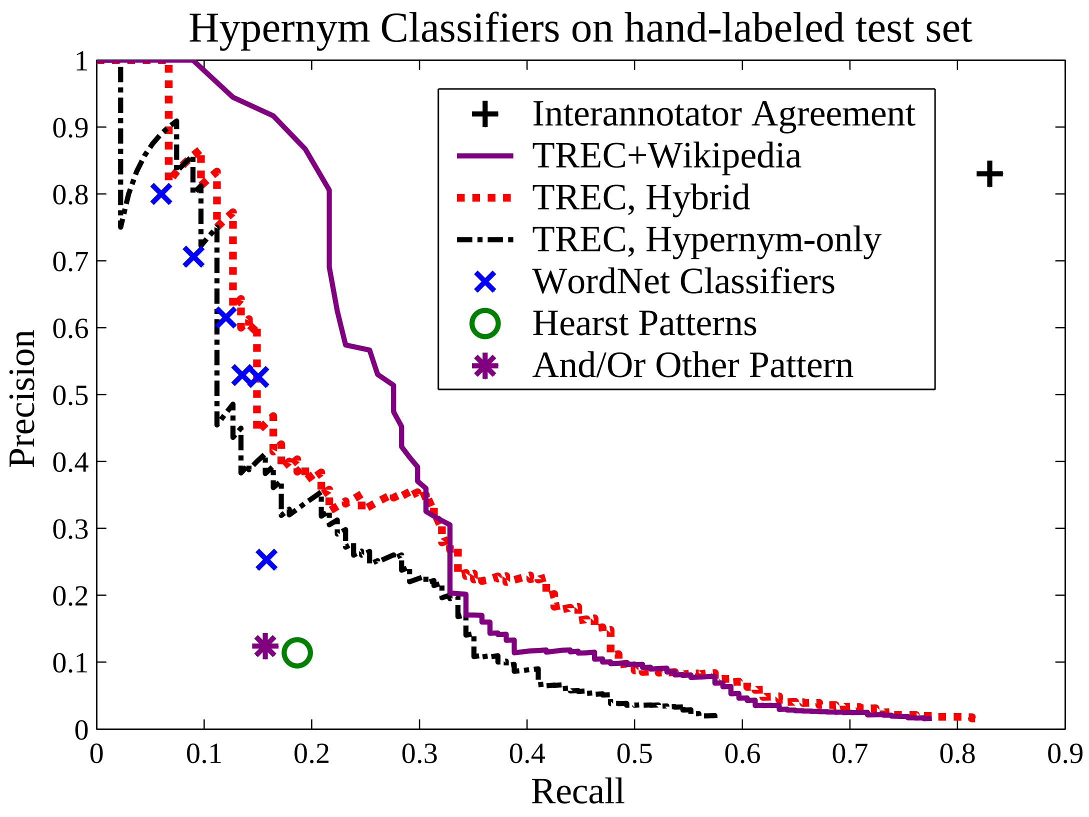


Figure 2.11: Hypernym classifiers on hand-labeled test set

infer structured knowledge from unstructured text at levels comparable to existing manually-constructed taxonomies.

2.9 List extraction for hypernym and coordinate prediction

So far we have considered the use of lexico-syntactic patterns and distributional similarity for the purpose of hypernym and coordinate classification. In this section we will consider semi-structured list information, an additional powerful source of evidence for taxonomic relations. List information has been successfully used for relation extraction in previous work [48, 22]. We will consider combined models of hypernym prediction and taxonomy augmentation using this evidence applied to the problem of

Classifier	**F-score**
Interannotator Agreement:	0.8318
TREC+Wikipedia Hypernym-only Classifier (Logistic Regression):	0.3592
TREC Hybrid Linear Interpolation Hypernym/Coordinate Model:	0.3268
TREC Hypernym-only Classifier (Logistic Regression):	0.2714
Best WordNet Classifier:	0.2339
Hearst Patterns Classifier:	0.1417
“And/Or Other” Pattern Classifier:	0.1386

Table 2.5: Maximum F-Score of hypernym classifiers on hand-labeled test set

Type of Noun Pair	Count	Example Pair
Named Entity: Person	7	“John F. Kennedy / president”, “Marlin Fitzwater / spokesman”
Named Entity: Place	7	“Diamond Bar / city”, “France / place”
Named Entity: Company	2	“American Can / company”, “Simmons / company”
Named Entity: Other	1	“Is Elvis Alive / book”
Not Named Entity:	9	“earthquake / disaster”, “soybean / crop”

Table 2.6: Analysis of improvements over WordNet

text classification in Section 6.3.

In this section we are primarily concerned with extraction of list items. Our general motivation for extracting the members of a lists that we expect members of a list to be similar to one another, and thus to give us evidence for the coordinate relation. Also, if we are able to extract the list header (i.e., a token indicating the *kind* of things in the list), we expect that this will give us evidence for the hypernym relation. This gives us an independent source of evidence for both the coordinate and hypernym relations, uncorrelated with our previously-discussed evidence sources.

2.9.1 Wrapper induction

In this section we propose a wrapper induction [74] algorithm for the purpose of list extraction. There are many ways in which a list can structurally indicate the items that belong to that list. For example, as shown in Figure 2.12, a list may specify its members by using of bullet point entries, or numbered entries, or a table of entries, or many other possible instantiations. We automatically infer the specific way that a list

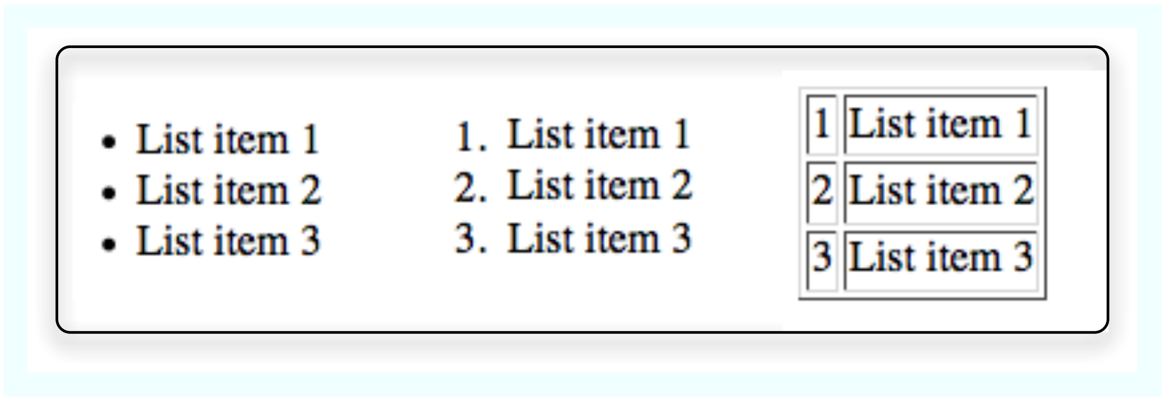


Figure 2.12: Three HTML list representations

identifies its members by automatically inferring the *wrapper*, or context that identifies the item's identity as a member of the list. Wrapper induction is a framework for automatically constructing extraction patterns to identify a particular set of items, typically within a structured or semi-structured context. In our experiments we will use wrapper induction specifically for inferring the *members* of lists.

Our algorithm for wrapper induction is similar in style and motivation to the list extraction algorithm used in the KnowItAll system [48]. For each run of our list extraction algorithm with initialize it with a set of seeds corresponding to the set of hyponyms of a particular noun synset in WordNet. For each term corresponding to a hyponym of a given synset, we issue a phrase query to a web search engine[4] to estimate the frequency of the term. We then rank the terms corresponding to the hyponyms of a synset by the number of results returned by the web search engine. This gives us an ordered list of hyponym terms per synset, with the most common terms appearing on top. We then search for these lists of hyponyms on the web, using the terms as seeds. We start off with the top k hyponym terms for each synset. We issue a query to a web search engine with these seed terms and consider the top-n results returned by the search engine which contain all of the seed terms. For our experiments we use $n = 20$, and we start with $k = 10$, then decrease by 1 if there are no search results, decreasing down until $k = 3$.

Our representation of the wrappers we infer uses the Document Object Model

[4]We use the Yahoo! Search API available at `http://developer.yahoo.com/search`.

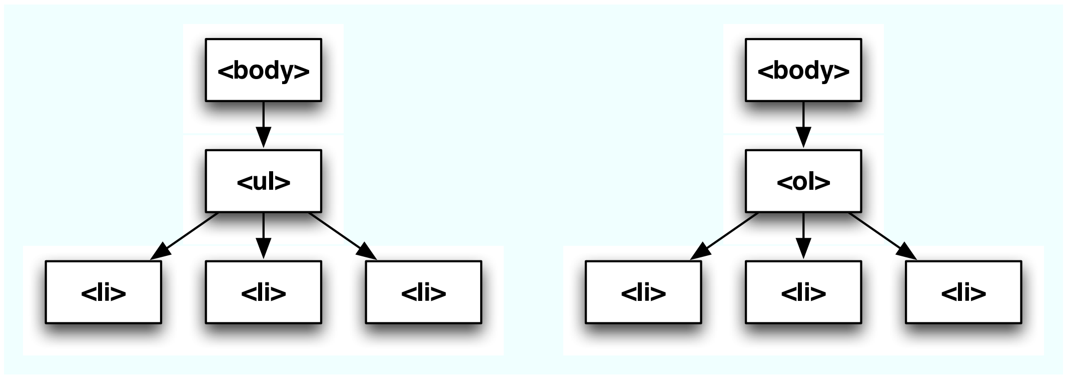


Figure 2.13: Document Object Model representations of unordered and ordered HTML lists

(DOM) tree of the webpage. The DOM tree is a representation of the structure of the webpage, using the tags within the HTML or XML markup to indicate hereditary relations among the objects in the page. As examples, selections of the DOM tree for the unordered and ordered lists given in Figure 2.12 are given in Figure **??**. After obtaining the results of our search, each document in the returned result set is processed as follows: first, the positions for the seed tokens in the Document Object Model (DOM) tree of the document are identified; second, a heuristic (described below) finds a pattern based on the DOM positions; finally, all terms found in DOM positions matching the above pattern are extracted.

The heuristic that determines the pattern of DOM nodes which contain possible list elements relies on an ordering of the nodes of the DOM tree. Each node in the DOM tree is labeled with the path that needs to be taken to reach it from the root of the DOM tree. This is done by associating child nodes with the label of the parent appended with the child-id for that node. If we start with a label of 'R' for the root for example, the first child of the root would be labeled 'R.1', the second child would be labeled 'R.2', the first child of 'R.1' would be 'R.1.1' and so on.

Once all nodes in the DOM tree are labeled, nodes that contain an exact match of any of the seeds are isolated, and a minimal pattern which matches all of them is determined. For example, if the selected nodes are the set { R.1, R.2, R.4, R.5 }, then the pattern generated is 'R.*d*', where *d* is a place-holder. Similarly, if the selected nodes are the set { R.1.1, R.2.1, R.3.1 }, then the pattern generated is 'R.*d*.1'. As an example, in the DOM trees of the lists displayed in Figure **??**, the "body" tag would form the root, and the corresponding list elements would be children of the "li" tags, or R.1.1.1, R.1.2.1, and R.1.3.1.

In the final step, all nodes that match the pattern are identified, and the elements contained in them are extracted. In our first example, the pattern 'R.*d*' identifies nodes labeled 'R.1', 'R.2', etc. if they exist. The text from these nodes are extracted and returned as possible candidates. A list of extracted terms is maintained by the system, to which the newly extracted terms are added. We repeat the process with a different query of unextracted terms until all terms in our synset have been covered by the extracted list, or we exhaust the list of seeds available.

Having retrieved the set of pages for the seed list and extracted the set of terms from each page, we may then identify the extracted elements as either *known-in-WordNet* or *unknown-in-WordNet*, and then pass these lists along to the list classifier described in Section 2.9.3.

2.9.2 Extracting list elements from Wikipedia

In addition to automatically discovering and extracting lists using wrapper induction, we also consider a set of lists contained with the popular web encyclopedia Wikipedia. We define the Wikipedia List corpus as the set of all articles in Wikipedia beginning with the term "List_of_". We process this corpus in the following manner: first, we assume that the heading of the list is a noun phrase, and parse this text as a noun phrase with the Stanford Parser. For example, if the title of the list is "List_of_male_tennis_players", we parse "male_tennis_players", and then order the set of nested noun phrases in order of decreasing length, i.e. {"male tennis players", "tennis players", "players"}. Having parsed this phrase, we then search in WordNet for the largest nested noun phrase from this phrase set known in WordNet. In our example, we would first search for "male tennis players", and then, not finding that term, search for "tennis players", which yields the noun synset "tennis_player" in WordNet 2.1.

Having matched the list header, we then identify the list elements from the Wikipedia page using a regular expression. We then similarly identify the list elements as either *known-in-WordNet* or *unknown-in-WordNet*, and then pass this list along to the list classifier described in Section 2.9.3. In our snapshot of Wikipedia we find 33,543 total lists; out of these we have 8,018 lists that contain a list header that is known in WordNet, and 18,198 lists that contain at least one list item that is known in WordNet.

2.9.3 Predicting taxonomic relations of list elements

Next we would like to use the list elements that we have inferred in order to predict actual taxonomic relations. We will discuss the application of our classifiers built from list information in Section 6.3. We construct a classifier for predicting relative taxonomic location based on list information using the notion of (m, n)-cousinhood in a hypernym taxonomy[5]. This notation states that that two synsets c_i and c_j are considered (m, n)-cousins if their closest *least common subsumer* (LCS), i.e., their closest shared hypernym, is within precisely m links from c_i and n links from c_j. In the work below we will refer to the probability of a particular (m, n)-cousinhood relation existing between two synsets c_i and c_j as $P\left(C_{ij}^{mn}\right)$. This notation gives us a general representation of probability distributions over neighborhoods within a taxonomy, rather than explicitly restricting the output of our classifiers to only relations of hypernym ancestry or immediately-shared hypernyms. We consider two classifiers for each list that we extract: a member-heading classifier and a member-member classifier.

The member-heading classifier

The member-heading classifier predicts the likelihood of a list member existing as the (m, n)-cousin of the extracted *heading* of the list; our hypothesis is that in general a member of a list is the hyponym of the heading of the list. In order to extract the heading of a list in the the Wikipedia List corpus, we parse the title of the list as a

[5]The notion of (m, n)-cousinhood is explained in more detail in Section 4.2.2

List item	**Hypernym chain**	**(m, n)-relation**
Babe Ruth	baseball player, athlete	(0,2)
Julius Erving	forward, basketball player, athlete	(0,3)
Wayne Gretzky	hockey player, athlete	(0,2)
Arthur Ashe	tennis player, athlete	(0,2)

Table 2.7: Athletes in both Wikipedia and WordNet, and (m, n)-relation to *athlete*

noun phrase, and then extract the largest noun phrase contained within the name of the page that is also known in WordNet. In the case of automated list extraction we do not attempt to directly classify a list heading, but rather we impose a list heading. Specifically we use the words of the initial WordNet synset from which the seed pairs were generated to discover the corresponding list.

For example, Wikipedia contains a list entitled *List of athletes on Wheaties boxes*. From this list title we extract the largest noun phrase which also occurs in WordNet, *athlete*. Extracting the list items and comparing to WordNet, we find many items on this list that also belong to WordNet: in Table 2.7 we list a subset of the list items that are also found in WordNet, along with their hypernym chain up to athlete, as well as their (m, n)-relation to the *athlete* concept in WordNet.

If we find that both the heading and some subset of the list items are known in WordNet, we may then use these discovered pairs to estimate the likelihood of list items existing in a particular (m, n)-cousinhood relation with the list heading. For example, from the items found in Table 2.7, we would expect our classifier to predict most strongly a (0-2)-cousinhood relation between the concept *athlete* and other items discovered in the *List of athletes on Wheaties boxes*. One option would be to use a simple maximum likelihood calculation to estimate the probability of an unknown list member being an (m, n)-cousin to the list heading as the fraction of all known list items that are observed to be in that $(m - n)$ relation with the heading. Specifically, for the particular list L, unknown list element c_i, list heading c_j, and known list elements c_k, we could estimate:

$$P_L\left(C_{ij}^{mn}\right) = \frac{\sum_{k \in L} 1\{C_{kj}^{mn}\}}{\sum_{k \in L} 1}$$

Here $1\{C_{kj}^{mn}\}$ is the indicator function that returns the value 1 when c_k and c_j are (m, n)-cousins and returns 0 otherwise. This probability for c_i and c_j being (m, n)-cousins is simply computing the fraction of known list elements that are found to be (m, n)-cousins already within WordNet. For example, if our training set was composed entirely of the four elements in Table 2.7, the maximum likelihood estimate would give us a probability of the (0,2)-cousinhood relation to *athlete* of 0.75, and the probability of the (0,3)-cousinhood of 0.25. This method of estimating probability can be seen as an extension of the *distant supervision* approach to learning taxonomic relations; here we are automatically labeling the extracted lists using the distant supervision of WordNet, and then training a simple probabilistic model for predicting (m, n)-cousinhood based on this distant supervision.

One major problem with this approach is the sparsity of the list extraction classifier. That is, there are several cases where very few members of a list are already known to be in WordNet. Further, many lists are short, and thus do not provide a large amount of data to allow a classifier to make confident predictions. Due to the sparsity of our list extraction classifier we propose incorporating a simple prior probability over the set of all lists that our list extraction classifier discovers. To compute this prior across our list member-header classifer we count the total number of known list items that are found to be (m, n)-cousins of their list heading for each C^{mn} with $m, n \leq 3$. We then incorporate this prior in our estimate of individual list probabilities by adding a single pseudocount split among the possible classes according to this prior distribution. That is, for the member-header classifier for a specific list L we predict the probability of an unknown list element c_i being an (m, n)-cousin of the known list header element c_j, given known list items c_k, as:

$$P_L\left(C_{ij}^{mn}\right) = \frac{\lambda_{mn} + \sum_{k \in L} 1\{C_{kj}^{mn}\}}{1 + \sum_{k \in L} 1}$$

Here λ_{mn} is the smoothing parameter corresponding to the prior probability of a known lit member being an (m, n)-cousin of the list header. Thus if we represent the list header of a given list L as $h(L)$, we calculate the smoothing parameter as:

List member 1	**List member 2**	**(m,n)-relation**
Babe Ruth	Julius Erving	(2,3)
Babe Ruth	Wayne Gretzky	(2,2)
Babe Ruth	Arthur Ashe	(2,2)
Julius Erving	Wayne Gretzky	(2,3)
Julius Erving	Arthur Ashe	(2,3)
Wayne Gretzky	Arthur Ashe	(2,2)

Table 2.8: Athletes in both Wikipedia and WordNet, and (m,n)-relation to each other

$$\lambda_{mn} = \frac{\sum_{\forall L: k \in L, j=h(L)} 1\{C^{mn}_{kl}\}}{\sum_{\forall L: k,l \in L} 1}$$

That is, λ_{mn} is calculated as the prior probability of any known list member c_k being in a C^{mn} relation with the list header $h(L)$ over all lists with known headers. Using this smoothing parameter has the desirable effect of backing off to the list member-header prior when there are no observed pairs from WordNet in a given list.

The member-member classifier

Whereas we used the member-heading classifier to predict the relative location of list member and a list header, we use the member-member classifier to predict the relative taxonomy position of any two members of a particular list. We expect that two elements of a list are likely to share a hypernym within close proximity; we estimate the actual likelihood of these elements sharing a hypernym by iterating through the set of pairs of list members where both items are known in WordNet and performing a calculation of what (m,n)-cousinhood relation the two corresponding synsets have in WordNet. In this way we may build an estimate for the probability of any two items in a particular list being C^{mn} cousins conditioned on their mutual presence in a list together. Again this is an extension of the *distant supervision* approach, now applied to automatically labeling the members of a list using the distant supervision of WordNet.

The member-member classifier probabilities are calculated in an identical fashion to the member-header classifier, except that instead of predicting the likelihood of a list member being in a specific taxonomic location relative to the list header, it is with respect to any other list member. For example, if our training set consisted of the four list members specified in Table 2.7, we would end up with the six pairwise (m, n)-relations given in Table 2.8. If we used only a simple maximum likelihood estimate we would calculate from this list that the probability of any two members on this list having a (2,2)-relation would be 0.50, and having a (2,3)-relation would also be 0.50. Again, however, we also add a single pseudocount corresponding to the prior distribution of all observed pairs of list members across all lists. That is, just as in the member-header classifier we predict the likelihood of the two synsets indicated by the list items being in a C^{mn} cousinhood relation as the maximum likelihood estimate based on the known elements of that list, with a single pseudocount distributed over the observed prior distribution of (m, n)-cousinhood relations over all lists.

2.9.4 Discussion

In this section we have presented two classifiers for predicting taxonomic relations from list information: a *member-header* classifier and a *member-member* classifier. These allow us to obtain information from a very different evidence source than our lexico-syntactic and distributional similarity classifiers. We use these classifiers as an important evidence source for constructing the largest of our augmented taxonomies, and we discuss the application of an augmented taxonomy constructed using these classifiers to the problem of text classification in Section 6.3.

2.10 Discussion

In this chapter we have proposed the use of *distant supervision* using an existing taxonomy in order to train relation extraction classifiers from unlabeled corpora. We have shown how this technique can be used to create hypernym and coordinate classifiers using lexico-syntactic patterns, coordinate classifiers from distributional similarity data, and hypernym and coordinate classifiers using semi-structured list data. Our experiments demonstrate that automatic methods can be competitive with WordNet for the identification of hypernym pairs in newswire corpora. These classifiers will be crucial in the following chapters, as they are the basis for our ability to create semantic taxonomies.

Chapter 3

Semantic Taxonomy Induction

3.1 Introduction

Accomplishing the task of relation extraction is necessary but not sufficient for effectively inferring structured knowledge from unstructured text. After running a successful algorithm for relation extraction many challenges still remain: first, there is the problem of combining evidence across multiple predictions. Much of the work in relation extraction has considered individual pairwise extractions as independent phenomena.

For example, Caraballo [23] infers a WordNet-like noun hypernym hierarchy by first performing a bottom-up clustering over words with high distributional similarity, and then attempting to label those clusters using two of Hearst's patterns. Specifically, Caraballo applies the Hearst Patterns ("NP_X *and other* NP_Y" and "NP_X *or other* NP_Y") to discover the potential hypernyms of the members of each cluster. In this way Caraballo created a hierarchy of over 20,000 nodes. In these experiments both the distributional similarity model and the hypernym extraction was performed on a corpus of text from the *Wall Street Journal*. A manual evaluation was then applied to 200 of the automatically extracted hyponym/hypernym pairs to determine the precision of using the distributional similarity and patterns for inferring a hypernym hierarchy. Here the evidence derived from distributional similarity and the evidence derived from application of the Hearst patterns are used in an entirely independent manner; we would hope that the precision of this technique would increase if an algorithm was able to jointly infer taxonomic relations over both sets of evidence.

While less common, some work has considered the use of probabilistic models for combining multiple predictions. For example, the KOG system developed by Wu and Weld [154] uses a Markov Logic Network in order to perform joint inference over several relations. While there has been a wide variety of approaches to joint inference in information extraction, record linkage, and entity resolution tasks [19, 152, 117], few have exploited the particular transitive properties of structured relations. Related work includes the noun hypernym hierarchy constructed in [23], the part-whole taxonomies constructed in [55], and a variety of related experiments described in [18]; however, these approaches typically consider only a single relation, or if they use multiple sources of evidence (as in [23] where both distributional similarity information and hypernym information is used), they are used in a linear pipeline, without the possibility of two-way interaction.

Lexical ambiguity is an additional challenge typically unaddressed by relation extraction algorithms. More generally, the accurate identification of the intended sense of a word in context is an extremely challenging and unsolved problem [71, 58, 104]. While still difficult, the challenge of identifying the intended sense of a word participating in a particular taxonomic relation is easier in some ways than the problem of disambiguating the specific intended sense of a word in a particular corpus instance. In the formulation of relation prediction that we propose, the decision to infer a particular relation is typically supported by several related predictions. These related predictions often interact in such a way as to pinpoint the intended sense of each word in the context of the inferred relations.

As an example, suppose our hypernym classifier is extracting relations from a document about airline companies. Suppose it encounters text such as "Continental is one of the few carriers to report monthly RASM data..." and "Continental has become the first U.S. carrier to cut flights...", and from these examples and others predicts that the noun *Continental* is a kind of a *carrier*. This seems like a positive prediction; however, the problem of lexical ambiguity makes this a difficult piece of information to integrate with an existing taxonomy. In WordNet, for example, there are 11 different senses of the noun *carrier*, and only one of them (*carrier#n#5* actually pertains to the sense of the airline company *Continental*. The other senses include "someone whose employment involves carrying something", a "vehicle designed specifically to carry something", "a large warship that carries planes", and so forth. Thus, in order to distinguish the appropriate sense of *Continental* we need further information—perhaps that *continental* is also a kind of *business*, or that it is similar to *United*, or something along these lines.

Most relation extraction algorithms do not attempt to disambiguate the actual word sense intended in the particular usage that a relation is extracted from; rather, attempts to reconcile extracted relationships between pairs of words with an existing sense inventory are typically performed as a post-processing step, if performed at all. Many approaches have simply chosen the first-listed sense in WordNet when reconciling a given word with a sense within the taxonomy; this is equivalent to using only the prior distribution of senses as obtained from the SemCor corpus, since WordNet lists word senses for a specific in order of the number of times that word occurred within the SemCor corpus. For example, this approach is taken by Suchanek et al. [143] in constructing the extended ontology YAGO, where the authors state:

> "We found out that mapping the head compound simply to the most frequent synset ... yields the correct synset in the overwhelming majority of cases".

Similarly, Wu and Weld [154] use this approach in their Kog system, stating that

> "[W]e select the most frequent sense of the mapped node in WordNet, which turns out to work well in most cases".

Other work, however, has found that lexical ambiguity is sometimes more problematic than these comments might suggest. For example, Soderland and Mandhani [141] found that in their work in "ontologizing" the output of their relation extraction algorithm, the baseline word sense disambiguation algorithm of always selecting the most frequent WordNet synset for extracted relations had an accuracy of only 55%. To combat this problem, the authors proposed a more creative solution to the problem of lexical ambiguity: they attempted to determine which sense in WordNet would effectively have the highest distributional similarity to the extracted entities using a context-based distributional similarity score between extracted entities and words related in WordNet. For example, if their relation extraction model extracted the relation "orange contains vitamin C", then they would construct a context vector of up to four words on either side of both "orange" and "vitamin C" by aggregating contexts across all sentences from which the relation was extracted. They then used a pointwise mutual information-based metric to compare the vectors constructed for the entities in the extracted relations to each of the synonyms, siblings and direct hyponyms of the different senses that the entities might map to, and selected the sense for which the similarity is highest. They found that this method significantly increased their accuracy of assigning the appropriate sense; by doing so they increased their sense disambiguation accuracy by 15%, to 70%. While the technique used by Soderland and Mandhani [141] is clearly superior to simply choosing the first sense in WordNet, there are still several sources of evidence that remain to be taken advantage of; we discuss a method for doing so in the following sections.

In the following sections we discuss a unified framework for joint inference over predicted relations capable of simultaneously increasing both consistent relation prediction accuracy and robust sense disambiguation. In this Section we have discussed related work in ontology induction; in Section 3.2 we outline the basic definitions of taxonomies, relations, and concepts; in Section 3.3 we give a probabilistic means for calculating taxonomy likelihood; in Section 3.4 we show how to extend the simple model to calculate evidence among word senses rather than simply word tokens; in Section 3.5 we discuss a few basic operations on taxonomies and their effects on the estimate of taxonomy likelihood; and in Section 3.6 we summarize our model and conclude.

3.2 Taxonomies, relations, and constraints

We define a taxonomy $\mathbf{T}$ as a function that maps each pair of concepts (c_i, c_j) in some domain of objects $\mathbf{C_T}$ to some (possibly empty) set of pairwise relations R_{ij} out of a domain of possible relations $\mathbf{R}$. For example, the domain of relations in WordNet include *hypernymy*, *holonymy*, *verb entailment*, and many others; the domain of concepts between which these relations hold in WordNet are its word senses or *synsets*.

We define that each relation $R_{ij} \in \mathbf{R}$ is a set of ordered or unordered pairs of objects $(c_i, c_j) \in \mathbf{C_T}$. For example, consider the hypernym relation "*Shakespeare* is a *playwright*" in WordNet: *Shakespeare* and *playwright* are represented by synsets in WordNet, and the relation between them in is the hypernym relation. Thus, if there are n total objects in the domain of concepts $\mathbf{C_T}$, we can represent a taxonomy $\mathbf{T}$ as an n by n matrix, where each element T_{ij} of the matrix lists the set of relations between elements $(c_i, c_j) \in \mathbf{C_T}$.

That is:

$$\mathbf{T} = \begin{pmatrix} R_{11} & R_{12} & \dots & R_{1n} \\ R_{21} & R_{22} & \dots & R_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ R_{n1} & R_{n2} & \dots & R_{nn} \end{pmatrix}$$

Typically this matrix is quite sparse; for example, if we consider the noun hierarchy in WordNet, the total size of concepts $|\mathbf{C_T}|$ is about 80,000; thus, the total number of entries in the matrix $\mathbf{T}$ is more than six billion, i.e., $(8 \times 10^4)^2$, or 6.4×10^9. However, the vast majority of these pairs have no direct relation in WordNet; the total number of direct hypernym relations is approximately 82,000 (nearly all noun hypernyms have only a single parent, but a small fraction have multiple parents), and the other relations are more sparsely populated, with about 20,000 meronym (part-whole) relations, and about 2,000 antonym relations.

A semantic taxonomy can enforce certain *taxonomic constraints* which disallow particular taxonomies. For example, the hypernym relation in WordNet is required to be transitive and acyclic (you cannot have a dog as a kind of a canine, and a canine as a kind of a dog); the hypernym transitivity constraint in WordNet requires that each synset inherits the hypernyms of its hypernym. Similarly, the meronym (part-whole) relation is acyclic, and the part-inheritance constraint requires that each synset inherits the meronyms of its hypernyms.

We can express these contraints more formally using the following notation: we can write a hypernym relation between two synsets (c_i, c_j) as H_{ij}; further, if c_i and c_j are separated by n steps in the hypernym chain, we can write the relation as H_{ij}^n, where H_{ij}^1 represents the direct hypernym relation. Similarly, we can represent the meronym relation between c_i and c_j as M_{ij}. Since meronyms are inherited from a synset's hypernym(s), there are two possible superscripts to keep track of: first, a synset can be a part of a part, so it is possible to be m steps away in the meronym chain alone. Second, a meronym may be inherited from an arbitrary hypernym n steps away. Thus we represent a meronym that is inherited from a synset n steps away in the hypernym chain, and the meronym itself is m steps away in the meronym chain as $M_{ij}^{m,n}$. Thus, for example, a direct meronym link will be represented as $M_{ij}^{1,0}$, whereas a meronym with a direct link to a synset's immediate parent will be $M_{ij}^{1,1}$.

Using this notation we can express the hypernym and meronym constraints formally as follows:

1. Hypernym Acyclicity Condition:

$$\forall n, m. H_{ij}^n \Rightarrow \neg H_{ji}^m$$

2. Hypernym Transitivity:

$$H_{ij}^m \wedge H_{jk}^n \Rightarrow H_{ik}^{m+n}$$

3. Meronym Acyclicity Condition:

$$\forall n, m. M_{ij}^n \Rightarrow \neg M_{ji}^m$$

4. Meronym Transitivity:

$$M_{ij}^{(m,n)} \wedge M_{jk}^{(m,o)} \Rightarrow M_{ik}^{(m,n+o)}$$

$$H_{ik}^{m} \wedge M_{jk}^{(n,o)} \Rightarrow M_{ik}^{(m+n,o)}$$

Constraint (1) here requires that no pair of synsets can be hypernyms of one another, and constraint (2) requires that each synset inherits the hypernyms of its direct hypernym. Similarly, contraint (3) requires that no pair of synsets can be meronyms of one another, and constraint (4) requires that each synset inherits the holonyms of its holonym, and also that each synset inherits the meronyms of its hypernym. These transitivity constraints imply that the addition of any new hypernym or meronym relation to a preexisting taxonomy will typically necessitate the addition of a set of other novel relations as implied by the taxonomic constraints. We will refer to the full set of novel relations implied by a particular relation link R_{ij} as $\mathbf{I}(R_{ij})$.

3.3 Taxonomy likelihood

We propose that the event $R_{ij} \in \mathbf{T}$ has some prior probability $P(R_{ij} \in \mathbf{T})$, and $P(R_{ij} \in \mathbf{T}) + P(R_{ij} \notin \mathbf{T}) = 1$. We define the probability of the taxonomy as a whole as the joint probability of its component relations; given that the taxonomy $\mathbf{T}$ is defined over relations over all pairs of concepts, we can express this as:

$$P(\mathbf{T}) = P(R_{11}, R_{12}, \ldots, R_{nn}) = P(\bigcup_{i,j} R_{ij})$$

Next, we assume that we have some set of observed evidence $\mathbf{E}$ consisting of observed features over pairs of objects in some domain $\mathbf{C_E}$; we'll begin with the assumption that our features are over pairs of words, and that the objects in the taxonomy also correspond directly to words.[1] We further assume that we can represent the full set of evidence $\mathbf{E}$ as the union of several distinct individual pieces of evidence,

[1]In the next subsection we relax this assumption, extending this model to manage lexical ambiguity.

each corresponding to a particular pair of concepts (c_i, c_j) and a specific relation R_{ij}, represented as E^R_{ij}, that is:

$$\mathbf{E} = \bigcup_{R,i,j} E^R_{ij}$$

Given a set of features $E^R_{ij} \in \mathbf{E}$, we assume we have some model for inferring $P(R_{ij} \in \mathbf{T} | E^R_{ij})$, i.e., the posterior probability of the event $R_{ij} \in \mathbf{T}$ given the corresponding evidence E^R_{ij} for that relation. For example, evidence for the hypernym relation E^H_{ij} might be the set of all observed lexico-syntactic patterns containing the words w_i and w_j in all sentences in some corpus. We make no assumptions here about how these estimates of $P(R_{ij} \in \mathbf{T} | E^R_{ij})$ are calculated; in this way we make the claim that this formulation of taxonomy likelihood is independent of the particular relation extraction methods used. All that is required is that a particular relation extraction algorithm produces probabilistic estimates of the form $P(R_{ij} \in \mathbf{T} | E^R_{ij})$ for some set of tokens or concepts $\mathbf{C_E}$ or $\mathbf{C_T}$. Thus, we may observe that the output of each relation classifier is actually a matrix of a similar form to the matrix for the taxonomy $\mathbf{T}$ itself, i.e., if there are n concepts in $\mathbf{C_E}$, then we can represent the output of the relation classifier as the matrix:

$$\begin{pmatrix} P(R_{11}|E) & P(R_{12}|E) & \dots & P(R_{1n}|E) \\ P(R_{21}|E) & P(R_{22}|E) & \dots & P(R_{2n}|E) \\ \vdots & \vdots & \ddots & \vdots \\ P(R_{n1}|E) & P(R_{n2}|E) & \dots & P(R_{nn}|E) \end{pmatrix}$$

For simplicity we make the following independence assumptions: first, we assume that each item of observed evidence E^R_{ij} is independent of all other observed evidence given the taxonomy $\mathbf{T}$, i.e.,

$$P(\mathbf{E}|\mathbf{T}) = \prod_{E^R_{ij} \in \mathbf{E}} P(E^R_{ij}|\mathbf{T})$$

This can be seen as a sort of "Naive Bayes" assumption; this is justified when the actual individual sets of evidence used by the underlying relation classifiers are uncorrelated from one another. This assumption is violated, however, if multiple relation classifiers use correlated evidence; for example, multiple relation classifiers could use exactly the same evidence to predict the same relation, and the violation of this assumption would result in an overconfident prediction.

Further, we assume that each item of observed evidence E_{ij}^R depends on the taxonomy $\mathbf{T}$ only by way of the corresponding relation R_{ij}, i.e.,

$$P(E_{ij}^R|\mathbf{T}) = \begin{cases} P(E_{ij}^R|R_{ij} \in \mathbf{T}) & \text{if } R_{ij} \in \mathbf{T} \\ P(E_{ij}^R|R_{ij} \notin \mathbf{T}) & \text{if } R_{ij} \notin \mathbf{T} \end{cases}$$

For example, if our evidence E_{ij}^H is a set of observed lexico-syntactic patterns indicative of hypernymy between two words i and j, we assume that whatever dependence the relations in $\mathbf{T}$ have on our observations may be explained entirely by dependence on the existence or non-existence of the single hypernym relation $H(i, j)$.

Applying these two independence assumptions we may express the conditional probability of our evidence given the taxonomy:

$$\begin{aligned} P(\mathbf{E}|\mathbf{T}) = & \prod_{R_{ij} \in \mathbf{T}} P(E_{ij}^R|R_{ij} \in \mathbf{T}) \\ & \cdot \prod_{R_{ij} \notin \mathbf{T}} P(E_{ij}^R|R_{ij} \notin \mathbf{T}). \end{aligned}$$

Rewriting the conditional probability in terms of our estimates of the posterior probabilities $P(R_{ij}|E_{ij}^R)$ using Bayes Rule, we obtain:

$$\begin{aligned} P(\mathbf{E}|\mathbf{T}) = & \prod_{R_{ij} \in \mathbf{T}} \frac{P(R_{ij} \in \mathbf{T}|E_{ij}^R)P(E_{ij}^R)}{P(R_{ij} \in \mathbf{T})} \\ & \cdot \prod_{R_{ij} \notin \mathbf{T}} \frac{P(R_{ij} \notin \mathbf{T}|E_{ij}^R)P(E_{ij}^R)}{P(R_{ij} \notin \mathbf{T})}. \end{aligned}$$

Within our model we define the goal of taxonomy induction to be to find the taxonomy $\hat{\mathbf{T}}$ that maximizes the conditional probability of our observations $\mathbf{E}$ given the relationships of $\mathbf{T}$, i.e., to find:

$$\hat{\mathbf{T}} = \arg\max_{\mathbf{T}} P(\mathbf{E}|\mathbf{T}).$$

3.4 Managing lexical ambiguity

Since word senses are typically not directly observable, if the objects in the taxonomy are word senses (as in WordNet), we must extend our model to allow for a many-to-many mapping (e.g., a word-to-sense mapping) between $\mathbf{C_E}$ and $\mathbf{C_T}$. For this setting we assume we know the function senses(i), mapping from the word i to all of $i's$ possible corresponding senses. For example, in the case of predicting that *Continental* is a kind *carrier*, the function senses(i) applied to the word *carrier* would result in the 11 noun senses of *carrier* listed in WordNet.

We assume that each set of word-pair evidence E_{ij}^R we possess is in fact sense-pair evidence E_{kl}^R for a specific pair of senses $k_0 \in \text{senses}(i), l_0 \in \text{senses}(j)$. Further, we assume that a new relation between two words is probable only between the correct sense pair, i.e.:

$$P(R_{kl}|E_{ij}^R) = 1\{k = k_0, l = l_0\} \cdot P(R_{ij}|E_{ij}^R).$$

When computing the conditional probability of a specific new relation $R_{kl} \in \mathbf{I}(R_{ab})$, we assume that the relevant sense pair k_0, l_0 is the one which maximizes the probability of the new relation, i.e. for $k \in \text{senses}(i), l \in \text{senses}(j)$,

$$(k_0, l_0) = \arg\max_{k,l} P(R_{kl} \in \mathbf{T}|E_{ij}^R).$$

Our independence assumptions for this extension need only to be changed slightly; we now assume that the evidence E_{ij}^R depends on the taxonomy $\mathbf{T}$ via only a single relation between sense-pairs R_{kl}. Using this revised independence assumption the derivation for taxonomy likelihood remains unchanged. One side effect of this revised independence assumption is that the addition of the single "sense-collapsed" relation R_{kl} in the taxonomy $\mathbf{T}$ will explain the evidence E_{ij}^R for the relation over words i and j now that such evidence has been revealed to concern only the specific senses k and

l.

3.5 Operations over taxonomies

Among the most primitive operations one can define over a taxonomy are the addition and removal of relations between concepts. Here we define the ADD-RELATION and REMOVE-RELATION operations, and derive the effect of these operations on the taxonomy likelihood. Additionally we define the corresponding ADD-CONCEPT and REMOVE-CONCEPT operations in terms of the ADD-RELATION and REMOVE-RELATION operations. Finally we define the MERGE-CONCEPTS operation, which has the effect of combining two similar synsets.

3.5.1 Adding relations

The most primary operation over a taxonomy is the addition of a novel link between two synsets. Here we refer to the operation of adding a new link of relation type R between node c_i and node c_j in the taxonomy $\mathbf{T}$ as ADD-RELATION$(R_{ij}, \mathbf{T})$. In order to calculate how the taxonomy likelihood is effected by this operation, we may define the multiplicative change $\Delta_{\mathbf{T}}(R_{ij})$ to the conditional probability $P(\mathbf{E}|\mathbf{T})$ given the addition of a single relation R_{ij}; this may be expressed as:

$$\begin{aligned}
\Delta_{\mathbf{T}}(R_{ij}) &= P(\mathbf{E}|\mathbf{T}')/P(\mathbf{E}|\mathbf{T}) \\
&= \frac{P(R_{ij} \in \mathbf{T}|E_{ij}^R)P(E_{ij}^R)}{P(R_{ij} \notin \mathbf{T}|E_{ij}^R)P(E_{ij}^R)} \cdot \frac{P(R_{ij} \notin \mathbf{T})}{P(R_{ij} \in \mathbf{T})} \\
&= k\left(\frac{P\left(R_{ij} \in \mathbf{T}|E_{ij}^R\right)}{1 - P\left(R_{ij} \in \mathbf{T}|E_{ij}^R\right)}\right).
\end{aligned}$$

Here k is the inverse odds of the prior on the event $R_{ij} \in \mathbf{T}$; we consider this to be a constant independent of c_i, c_j, and the taxonomy $\mathbf{T}$.

To enforce the taxonomic constraints in $\mathbf{T}$, for each application of the ADD-RELATION operator we must add all new relations in the implied set $\mathbf{I}(R_{ij})$ not already in $\mathbf{T}$.[2] Thus we define the multiplicative change of the full set of implied relations as the product over all new relations:

$$\Delta_{\mathbf{T}}(\mathbf{I}(R_{ij})) = \prod_{R \in \mathbf{I}(R_{ij})} \Delta_{\mathbf{T}}(R).$$

In Section 4.3 we will discuss a search algorithm over the space of taxonomies using this operator.

3.5.2 Removing relations

Similarly to Add-Relation, we may define the operation of removing a single link of relation type R between two nodes c_i and c_j in a taxonomy $\mathbf{T}$ as Remove-Relation$(R_{ij}, \mathbf{T})$. In order to calculate how the taxonomy likelihood is effected by this operation, we may define the multiplicative change $\Delta_{\mathbf{T}}(R_{ij})$ to the conditional probability $P(\mathbf{E}|\mathbf{T})$ given the removal of a single relation R_{ij}; this may be expressed as:

$$\begin{aligned}\Delta_{\mathbf{T}}(R_{ij}) &= P(\mathbf{E}|\mathbf{T}')/P(\mathbf{E}|\mathbf{T}) \\ &= \frac{P(R_{ij} \notin \mathbf{T}|E_{ij}^R)P(E_{ij}^R)}{P(R_{ij} \in \mathbf{T}|E_{ij}^R)P(E_{ij}^R)} \cdot \frac{P(R_{ij} \in \mathbf{T})}{P(R_{ij} \notin \mathbf{T})} \\ &= k\left(\frac{1 - P\left(R_{ij} \in \mathbf{T}|E_{ij}^R\right)}{P\left(R_{ij} \in \mathbf{T}|E_{ij}^R\right)}\right).\end{aligned}$$

Here k is the odds of the prior of the event $R_{ij} \in \mathbf{T}$; as in the case of Add-Relation, we consider this to be a constant independent of i, j, and the taxonomy $\mathbf{T}$.

As before, the removal of a particular relation may potentially violate certain taxonomic constraints in $\mathbf{T}$; upon the application of the Remove-Relation operator

[2]For example, in order to add the new synset *microsoft* under the noun synset *company#n#1* in WordNet 2.1, we must necessarily add the new relations $H^2(microsoft, institution\#n\#1)$ $C^{11}(microsoft, dotcom\#n\#1)$, and so on.

we must remove all other relations previously implied by that single link. Thus we may again consider the set of implied relations by the relation to be removed as $\mathbf{I}(R_{ij})$; in order to express the removal of the single link, we will remove all such links already in $\mathbf{T}$.[3]

Thus we define the multiplicative change of removing the full set of implied relations as the product over all implied relations:

$$\Delta_{\mathbf{T}}(\mathbf{I}(R_{ij})) = \prod_{R \in \mathbf{I}(R_{ij})} \Delta_{\mathbf{T}}(R).$$

3.5.3 Adding concepts

The next most primitive operations we can apply to a taxonomy are to add and remove concepts. We can easily perform these operations using the previously defined Add-Relation and Remove-Relation operations. In particular, if we consider that we are adding a concept c_i with a set of relations $\mathbf{R}$, we need only to apply the Add-Relation operator to each relation in $\mathbf{R}$. In this notation we will state that the Add-Concept operator takes three arguments: the concept c_i being added, the set of relations $\mathbf{R}$ being added between the concept c_i and other concepts in the taxonomy, and the particular taxonomy $\mathbf{T}$ being modified. Using this terminology, we may write the effect of the Add-Concept operator as:

```
Add-Concept($c_i, \mathbf{R}, \mathbf{T}$) :
  $\forall R_{ij} \in \mathbf{R}$ :
      Add-Relation($R_{ij}, \mathbf{T}$)
```

[3]For example, in the inverse case of the Add-Relation operation above, if we *remove* a hypernym link between *microsoft* under the noun synset *company#n#1* in WordNet 2.1, we must necessarily remove the implied relations $H^2(microsoft, institution\#n\#1)$, $C^{11}(microsoft, dotcom\#n\#1)$, and so on.

3.5.4 Removing concepts

Similarly we may define the REMOVE-CONCEPT operation for a specific concept c_i by identifying the set of relations $\mathbf{R}$ that the concept c_i participates in, and then iteratively removing those relations from the taxonomy using the REMOVE-RELATION operator. As above we will state that the REMOVE-CONCEPT operator takes three arguments: the concept c_i being added, the set of relations $\mathbf{R}$ being added between the concept c_i and other concepts in the taxonomy, and the particular taxonomy $\mathbf{T}$ being modified. Using this terminology, we may write the effect of the REMOVE-CONCEPT operator as:

```
REMOVE-CONCEPT($c_i, \mathbf{R}, \mathbf{T}$) :
  $\forall R_{ij} \in \mathbf{R}$ :
      REMOVE-RELATION($R_{ij}, \mathbf{T}$)
```

3.5.5 Merging concepts

Having defined methods for adding and removing relations and concepts, we may build upon these operations in order to describe an additional MERGE-CONCEPTS operation. Here we will consider concepts and synsets interchangeably. There are several reasons for calculating whether two concepts, or synsets should be merged; there may be multiple synsets containing different words for the same concept, or the senses used in a particular taxonomy may be too fine-grained for a particular operation.

We define the MERGE-CONCEPTS operator so that it takes four arguments: the two concepts c_i and c_j to be merged, the set of all existing relations in the taxonomy in which concept c_j participates, and the taxonomy $\mathbf{T}$ being modified. With this terminology we may write the effect of the MERGE-CONCEPTS operator as a combination of the ADD-RELATION and REMOVE-SYNSET operators, as follows:

```
Merge-Concepts($c_i, c_j, \mathbf{R}, \mathbf{T}$) :
  $\forall R_{jk} \in \mathbf{R}$ :
    Add-Relation($R_{ik}, \mathbf{T}$)
  Remove-Synset($c_j, \mathbf{R}, \mathbf{T}$).
```

In this formulation we assume we have a classifier capable of predicting whether two synsets should be merged based on some set of evidence $\mathbf{E}$; we describe one such classifier in Chapter 5. We derive the effect of the Merge-Concepts operator on taxonomy likelihood and corresponding search algorithm for coarse-grained taxonomies in Chapter 5.

3.6 Discussion

We have presented a framework for inducing semantic taxonomies which attempts to globally optimize the entire structure of the taxonomy. Our probabilistic architecture also includes a model for learning coordinate terms based on (m, n)-cousin classification. The model's ability to integrate heterogeneous evidence from different classifiers offers the ability to formally quantify the probabilistic effects of several operations over a taxonomy, including adding and removing relations, and adding, removing, and merging concepts.

Chapter 4

Creating Augmented Taxonomies

4.1 Introduction

A primary motivation for our work in semantic taxonomy induction has been the goal of providing a means for automatically augmenting taxonomies from textual data. This chapter focuses on the largest of our projects in this direction: the set of augmented taxonomies that we have released as part of the Stanford Wordnet Project. The lexical resources discussed in this chapter can be found at the Stanford Wordnet Project homepage, at:

`http://ai.stanford.edu/~rion/swn`

In this chapter we discuss the application of our framework for taxonomy induction to the problem of automatically augmenting the WordNet taxonomy. In Section 4.2 we discuss the specific relations and taxonomic constraints in WordNet that we augment and use; in Section 4.3 we propose a search algorithm over the space of taxonomies using an operator for adding concepts; in Section 4.4 we discuss the relation classifiers we use in order to construct augmented taxonomies; in Section 4.5 we give examples of inference and give specific details of our implementation; in Section 4.6 we evaluate our taxonomy induction using a variety of performance measures; and finally in Section 4.7 we conclude with discussion of our results.

4.2 WordNet relations and constraints

In this section we discuss the relations and taxonomic constraints that we exploit in order to automatically augment WordNet. For the case of hyponym acquisition, the objects in our taxonomy are WordNet *synsets*. In this set of experiments we focus on augmenting WordNet by inferring novel noun hyponyms. We focus on the two WordNet relationships most relevant to hyponym induction: the hypernym relation and the coordinate term relation.

4.2.1 The WordNet hypernym relation

Here we discuss our notation of the hypernym relation. We use the notation H^n_{ij} to describe the relation between two concepts c_i and c_j if a sense c_j is the n-th ancestor of a sense c_i in the hypernym hierarchy. We will simply use H_{ij} to indicate that c_j is an ancestor of c_i at some unspecified level. As an example, in Figure 4.1 we have depicted a section of the WordNet noun hypernym hierarchy regarding animals. According to the relations depicted in Figure 4.1 we would state that *dog* and *canine* are in an H^1 relation, while *dog* and *mammal* are in a H^4 relation.

4.2.2 The WordNet coordinate term relation

Two senses are commonly considered to be "coordinate terms" or "taxonomic sisters" if they share an immediate parent in the hypernym hierarchy. As introduced in Section 2.9.3, we generalize this notion of siblinghood to state that two senses c_i and c_j are (m, n)-*cousins* if their closest *least common subsumer* (LCS) is within exactly m and n links, respectively. A least common subsumer $LCS(c_i, c_j)$ is defined as a synset that is an ancestor in the hypernym hierarchy of both c_i and c_j which has no child that is also an ancestor of both c_i and c_j. When there is more than one LCS (due to multiple inheritance), we refer to the *closest* LCS, i.e.,the LCS that minimizes the maximum distance to c_i and c_j. In this definition, an (m, n)-cousin for $m \geq 2$ corresponds to the English kinship relation "$(m - 1)$-th cousin $|m - n|$-times removed". We use the notation C^{mn}_{ij} to denote that i and j are (m, n)-cousins. Thus coordinate terms are

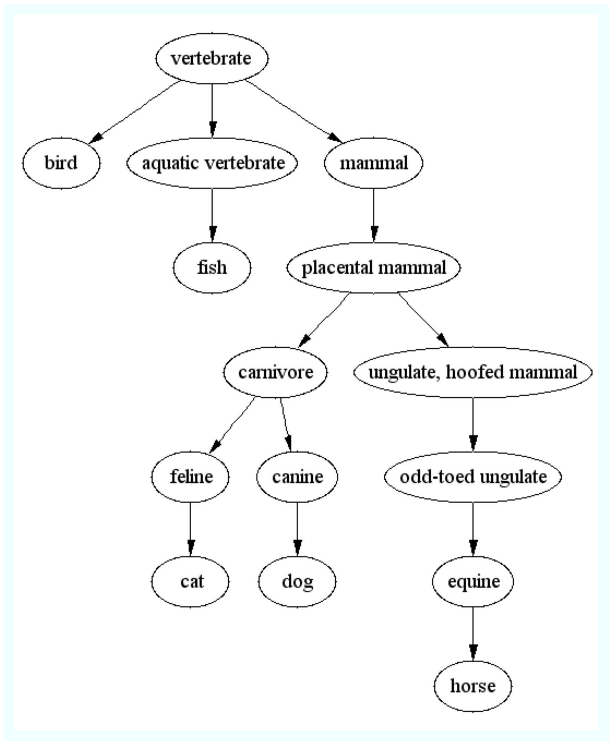


Figure 4.1: Example of (m, n)-relations among animal concepts in WordNet

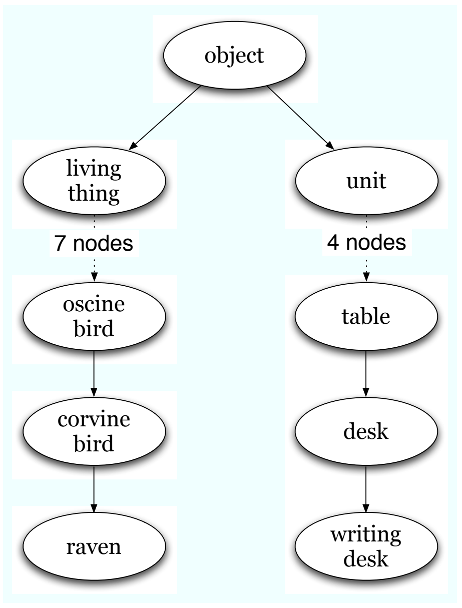


Figure 4.2: Example of (m, n)-relation between *raven* and *writing desk* in WordNet

$(1, 1)$-cousins; technically the hypernym relation may also be seen as a specific case of this representation; an immediate parent in the hypernym hierarchy is a $(1, 0)$-cousin, and the k-th ancestor is a $(k, 0)$-cousin.

As an example of (m, n)-cousins in WordNet, consider Figure 4.1. Here we have *dog* and *cat* in a C^{22} relation, *bird* and *fish* in a C^{12} relation, *bird* and *horse* in a C^{16} relation, and so forth.

As a sidenote, we can use the notion of (m, n)-cousins and WordNet to answer Lewis Carroll's riddle from *Alice in Wonderland*, "Why is a raven like a writing desk?"—albeit in a rather boring way. As demonstrated in Figure 4.2, we may say that *raven* and *writing desk* are $(8-11)$-cousins by way of their shared least common subsumer, *object*.

4.2.3 Taxonomic constraints

As discussed in 3.2, a semantic taxonomy such as WordNet enforces certain *taxonomic constraints* which disallow particular taxonomies $\mathbf{T}$. For example, the hypernym transitivity constraint in WordNet requires that each synset inherits the hypernyms of its hypernym, and the part-inheritance constraint requires that each synset inherits the meronyms of its hypernyms.

For the specific case of hyponym acquisition we are concerned with the following three taxonomic constraints on the hypernym and (m, n)-cousin relations: hypernym acyclicity, hypernym transitivity, and (m, n)-cousinhood:

1. Hypernym Acyclicity Condition:

$$\forall n, m. H^n_{ij} \Rightarrow \neg H^m_{ji}$$

2. Hypernym Transitivity:

$$H^m_{ij} \wedge H^n_{jk} \Rightarrow H^{m+n}_{ik}.$$

3. Definition of (m, n)-cousinhood:

$$C^{mn}_{ij} \Leftrightarrow \exists k. k = LCS(i, j) \wedge H^m_{ik} \wedge H^n_{jk}.$$

Constraint (1) requires that no pair of synsets can be in a hypernym relationship with each other, while constraint (2) requires that the each synset inherits the hypernyms of its direct hypernym, and constraint (3) simply defines the (m, n)-cousin relation in terms of the atomic hypernym relation.

The addition of any new hypernym relation to a preexisting taxonomy will usually necessitate the addition of a set of other novel relations as implied by the taxonomic constraints. We refer to the full set of novel relations implied by a new link R_{ij} as $\mathbf{I}(R_{ij})$. For example, if we were to add the single hyponym *dachshund* under the concept *dog* in Figure 4.1, there would be a large number of implied relations in addition to the direct H^1 relation with *dog*: there would be a H^2 relation with canine, an H^3 relation with *placental mammal*, an H^4 relation with *mammal*, as well as a C^{23} relation with *cat*, a C^{13} relation with *feline*, and so forth.

4.3 Searching over taxonomies

In Chapter 3 we stated that the goal of semantic taxonomy induction was to find the the taxonomy $\hat{\mathbf{T}}$ that satisfies the following equation:

$$\hat{\mathbf{T}} = \arg\max_{\mathbf{T}} P(\mathbf{E}|\mathbf{T}).$$

Here we propose a search algorithm for finding $\hat{\mathbf{T}}$ for the case of hyponym acquisition. We assume we begin with some initial (possibly empty) taxonomy $\mathbf{T}$. We restrict our consideration of possible new taxonomies to those created by the single operation ADD-CONCEPT$(R_{ij}, \mathbf{T})$.

As described in Section 3.5, ADD-CONCEPT adds the single concept c_i to $\mathbf{T}$; here we suppose that we are adding the single relationship R_{ij} between the concept c_i and the chosen hypernym c_j. As derived in Chapter 3, The multiplicative change $\Delta_{\mathbf{T}}(R_{ij})$ to the conditional probability $P(\mathbf{E}|\mathbf{T})$ given the addition of the single relation R_{ij} can be expressed as :

$$\begin{aligned}
\Delta_{\mathbf{T}}(R_{ij}) &= P(\mathbf{E}|\mathbf{T}')/P(\mathbf{E}|\mathbf{T}) \\
&= \frac{P(R_{ij} \in \mathbf{T}|E_{ij}^R)P(E_{ij}^R)}{P(R_{ij} \notin \mathbf{T}|E_{ij}^R)P(E_{ij}^R)} \cdot \frac{P(R_{ij} \notin \mathbf{T})}{P(R_{ij} \in \mathbf{T})} \\
&= k\left(\frac{P\left(R_{ij} \in \mathbf{T}|E_{ij}^R\right)}{1 - P\left(R_{ij} \in \mathbf{T}|E_{ij}^R\right)}\right)
\end{aligned}$$

Where k is the inverse odds of the prior on the event $R_{ij} \in \mathbf{T}$; we consider this to be a constant independent of i, j, and the taxonomy $\mathbf{T}$.

To enforce the taxonomic constraints in $\mathbf{T}$, for each application of the ADD-CONCEPT operator we must add all new relations in the implied set $\mathbf{I}(R_{ij})$ not already in $\mathbf{T}$. Thus we define the multiplicative change of the full set of implied relations as the product over all new relations:

$$\Delta_{\mathbf{T}}(\mathbf{I}(R_{ij})) = \prod_{R \in \mathbf{I}(R_{ij})} \Delta_{\mathbf{T}}(R).$$

This definition leads to the following best-first search algorithm for hyponym acquisition, which at each iteration defines the new taxonomy as the union of the previous taxonomy $\mathbf{T}$ and the set of novel relations implied by the relation R_{ij} that maximizes $\Delta_{\mathbf{T}}(\mathbf{I}(R_{ij}))$ and thus maximizes the conditional probability of the evidence over all possible single relations:

$$\textsc{While} \max_{R_{ij} \notin \mathbf{T}} \Delta_{\mathbf{T}}(\mathbf{I}(R_{ij})) > 1$$
$$\mathbf{T} \leftarrow \mathbf{T} \cup \mathbf{I}(\arg \max_{R_{ij} \notin \mathbf{T}} \Delta_{\mathbf{T}}(\mathbf{I}(R_{ij})))$$

We discuss the details of our specific implementation of local search in Section 4.5.

4.4 Use of relation classifiers

Here we describe how we use the hyponym and coordinate term classifiers in the creation of our augmented taxonomies.

4.4.1 Hyponym classification

Our classifier for the hypernym relation is derived from the "hypernym-only" classifier described in Chapter 2. The features used for predicting the hypernym relationship are obtained by parsing a large corpus of newswire and encyclopedia text with MINIPAR [81]. From the resulting dependency trees the evidence E^H_{ij} for each word pair (w_i, w_j) is constructed; the evidence takes the form of a vector of counts of occurrences of each labeled syntactic dependency path as the shortest path connecting w_i and w_j in some dependency tree. The labeled training set is constructed by labeling the collected feature vectors as positive "known hypernym" or negative "known non-hypernym" examples using WordNet 2.0; 49,922 feature vectors were labeled as positive training examples, and 800,828 noun pairs were labeled as negative training examples. The model for predicting $P(H_{ij}|E^H_{ij})$ is then trained using logistic regression, predicting the noun-pair hypernymy label from WordNet from the feature vector of lexico-syntactic patterns.

The hypernym classifier described above predicts the probability $P(H_{ij}|E^H_{ij})$ of the generalized hypernym-ancestor relation over *words*. This is typical of algorithms that attempt to predict relations from natural text, due to the difficulty of disambiguating the sense of a word in context. Nonetheless, being able to determine the appropriate sense of the word participating in the semantic relationship is critical to being able to integrate this knowledge into an existing taxonomy. Mistakenly choosing the wrong sense of the word to add could be disastrous for applications using that knowledge source; due to the transitivity of the relations in a knowledge base, a single mistaken disambiguation could result in many further mistaken inferences further down the pipeline. Thus for the purposes of taxonomy induction, we would much prefer to have a set of classifiers over *senses*, i.e., for $c_k \in \text{senses}(w_i), c_l \in \text{senses}(w_j)$, the set of classifiers estimating $\{P(H^1_{kl}|E^H_{ij}), P(H^2_{kl}|E^H_{ij}), \dots\}$.

One problem that arises from directly assigning the probability $P(H^n_{ij}|E^H_{ij}) \propto P(H_{ij}|E^H_{ij})$ for all n is the possibility of adding a novel hyponym to an overly-specific hypernym, which might still satisfy $P(H^n_{ij}|E^H_{ij})$ for some very large n. For example, we might predict that *San Francisco* is a kind of *city*, and that it is similar to *London*, and thus perhaps infer that *San Francisco* is a *capital city*. In order to discourage unnecessary overspecification, we penalize each probability $P(H^k_{ij}|E^H_{ij})$ by a factor λ^{k-1} for some $\lambda < 1$, and renormalize: $P_{new}(H^k_{ij}|E^H_{ij}) \propto \lambda^{k-1}P(H_{ij}|E^H_{ij})$. In our experiments we set $\lambda = 0.95$.

4.4.2 (m, n)-cousin classification

Our classifier for learning coordinate terms relies on the notion of *distributional similarity*, i.e., the idea that two words with similar meanings will be used in similar contexts [62]. We extend this notion to suggest that words with similar meanings should be near each other in a semantic taxonomy, and in particular will likely share a hypernym as a near parent. Distributional similarity is an extremely powerful source of evidence for semantic relationships between concepts. In particular it is a much denser form of evidence than our hypernym evidence; because we can collect evidence about the distributional properties of a word in every occurrence of that word, our distributional similarity classifier is capable of collecting evidence for taxonomic relations across many more words than our hypernym classifier.

Our classifier for predicting (m, n)-cousins from distributional similarity is derived from the algorithm and corpus given in [119]. In that work an efficient randomized algorithm is derived for computing clusters of similar nouns. First, the features for each noun are collected using a word window, adding the two words on the left and two words on the right of each noun as features; these features of co-occurring words are collected from a very large corpus of over 70 million web pages. Each element in the feature vector for each noun is computed as the pointwise mutual information between that noun and one of its co-occurring features, across all occurrences of that feature for all nouns.

We use a set of more than 1000 distinct clusters of English nouns collected by their algorithm over 70 million webpages, with each noun w_i having a score with respect to the cluster equal to its cosine similarity to the centroid c of the cluster to which it belongs, i.e., $\cos(\theta(w_i, c))$. As a preprocessing step we hand-edit the clusters to remove those containing non-English words, terms related to adult content, and other webpage-specific clusters. An example of some clusters generated by this algorithm for airlines, operating systems, and biological entities is given in Figure 4.3.

We use the cluster scores of noun pairs as input to our own algorithm for predicting the (m, n)-cousin relationship between the senses of two words w_i and w_j. If two words w_i and w_j appear in a cluster together, with cluster centroid c, we set our single coordinate input feature to be the minimum cluster score $\min(\cos(\theta(w_i, c)), \cos(\theta(w_j, c)))$, and zero otherwise. For each such noun pair feature, we construct a labeled training set of (m, n)-cousin relation labels from WordNet 2.1. We define a noun pair (w_i, w_j) to be a "known (m, n)-cousin" if for some senses $c_k \in \text{senses}(w_i), c_l \in \text{senses}(w_j)$, $C_{ij}^{mn} \in \text{WordNet}$. That is, we automatically label a noun pair as an (m, n)-cousin if we can find that pair connected in WordNet by that relation. In this way we are able to automatically construct a training set over an arbitrarily large unannotated corpus by using the distant supervision of WordNet. If more than one potential

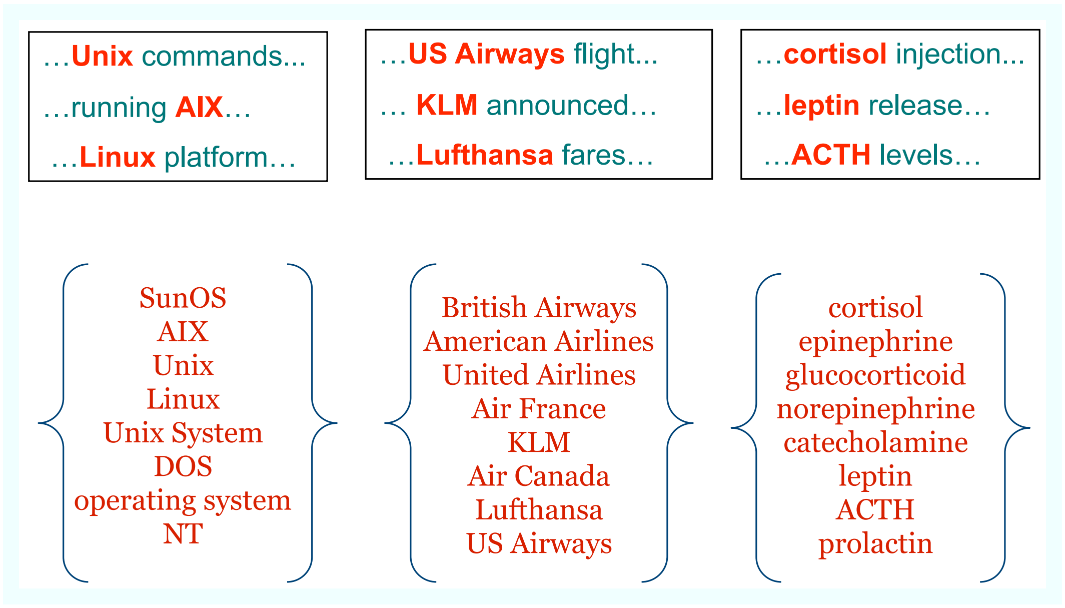


Figure 4.3: Examples of coordinate clusters generated from distributional similarity

(m, n)-cousinhood relation exists between the noun pair, we assume that (w_i, w_j) are an example of the (m, n)-cousinhood relation with smallest sum $m + n$, breaking ties by smallest absolute difference $|m - n|$.

For constructing our distantly-supervised training set for coordinate terms we consider all (m, n)-cousinhood relationships labeled from WordNet with $0 \leq m, n \leq 7$. Those pairs of words that don't have a corresponding pair of synsets connected within the hypernym hierarchy in WordNet, or which are connected but which have a least common subsumer with a distance of more than 7 hops away in the hypernym graph, are assigned to a single class C^∞ representing pairs of nouns that are "effectively unconnected" by a taxonomic relation in WordNet. Finally, we note that our measure for calculating distributional similarity is symmetric, and for the purpose of training our coordinate classifier we explicitly merge instances of (m, n)-cousinhood with their corresponding (n, m)-cousin instances. That is, we define each (m, n) class to be the union of (m, n)-cousins and (n, m)-cousins, regardless of order, i.e., $C^{mn} = C^{mn} \cup C^{nm}$. Due to this merging of the symmetric cousinhood classes, the resulting classifier for the cousinhood relations will predict, as expected given a symmetric input, $P(C^{mn}_{kl}|E^C_{ij}) =$

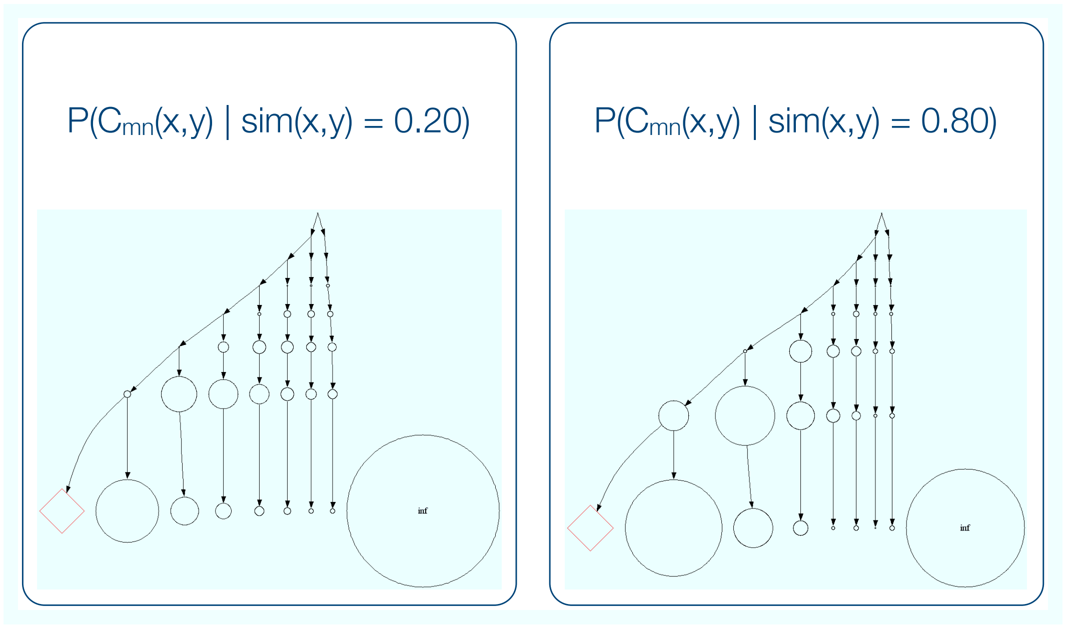


Figure 4.4: Visualization of probability of (m, n)-cousinhood given similarity score

$P(C^{nm}_{kl}|E^C_{ij})$.

We find 333,473 noun synset pairs in our training set with similarity score greater than 0.15. We next apply softmax regression to learn a classifier that predicts $P(C^{mn}_{ij}|E^C_{ij})$, predicting the WordNet class labels from the single similarity score derived from the noun pair's cluster similarity. A visualization of the probability of (m, n)-cousinhood given the distributional similarity score is shown in Figure 4.4. Here two similarity scores are visualized: 0.20 and 0.80. For each similarity score the probability of a term being an (m, n)-cousin is represented as proportional to the area of the circle in the corresponding relation to the red diamond in the lower left corner of the diagram. For example, for both similarity scores, the greatest probability is the ∞-cousin relation, or "no relation" within a distance of 7, followed by the $(1, 1)$-relation, followed by the $(2, 1)$-relation, and so forth. However, the relative proportions of a node being in a $(1, 1)$ or other close relation is much higher in the case of the higher similarity score.

4.5 Examples and discussion of implementation

Hyponym acquisition is among the simplest and most straightforward of the possible applications of our model; here we show how we efficiently implement our algorithm for this problem. First, we identify the set of all the word pairs (w_i, w_j) over which we have hypernym and/or coordinate evidence, and which might represent the addition of a novel hyponym to the WordNet 2.1 taxonomy (i.e., that has a known noun hypernym and an unknown hyponym, or has a known noun coordinate term and an unknown coordinate term). This yields a list of 95,000 single links over threshold $P(R_{ij}|\mathbf{E}) > 0.12$.

For each unknown hyponym w_i we may have several pieces of evidence; for example, for the unknown term *continental* we have 21 relevant pieces of hypernym evidence, with links to possible hypernyms $\{$*carrier, airline, unit, ...*$\}$; and we have 5 pieces of coordinate evidence, with links to possible coordinate terms $\{$*airline, american eagle, airbus, ...*$\}$.

For each proposed novel hyponym w_i, we explicitly compute the set of proposed candidate hypernyms for w_i based on the set of predicted hypernym and coordinate links involved with w_i. In practice, for each predicted hypernym link predicting the hypernym w_j, we add all senses of w_j to the set of considered hypernyms. For each predicted coordinate term w_k we add to set of considered hypernyms all senses of w_k, as well as the first two hypernym ancestors of each possible sense of w_k. Our choice to consider only these taxonomic relations for the set of candidate hypernyms is justified in an analysis of the probability distributions given by the hypernym and coordinate classifiers.

First, the hypernym classifier predicting the relation H^n_{kl} between two words w_k and w_l will predict w_l as the immediate hypernym with higher probability than any other potential word or concept due to the explicit construction of the classifier as discussed in Section 4.4.1. More formally, the hypernym classifier is constructed such that, considering only the evidence for the direct relation H^1_{kl}, $P(H^1_{kl}|E^H_{ij}) > P(H^n_{kl}|E^H_{ij})$ for all $n > 1$. Thus the immediately predicted word is the most relevant word to consider as an immediate hypernym when formulating the set of candidate hypernyms.

Second, the coordinate classifier predicts a distribution over C^{mn} classes as discussed in Section 4.4.2 and visualized in Figure 4.4. We may observe from Figure 4.4 that for high similarity pairs, the immediate coordinate (or (1,1)-cousin) is the most likely relation, followed by the (2-1)-cousin, followed by the (2,2)-cousin, and so forth. Therefore the most important possible hypernyms to consider are the immediate hypernym of the coordinate term and the second hypernym of the coordinate term; considering the coordinate term itself is justified by the large probability of the (0-1)-cousin. More formally, we can observe that generally the relation holds in Figure 4.4 that:

$$P(C^{11}_{kl}|E^C_{ij}) > P(C^{12}_{kl}|E^C_{ij}) > P(C^{22}_{kl}|E^C_{ij}) > P(C^{mn}_{kl}|E^C_{ij})\forall m > 2.$$

.

As an example of the explicit construction of the potential hypernym candidate set, in the *continental* example, from the 26 individual pieces of evidence over words we construct a set of 99 unique synsets that we will consider as possible hypernyms. This set includes the two senses of the noun *airline*, the eleven senses of the noun *carrier*, and so forth.

Next, we iterate through each of the possible hypernym synsets c_l under which we might add the new word w_i; for each synset c_l we compute the change in taxonomy score resulting from adding the implied relations $\mathbf{I}(H^1_{il})$ required by the taxonomic constraints of $\mathbf{T}$. Since typically our set of all evidence involving w_i will be much smaller than the set of possible relations in $\mathbf{I}(H^1_{il})$, we may efficiently check whether, for each sense $s \in \text{senses}(w)$, for all words where we have some evidence E^R_{iw}, whether s participates in some relation with w_i in the set of implied relations $\mathbf{I}(H^1_{il})$.[1] If there is more than one sense $s \in \text{senses}(w)$, we add to $\mathbf{I}(H^1_{il})$ the single relationship $R_{is'}$

[1]Checking whether or not $R_{is} \in \mathbf{I}(H^1_{il})$ may be efficiently computed by checking whether s is in the hypernym ancestors of c_l or if it shares a least common subsumer with c_l within 7 steps.

that maximizes the taxonomy likelihood, that is:

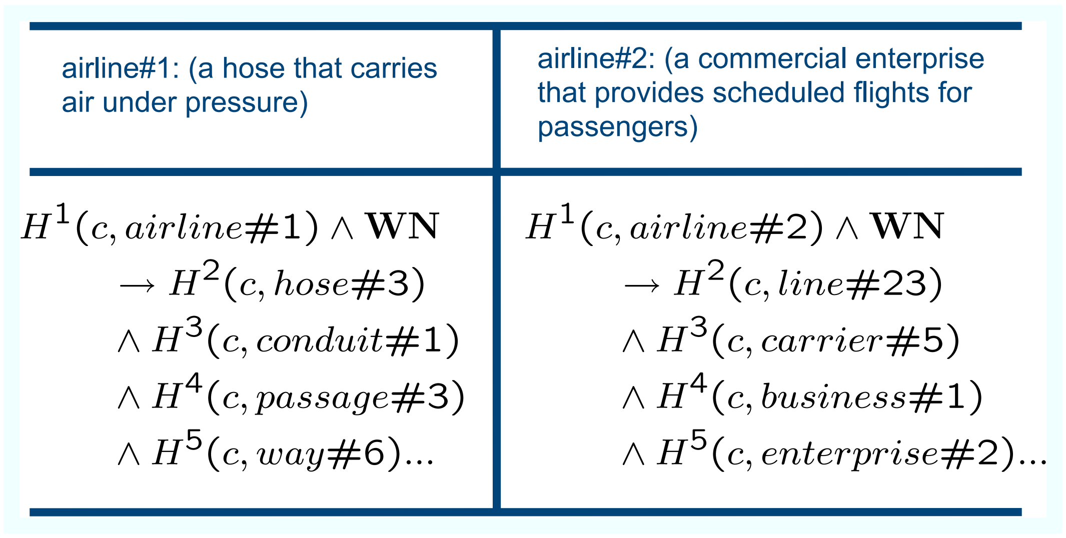
$$s' = \arg \max_{s \in \text{senses}(w)} \Delta_{\mathbf{T}}(R_{is})$$

4.5.1 Hypernym sense disambiguation

A major strength of our model is its ability to correctly choose the sense of a hypernym to which to add a novel hyponym, despite collecting evidence over untagged word pairs. Word sense disambiguation is an implicit side-effect of our algorithm; since our algorithm chooses to add the single link which, with its implied links, yields the most likely taxonomy, and since each distinct synset in WordNet has a different immediate neighborhood of relations, our algorithm simply disambiguates each node based on its surrounding structural information.

As an example of sense disambiguation in practice, consider our example of *continental*. Suppose we are iterating through each of the 99 possible synsets under which we might add *continental* as a hyponym, and we come to the synset *airline#n#2* in WordNet 2.1, i.e., "a commercial organization serving as a common carrier." In our consideration of the implied relations of adding *continental* to this sense of *airline*, we must consider each of the relations shown in the right hand side of Figure 4.5. In this case we will iterate through each piece of hypernym and coordinate evidence; we find that the relation H(*continental, carrier*) is satisfied with high probability for the specific synset *carrier#n#5*, the grandparent of *airline#n#2*; thus the factor $\Delta_{\mathbf{T}}(H^3($*continental, carrier#n#5*$))$ is included in the factor of the set of implied relations $\Delta_{\mathbf{T}}(\mathbf{I}(H^1($*continental, airline#n#2*$)))$.

Suppose we instead evaluate the *first* synset of *airline*, i.e., *airline#n#1*, with the gloss "a hose that carries air under pressure". Again we must consider each of the implied relations, as shown in the left-hand side of Figure 4.5. For this synset none of the other 20 relationships directly implied by hypernym evidence or the 5 relationships implied by the coordinate evidence are implied by adding the single link $H^1($*continental,airline#n#1*$)$; thus the resulting change in the set of implied links given by the correct "carrier" sense of airline is much higher than that of the "hose" sense. In fact, the correct sense is the most strongly predicted of all the 99 considered



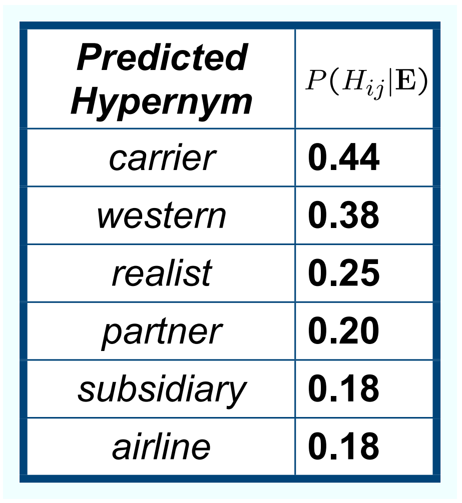
airline#1: (a hose that carries air under pressure)	airline#2: (a commercial enterprise that provides scheduled flights for passengers)
$H^1(c, airline\#1) \wedge \mathbf{WN} \rightarrow H^2(c, hose\#3) \wedge H^3(c, conduit\#1) \wedge H^4(c, passage\#3) \wedge H^5(c, way\#6)\ldots$	$H^1(c, airline\#2) \wedge \mathbf{WN} \rightarrow H^2(c, line\#23) \wedge H^3(c, carrier\#5) \wedge H^4(c, business\#1) \wedge H^5(c, enterprise\#2)\ldots$

Figure 4.5: Example of enforcing taxonomic constraints for adding *continental* to two different senses of *airline*

hypernym links for *continental*, and in this experiment $H^1(\textit{continental, airline\#n\#2})$ is link #18,736 added to the taxonomy by our algorithm.

4.5.2 Example inference

In order to see a representation of the inference process, consider the following figures. Here we are attempting to determine the appropriate concept to pick as the hypernym of a novel concept node for the token *continental*. At this point, WordNet contains no concept for the noun term *continental*. In order to add a new term, we must determine which concept in WordNet we ought to choose as the appropriate hypernym. As an example baseline, consider the algorithm which simply chooses the first sense of most probable single link predicted by the underlying relation classifiers. As shown in Figure 4.6, the most probable single link predicted by the hypernym classifier is that *continental* is a kind of *carrier*. While this inference is correct for one sense of *carrier* (in fact, the fifth sense of *carrier* listed in WordNet), it is only true for that one sense – and there are 11 different senses of *carrier* to choose from. Thus the naive algorithm would fail to choose to correct sense; instead, it chooses the first sense of *carrier*, a kind of a person. The gloss for that sense reads: “someone whose employment



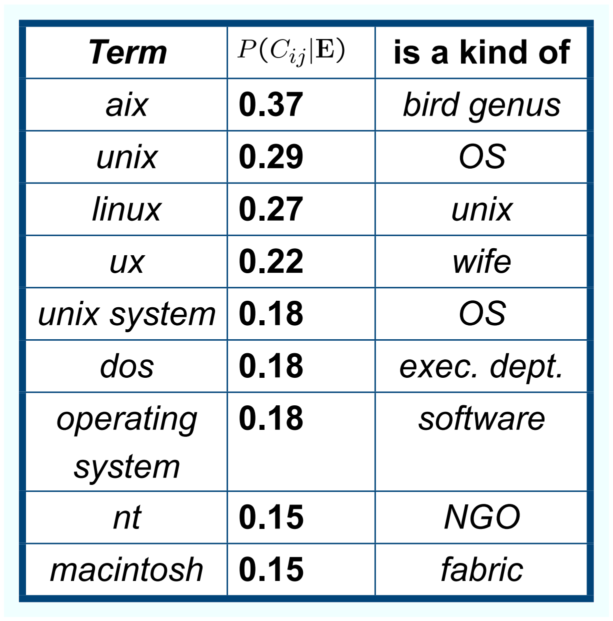
Predicted Hypernym	$P(H_{ij}\|\mathbf{E})$
carrier	**0.44**
western	**0.38**
realist	**0.25**
partner	**0.20**
subsidiary	**0.18**
airline	**0.18**

Figure 4.6: Example of hypernym predictions for the token *continental*

involves carrying something".

Suppose we consider the effect of taking into account the strongest prediction in Figure 4.6, that *continental* is a kind of a *carrier*. Because our relation classifier does not give us any explicit information about which word sense of carrier the relation might refer to, our model correspondingly increments the prediction of each of the 11 different possible senses of *carrier* as the possible hypernym equally. The different senses of the word *carrier* are not the only potential hypernyms whose probability are increased; estimates that *continental* is a kind of *airline*, *railway*, *bus line*, *Typhoid Mary*, *luggage carrier*, and *ski rack* all increase as well (along with many other concepts), as these are each extended hyponyms of the various senses of *carrier*. Thus, if continental is chosen as, say, a kind of *bus line*, this would explain the hypernym prediction of *continental* as a *carrier*. In particular the concept of *airline#n#2* increases slightly due to this evidence, as *airline#n#2* is a hyponym of *carrier#n#5*.

After taking into account the next piece of evidence from Figure 4.6—that *continental* is a kind of a *western*—we see little change in the overall estimates. Here WordNet contains two senses for *western*—one as a kind of *film*, and one as a kind of a *sandwich*. These senses do not interact with the senses of carrier until very high up in the hierarchy (as a *physical entity*), and thus the relative probabilities for the different senses of *carrier* or the other senses are unaffected.



Term	$P(C_{ij}\|\mathbf{E})$	**is a kind of**
aix	**0.37**	*bird genus*
unix	**0.29**	*OS*
linux	**0.27**	*unix*
ux	**0.22**	*wife*
unix system	**0.18**	*OS*
dos	**0.18**	*exec. dept.*
operating system	**0.18**	*software*
nt	**0.15**	*NGO*
macintosh	**0.15**	*fabric*

Figure 4.7: Example of coordinate and corresponding hypernym predictions for the token *SunOS*

After considering the sixth piece of evidence, however, the landscape changes; here we see evidence for *continental* as a kind of *airline*. There are two senses of *airline* in WordNet – one as a kind of *hose*, and the other as a kind of *carrier*. Because there is a strong interaction between the second sense of *airline* and the fifth sense of *carrier*, the hypothesis that *continental* is a kind of *airline#n#2*is now the most probable interpretation of the evidence. We find that after these six relations are considered, the prediction of *continental* as a kind of *airline#n#2* remains as the highest prediction; thus, out of the more than 80,000 possible concepts which our system might choose as a hypernym, this inference process succeeds in finding the most likely hypernym for this novel concept.

Similarly we may consider the same inference process for the novel term *SunOS*; while WordNet does not contain a concept for this operating system, it does contain *Unix*, *Linux*, *DOS* and other operating systems. Here in Figure 4.7 we display the most highly predicted coordinate terms using our coordinate classifier, along with their direct hypernyms known in WordNet.

Here we see that our coordinate classifier predicts several strong associations with

SunOS, including AIX, UNIX, Linux, and so forth. Unfortunately, WordNet only knows a fraction of these as operating systems; for instance, WordNet only contains the term *AIX* as a kind of *bird genus*, and the term *UX* as a kind of *wife*. Thus, again, the naive algorithm of selecting the first sense of the most likely link will fail again here: in this case, selecting *AIX* as the most likely coordinate, the naive algorithm would assign the hypernym *bird genus* to *SunOS* as well.

Consider our estimate of the taxonomy likelihood after considering only the first prediction for *SunOS*, that is, the prediction that *SunOS* is similar to *AIX*, and thus a kind of a *bird genus*. Since there is only one sense of *AIX* listed in WordNet, the probability for the hypernym *bird genus* and nearby concepts is increased by taking into account this prediction.

After considering the next piece of evidence from Figure 4.7, that *SunOS* is similar to *UNIX*, this gives the largest probability increase to *SunOS* sharing a hypernym with *UNIX*, namely stating that *SunOS* is a kind of an *operating system*. However, the probability that *SunOS* is a kind of a *UNIX* also increases, though not as much.

After considering six predictions, the distribution of hypernym probabilities begins to stabilize. Here the sixth piece of evidence is that *SunOS* is similar to *DOS*. In this case there are actualy two relevant senses of *DOS* within WordNet; first, the sense that *DOS* is the *Department of State*, that is, an executive department. The second listed sense of *DOS* is as the *Disk Operating System*, an operating system. Thus taking into account this evidence increases the likelihood for both *SunOS* as a kind of an *executive department* and as a kind of *operating system*; however, no other evidence supports the interpretation of *SunOS* as an *executive department*, whereas the interpretations of evidence of similarity to *UNIX*, *LINUX*, and *UNIX operating system* all support the interpretation that *SunOS* is an *operating system*. Thus after taking into account this evidence, *operating system* is the most likely hypernym by far; this interpretation remains the most likely after taking into account all further evidence.

In Figure 4.8 we give a brief listing of some of the other novel hyponyms inferred by our algorithm. While some of the inferred hypernyms are fairly specific named entities, such as new companies and markup languages, many of the inferences are

Hypernym	Hyponyms in WordNet	Not in WN, but added by our algorithm
hormone	insulin progesterone	leptin pregnenolone
quality	combustibility navigability arability	affordability reusability extensibility
markup language	HTML SGML	XML XHTML
company	"East India Company"	Microsoft IBM

Figure 4.8: Example novel hypernyms inferred in SWN

actually quite general concepts, such as novel "qualities" which are, for one reason or another, simply not listed in WordNet, including *affordability*, *reusability*, and *extensibility*.

4.6 Evaluation

In order to evaluate our framework for taxonomy induction, we have applied hyponym acquisition to construct several distinct taxonomies, starting with the base of WordNet 2.1 and only adding novel noun hyponyms. Further, we have constructed taxonomies using a baseline algorithm, which uses the identical hypernym and coordinate classifiers used in our joint algorithm, but which does not combine the evidence of the classifiers.

Here we describe our evaluation methodology, analyze the fine-grained precision and disambiguation precision of our algorithm compared to the baseline, and compare the coarse-grained precision of our links (motivated by categories defined by

1 Tops	8 communication	15 object	22 relation
2 act	9 event	16 person	23 shape
3 animal	10 feeling	17 phenomenon	24 state
4 artifact	11 food	18 plant	25 substance
5 attribute	12 group	19 possession	26 time
6 body	13 location	20 process	
7 cognition	14 motive	21 quantity	

Table 4.1: The 26 WordNet supersenses

the WordNet *supersenses*) against the baseline algorithm and against an “oracle” for named entity recognition. Finally, in section 4.6.5 we evaluate the taxonomies inferred by our algorithm directly against the WordNet 2.1 taxonomy; we perform this evaluation by testing each taxonomy on a set of human judgments of hypernym and non-hypernym noun pairs sampled from newswire text.

4.6.1 Methodology

We evaluate the quality of our acquired hyponyms by direct judgment. In four separate annotation sessions, two judges labeled {50,100,100,100} samples uniformly generated from the first {100,1000,10000,20000} single links added by our algorithm, respectively.

For the direct measure of fine-grained precision, we simply ask for each link $H(X, Y)$ added by the system, is X a Y? In addition to the fine-grained precision, we give a coarse-grained evaluation, inspired by the idea of supersense-tagging in [31]. The 26 supersenses used in WordNet 2.1 are listed in Table 4.1; we label a hyponym link as correct in the coarse-grained evaluation if the novel hyponym is placed under the appropriate supersense. This evaluation task is similar to a fine-grained Named Entity Recognition [51] task with 26 categories; for example, if our algorithm mistakenly inserts a novel non-capital city under the hyponym *state capital*, it will inherit the correct supersense *location*.

Finally, we evaluate the ability of our algorithm to correctly choose the appropriate sense of the hypernym under which a novel hyponym is being added. Our labelers categorize each candidate sense-disambiguated hypernym synset suggested by our algorithm into the following categories:

c_1: Correct sense-disambiguated hypernym.

c_2: Correct hypernym word, but incorrect sense of that word.

c_3: Incorrect hypernym, but correct supersense.

c_4: Any other relation is considered incorrect.

A single hyponym/hypernym pair is allowed to be simultaneously labeled 2 and 3.

4.6.2 Fine-grained evaluation

Table 4.2 displays the results of our evaluation of fine-grained precision for the baseline non-joint algorithm (**Base**) and our joint algorithm (**Joint**), as well as the relative error reduction (**ER**) of our algorithm over the baseline. When we report accuracy use the minimum of the two judges' scores. Here we define fine-grained precision as $c_1/total$. We see that our joint algorithm strongly outperforms the baseline, and has high precision for predicting novel hyponyms up to 10,000 links.

4.6.3 Hypernym sense disambiguation

Also in Table 4.2 we compare the sense disambiguation precision of our algorithm and the baseline. Here we measure the precision of sense-disambiguation among all examples where each algorithm found a correct hyponym word; our calculation for disambiguation precision is $c_1/(c_1 + c_2)$. Again our joint algorithm outperforms the baseline algorithm at all levels of recall. Interestingly the baseline disambiguation precision improves with higher recall; this may be attributed to the observation that the highest-confidence hypernyms predicted by individual classifiers are likely to be polysemous, whereas hypernyms of lower confidence are more frequently monosemous (and thus trivial to disambiguate).

	Fine-grained Pre.			**Disambiguation Pre.**		
#Links	Base	Joint	ER	Base	Joint	ER
100	0.60	1.00	100%	0.86	1.00	100%
1000	0.52	0.93	85%	0.84	1.00	100%
10000	0.46	0.84	70%	0.90	1.00	100%
20000	0.46	0.68	41%	0.94	0.98	68%

Table 4.2: Fine-grained and disambiguation precision and error reduction (ER) for hyponym acquisition

# Links	NER Oracle	Base	Joint	ER vs. NER	ER vs. Base
100	1.00	0.72	1.00	0%	100%
1000	0.69	0.68	0.99	97%	85%
10000	0.45	0.69	0.96	93%	70%
20000	0.54	0.69	0.92	83%	41%

Table 4.3: Coarse-grained precision and error reduction (ER) vs. non-joint baseline and named entity recogzier (NER) oracle

4.6.4 Coarse-grained evaluation

We compute coarse-grained precision as $(c_1 + c_3)/total$. Inferring the correct coarse-grained supersense of a novel hyponym can be viewed as a fine-grained (26-category) Named Entity Recognition task; our algorithm for taxonomy induction can thus be viewed as performing high-accuracy fine-grained NER. Here we compare against both the baseline non-joint algorithm as well as an "oracle" algorithm for Named Entity Recognition, which perfectly classifies the supersense of all nouns that fall under the four supersenses $\{person, group, location, quantity\}$, but works only for those supersenses. Table 4.3 shows the results of this coarse-grained evaluation. We see that the baseline non-joint algorithm has higher precision than the NER oracle as 10,000 and 20,000 links; however, both are significantly outperformed by our joint algorithm, which maintains high coarse-grained precision (92%) even at 20,000 links.

# Links	Baseline	Joint	ER
100	0.86	1.00	100%
1000	0.84	1.00	100%
10000	0.90	1.00	100%
20000	0.94	0.98	67%

Table 4.4: Sense-disambiguation precision and error reduction of sense-disambiguated hyponym acquisition

	WN	+10K	+20K	+30K	+40K
PRE	0.524	0.524	0.574	**0.583**	0.571
REC	0.165	0.165	0.203	**0.211**	0.211
F	0.251	0.251	0.300	**0.309**	0.307

Table 4.5: Taxonomy hypernym classification vs. WordNet 2.1 on hand-labeled test-set

4.6.5 Comparison of inferred taxonomies and WordNet

For our final evaluation we compare our learned taxonomies directly against the currently existing hypernym links in WordNet 2.1. In order to compare taxonomies we use a hand-labeled test set of over 5,000 noun pairs, randomly-sampled from newswire corpora, as described in Section 2.4. We measured the performance of both our inferred taxonomies and WordNet against this test set.[2] The performance and comparison of the best WordNet classifier vs. our taxonomies is given in Table 4.5. Our best-performing inferred taxonomy on this test set is achieved after adding 30,000 novel hyponyms, achieving a 23% relative improvement in F-score over the WN2.1 classifier.

[2]We found that the WordNet 2.1 model achieving the highest F-score used only the first sense of each hyponym, and allowed a maximum distance of 4 edges between each hyponym and its hypernym.

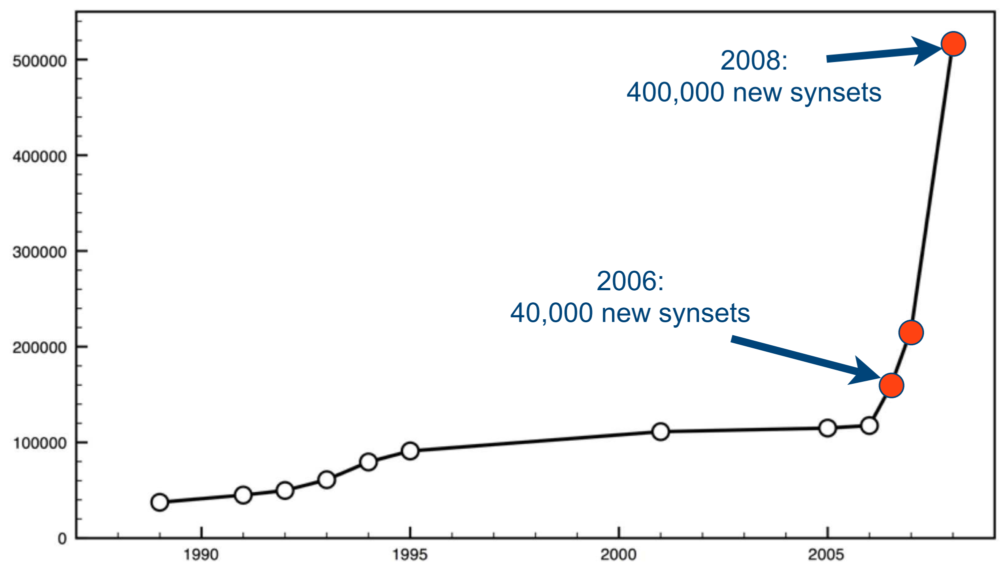


Figure 4.9: Growth of the Stanford Wordnet

4.7 Discussion

In this chapter we have demonstrated the application of our framework for taxonomy induction to the problem of automatically augmenting the WordNet taxonomy. We have presented a local search algorithm using an operation for adding novel concepts, and shown that this algorithm is effective for creating augmented taxonomies. We have shown that our taxonomy induction framework significantly outperforms a baseline algorithm for taxonomy augmentation, and that our augmented taxonomies significantly outperform WordNet in a hyponym recognition task in newswire text.

As shown in Figure 4.9, since the beginning of the Stanford Wordnet in 2006 we have expanded the total size of the noun hypernym hierarchy in WordNet by more than 400%; the first release of the Stanford Wordnet was in 2006, with 40,000 novel noun synsets; in 2008 we released a version with a total of 400,000 new noun synsets, also available at the Stanford Wordnet homepage at

`http://ai.stanford.edu/~rion/swn`.

Chapter 5

Creating Sense-clustered Taxonomies

5.1 Introduction

A major benefit and use case of taxonomies such as WordNet are as repositories for the different possible senses or meanings of lexical items. Defining a discrete inventory of senses for a word is extremely difficult [71, 58, 104]. Perhaps the greatest obstacle is the dynamic nature of sense definition: the correct granularity for word senses depends on the application. For language learners, a fine-grained set of word senses may help in learning subtle distinctions, while coarsely-defined senses are probably more useful in NLP tasks like information retrieval [57], query expansion [97], and word sense disambiguation (WSD) [121, 104].

Lexical resources such as WordNet [50] use extremely fine-grained notions of word sense, which carefully capture even minor distinctions between different possible word senses (e.g., the 8 noun senses of *bass* shown in Figure 5.1). These very similar senses can be clustered together, however, in order to produce more coarse-grained inventories. Producing sense-clustered inventories of arbitrary sense granularity is thus crucial for tasks which depend on lexical resources like WordNet, and is also important for the task of automatically constructing new WordNet-like taxonomies. A solution to this problem must also deal with the constraints of the WordNet taxonomy

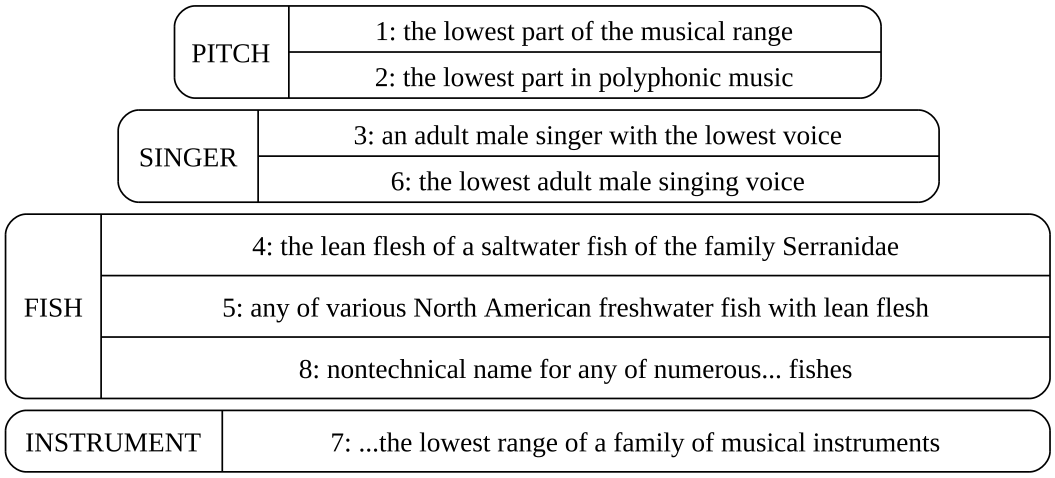

PITCH
1: the lowest part of the musical range
2: the lowest part in polyphonic music
SINGER
3: an adult male singer with the lowest voice
6: the lowest adult male singing voice
FISH
4: the lean flesh of a saltwater fish of the family Serranidae
5: any of various North American freshwater fish with lean flesh
8: nontechnical name for any of numerous... fishes
INSTRUMENT
7: ...the lowest range of a family of musical instruments

Figure 5.1: Sense clusters for the noun *bass*; the eight WordNet senses as clustered into four groups in the Senseval-2 coarse-grained evaluation data

itself; for example when clustering two senses, we need to consider the transitive effects of merging synsets.

The state of the art in sense clustering is insufficient to meet these needs. Current sense clustering algorithms are generally unsupervised, each relying on a different set of useful features or hand-built rules. But hand-written rules have little flexibility to produce clusterings of different granularities, and previously proposed methods offer little in the direction of intelligently combining and weighting the many proposed features.

In response to these challenges, we propose a new algorithm for clustering large-scale sense hierarchies like WordNet. Our algorithm is based on a supervised classifier that learns to make graduated judgments corresponding to the estimated probability that each particular sense pair should be merged. This classifier is trained on gold standard sense clustering judgments using a diverse feature space. We are able to use the outputs of our classifier to produce a ranked list of sense merge judgments by merge probability, and from this create sense-clustered inventories of arbitrary sense granularity.[1]

[1]We have made sense-clustered Wordnets using the algorithms discussed in this chapter available for download at http://ai.stanford.edu/∼rion/swn.

A wide number of manual and automatic techniques have been proposed for clustering sense inventories and mapping between sense inventories of different granularities. Much work has gone into methods for measuring synset similarity; early work in this direction includes [46], which attempted to discover sense similarities between dictionary senses. A variety of synset similarity measures based on properties of WordNet itself have been proposed; nine such measures are discussed in [112], including gloss-based heuristics [79, 7], information-content based measures [120, 82, 66], and others. Other approaches have used specific cues from WordNet structure to inform the construction of semantic rules; for example, [114] suggest clustering two senses based on a wide variety of structural cues from WordNet, including if they are *twins* (if two synsets share more than one word in their synonym list) or if they represent an example of *autohyponymy* (if one sense is the direct descendant of the other). [93] implements six semantic rules, using *twin* and *autohyponym* features, in addition to other WordNet-structure-based rules such as whether two synsets share a *pertainym*, *antonym*, or are clustered together in the same *verb group*.

A large body of work has attempted to capture corpus-based estimates of word similarity [113, 81]; however, the lack of large sense-tagged corpora prevent most such techniques from being used effectively to compare different senses of the same word. Some corpus-based attempts that are capable of estimating similarity between word senses include the *topic signatures* method; here, [3] collect contexts for a polysemous word based either on sense-tagged corpora or by using a weighted agglomeration of contexts of a polysemous word's monosemous relatives (i.e., single-sense synsets related by hypernym, hyponym, or other relations) from some large untagged corpus. Other corpus-based techniques developed specifically for sense clustering include [88], which uses a combination of word-to-word distributional similarity combined with the JCN WordNet-based similarity measure, and work by [30] in finding co-occurrences of senses within documents in sense-tagged corpora. Other attempts have exploited disagreements between WSD systems [3] or between human labelers [28] to create synset similarity measures; while promising, these techniques are severely limited by the performance of the WSD systems or the amount of available labeled data.

Some approaches to clustering have made use of regular patterns of polysemy among words. [114] uses the Cousin relation defined in WordNet 1.5 to cluster hyponyms of categorically related noun synsets, e.g., “container/quantity” (e.g., for clustering senses of “cup” or “barrel”) or ”organization/construction” (e.g., for the building and institution senses of “hospital” or “school”); other approaches based on systematic polysemy include the hand-constructed CORELEX database [17], and automatic attempts to extract patterns of systematic polysemy based on minimal description length principles [145].

Another family of approaches has been to use either manually-annotated or automatically constructed mappings to coarser-grained sense inventories; an attempt at providing coarse-grained sense distinctions for the Senseval-1 exercise included a mapping between WordNet and the Hector lexicon [104]. Other attempts in this vein include mappings between WordNet and PropBank [103] and mappings to Levin classes [80, 104]. [99] presents an automatic approach for mapping between sense inventories; here similarities in gloss definition and structured relations between the two sense inventories are exploited in order to map between WordNet senses and distinctions made within the coarser-grained Oxford English Dictionary. Other work has attempted to exploit translational equivalences of WordNet senses in other languages, for example using foreign language WordNet interlingual indexes [57, 30].

In Section 5.2 we discuss the gold standard datasets that we use in our work; in Section 5.2 we introduce our battery of features and explain our sense merge classifier; in Section 5.4 we show how to extend our sense-merging model to cluster full taxonomies like WordNet; in Section 5.5 we evaluate our classifier against thirteen previously proposed methods for sense clustering, and in 5.6 we summarize and discuss our results.

5.2 Gold standard sense clustering data

Our approach for learning how to merge senses relies upon the availability of labeled judgments of sense relatedness. In this work we focus on two datasets of hand-labeled sense groupings for WordNet: first, a dataset of sense groupings over nouns, verbs, and adjectives provided as part of the Senseval-2 English lexical sample WSD task

Senseval-2

POS	Total Pairs	Merged Pairs	Proportion
Nouns	16403	2593	0.1581
Verbs	30688	3373	0.1099
Adjectives	8368	2209	0.2640

OntoNotes

POS	Total Pairs	Merged Pairs	Proportion
Nouns	3552	347	0.0977
Verbs	4663	1225	0.2627

Table 5.1: Gold standard datasets for sense merging; only sense pairs that share a word in common are included; proportion refers to the fraction of synsets sharing a word that have been merged

[72], and second, a corpus-driven mapping of nouns and verbs in WordNet 2.1 to the Omega Ontology [115], produced as part of the OntoNotes project [64]. Each of these datasets provide a clustering of the senses of WordNet similar to that displayed in Figure 5.1. That is, for a specific set of words, all the senses of those words are presented in a single, fixed, coarse-grained clustering.

A wide variety of semantic and syntactic criteria were used to produce the Senseval-2 groupings [103, 104]; this data covers all senses of 411 nouns, 519 verbs, and 257 adjectives, and has been used as gold standard sense clustering data in previous work [3, 88][2]. The number of judgments within this data (after mapping to WordNet 2.1) is displayed in Table 5.1.

Due to a lack of interannotator agreement data for this dataset, [88] performed an annotation study using three labelers on a 20-noun subset of the Senseval-2 groupings; the three labelers were given the task of deciding whether the 351 potentially-related sense pairs were "Related", "Unrelated", or "Don't Know"[3]. In this task the pairwise interannotator F-scores were (0.4874, 0.5454, 0.7926), for an average F-score of 0.6084.

[2]In order to facilitate future work in this area, we have made cleaned versions of these groupings available at http://ai.stanford.edu/∼rion/swn along with a "diff" with the original files.

[3]This gold standard data has been made available at
`ftp://ftp.informatics.susx.ac.uk/pub/users/dianam/relateGS/`.

POS	Overlap	ON-True		ON-False		F-Score
		S-T	S-F	S-T	S-F	
Nouns	2116	121	55	181	1759	0.5063
Verbs	3297	351	503	179	2264	0.5072

Table 5.2: Agreement data for gold standard datasets

The OntoNotes dataset[4] covers a smaller set of nouns and verbs, but it has been created with a more rigorous corpus-based iterative annotation process. For each of the nouns and verbs in question, a 50-sentence sample of instances is annotated using a preliminary set of sense distinctions; if the word sense interannotator agreement for the sample is less than 90%, then the sense distinctions are revised and the sample is re-annotated, and so forth, until an interannotator agreement of at least 90% is reached.

An interesting point to note on the different gold standard datasets is the different sense resolutions at which senses are clustered together; for example, Figure 5.2 shows the different clusterings used by the Senseval-2 and the OntoNotes judges for the verb *oppose*. Here we see that Senseval-2 groups "passive" and "active" senses of oppose, whereas OntoNotes does not distinguish those; whereas OntoNotes *does* distinguish between two other senses of *oppose*. This difference in sense resolution in the gold standard datasets lends credence to the notion of having an automatic method for determining variable sense clusterings of various resolutions. Here is one case where rational human judges came to different decision about which resolution certain senses should be clustered to; clearly there are multiple reasonable decision points, and a more nuanced representation capable of capturing multiple decision points is desirable.

We construct a combined gold standard set from these Senseval-2 and OntoNotes groupings, removing disagreements. The overlap and agreement/disagreement data between the two groupings is given in Table 5.2; here, for example, the column with

[4]The OntoNotes groupings were used as the coarse-grained sense inventory for the English lexical sample task (Task 17) of SemEval-2007, and are now available through the Linguistic Data Consortium.

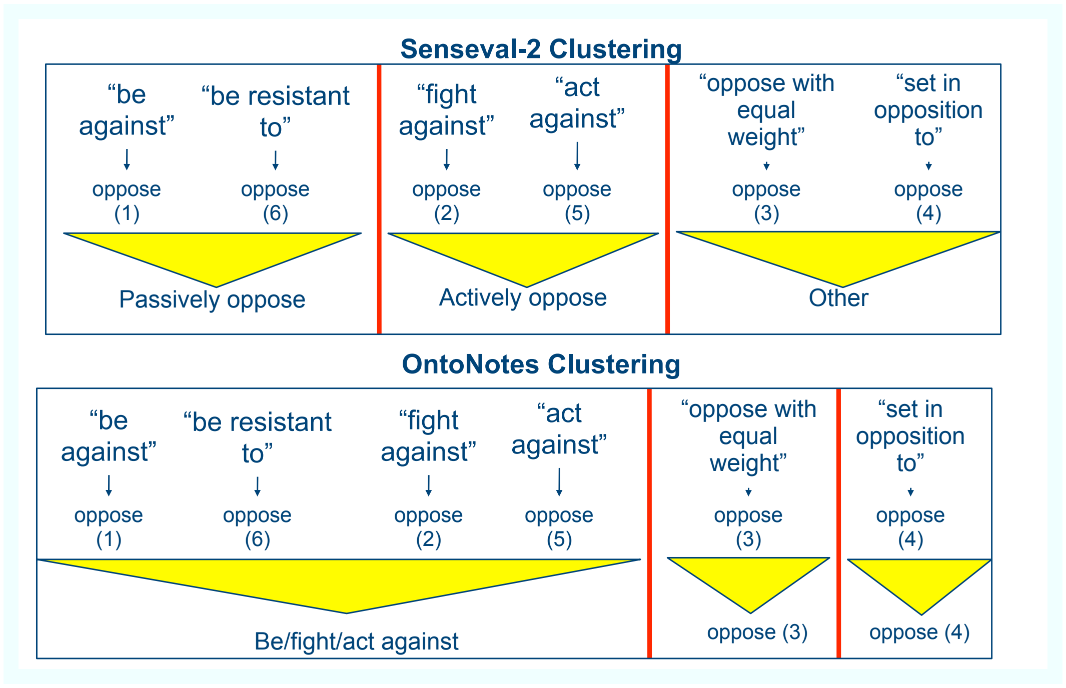


Figure 5.2: Examples of different sense clustering resolutions in the gold standard datasets for the verb *oppose*

ON-True and **S-F** indicates the count of senses that OntoNotes judged as positive examples of sense merging, but that Senseval-2 data did not merge. We also calculate the F-score achieved by considering only one of the datasets as a gold standard, and computing precision and recall for the other. Since the two datasets were created independently, with different annotation guidelines, we cannot consider this as a valid estimate of interannotator agreement; nonetheless the F-score for the two datasets on the overlapping set of sense judgments (50.6% for nouns and 50.7% for verbs) is roughly in the same range as those observed in [88].

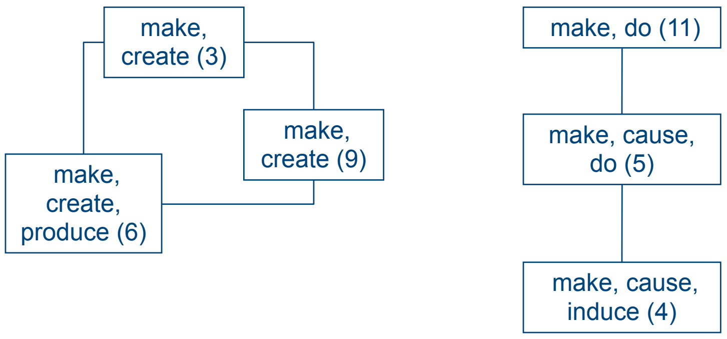


Figure 5.3: Examples of twin features for the verb *make*

5.3 Learning to merge word senses

5.3.1 WordNet-based features

Here we describe the feature space we construct for classifying whether or not a pair of synsets should be merged; first, we employ a wide variety of linguistic features based on information derived from WordNet. We use eight similarity measures implemented within the WordNet::Similarity package[5], described in [112]; these include three measures derived from the paths between the synsets in WordNet: HSO [63], LCH [77], and WUP [155]; three measures based on information content: RES [120], LIN [82], and JCN [66]; the gloss-based Extended Lesk Measure LESK, [7], and finally the gloss vector similarity measure VECTOR [110]. We implement the TWIN feature [114], which counts the number of shared synonyms between two synsets; an example of the twin feature for the verb *make* is shown in Figure 5.3.

Additionally we produce pairwise features indicating whether two senses share an ANTONYM, PERTAINYM, or derivationally-related forms (DERIV). An example of the DERIV feature for the word *bank* is shown in Figure 5.4.

We also create the verb-specific features of whether two verb synsets are linked in a VerbGroup (indicating semantic similarity) or share a VerbFrame, indicating syntactic similarity. Also, we encode a generalized notion of siblinghood in the MN features, recording the distance of the synset pair's nearest least common subsumer

[5]We choose not to use the PATH measure due to its negligible difference from the LCH measure.

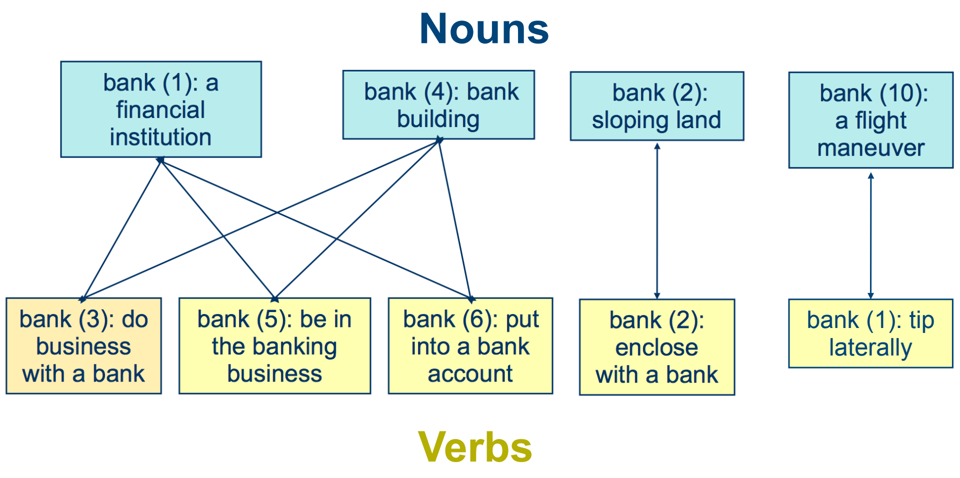

Nouns
bank (1): a financial institution
bank (4): bank building
bank (2): sloping land
bank (10): a flight maneuver
bank (3): do business with a bank
bank (5): be in the banking business
bank (6): put into a bank account
bank (2): enclose with a bank
bank (1): tip laterally
Verbs

Figure 5.4: Examples of shared derivational features for the noun and verb senses of *bank*

(i.e., closest shared hypernym) from the two synsets, and, separately, the maximum of those distances (MaxMN). These features represent a superset of the autohyponym feature previously used in other research; an example of autonhyponymy in the verb *make* is shown in Figure 5.5.

Previous attempts at categorizing systematic polysemy patterns within WordNet has resulted in the Cousin feature[6] These cousins include regular polysemy patterns like “container/quantity” (e.g., for “cup” or “barrel”) and ”organization/construction” (e.g., for the building and institution senses of “hospital” or “school”). For example, in Figure 5.6 we show an example of the regular polysemy patterns for *organization* and *construction* as represented in the “cousin.tops” file. We create binary features which indicate whether a synset pair belongs to hypernym ancestries indicated by one or more of these Cousin features, and the specific cousin pair(s) involved. Finally we create sense-specific features, including SenseCount, the total number of senses associated with the shared word between the two synsets with the highest number of senses, and SenseNum, the specific pairing of senses for the shared word with the highest number of senses. From this we might hope to learn whether

[6]This data is included in the WordNet 1.6 distribution as the “cousin.tops” file.

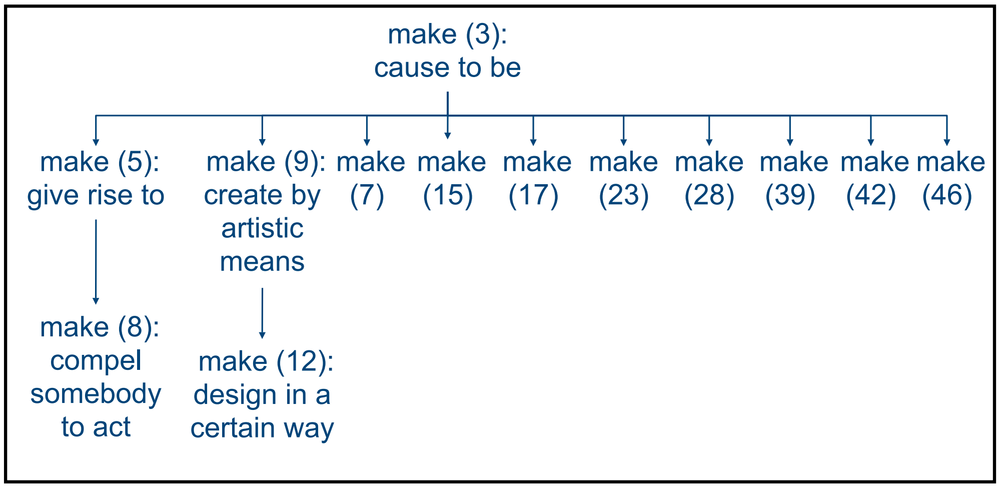


Figure 5.5: Examples of autohyponymy under the verb *make*

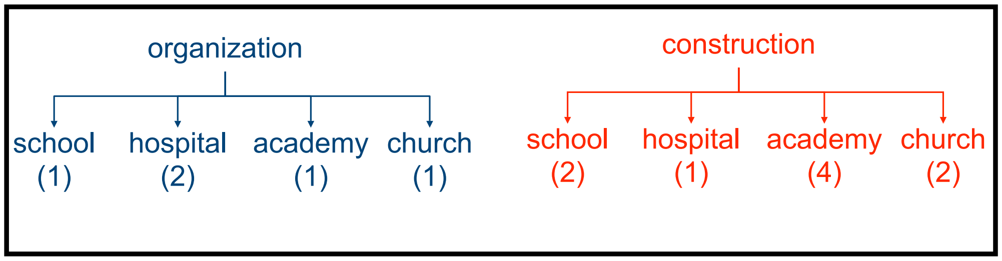


Figure 5.6: Examples of regular polysemy under *organization* and *construction*

the most frequent sense of a word has a higher chance of having infrequently-occurring derivative senses.

5.3.2 Features derived from corpora and other lexical resources

In addition to WordNet-based features, we use a number of features derived from corpora and other lexical resources. We use the publicly available topic signature data[7] described in [4], yielding representative contexts for all nominal synsets from

[7]The topic signature data is available for download at http://ixa.si.ehu.es/Ixa/resources/sensecorpus.

WordNet 1.6. These topic signatures were obtained by weighting the contexts of monosemous relatives of each noun synset (i.e., single-sense synsets related by hypernym, hyponym, or other relations); the text for these contexts were extracted from snippets using the Google search engine. We then create a sense similarity feature by taking a thresholded cosine similarity between pairs of topic signatures for these noun synsets.

Additionally, we use the WordNet domain dataset described in [85, 10]. This dataset contains one or more labels drawn from 164 hierarchically organized "domains" or "subject fields" for each noun, verb, and adjective synset in WordNet; we derive a set of binary features from this data, with a single feature indicating whether or not two synsets share a domain, and one indicator feature per pair of domains indicating respective membership of the sense pair within those domains.

Finally, we use as a feature the mappings produced in [99] of WordNet senses to Oxford English Dictionary senses. This OED dataset was used as the coarse-grained sense inventory in the Coarse-grained English all-words task of SemEval-2007[8]; we specify a single binary feature for each pair of synsets from this data; this feature is true if the words are clustered in the OED mapping, and false otherwise.

5.3.3 Classifier, training, and feature selection

We encode the above features into a sparse binary feature space of more than 20,000 features. Most of these features come from the WordNet domain dataset, since we represent the set of pairs of domains as individual binary features. Most of the other features we consider only have a single binary representation. Except where explicitly stated above, we construct binary features for each individual value of integer features above (e.g., for TWIN and MaxMN), and we convert real-valued features (i.e., WN-SIM and TOPICSIGNATURE) into binary features by binning them in a specific quantile range; we choose the number of bins using 10-fold cross validation on our training set.

[8] `http://lcl.di.uniroma1.it/coarse-grained-aw/index.html`

For each part of speech, we split the merged gold standard data into a part-of-speech-specific training set (70%) and a held-out test set (30%). For every synset pair we use the binary "merged" or "not-merged" labels to train a support vector machine classifier[9] [68] for each POS-specific training set. We perform feature selection and regularization parameter optimization using 10-fold cross-validation.

5.4 Clustering senses in WordNet

The previous section describes a classifier which predicts whether two synsets should be merged; we would like to use the pairwise judgments of this classifier to cluster the senses within a sense hierarchy. In this section we present the challenge implicit in applying sense merging to full taxonomies, and present our model for clustering within a taxonomy.

5.4.1 Challenges of clustering a sense taxonomy

The task of clustering a sense taxonomy poses certain challenges not present in the problem of clustering the senses of a word; in order to create a consistent clustering of a sense hierarchy an algorithm must consider the transitive effects of merging synsets. This problem is compounded in sense taxonomies like WordNet, where each synset may have additional structured relations, e.g., hypernym (IS-A) or holonym (is-part-of) links. In order to consistently merge two noun senses with different hypernym ancestries within WordNet, for example, an algorithm must decide whether to have the new sense inherit both hypernym ancestries, or whether to inherit only one, and if so it must decide which ancestry is more relevant for the merged sense.

Without strict checking, human labelers will likely find it difficult to label a sense inventory with transitively-consistent judgments. As an example, consider the Senseval-2 clusterings of the verbs *require* and *need*, as shown in Figure 5.7. In WN 2.1 *require* has four verb senses, of which the first has synonyms {*necessitate*, *ask*,

[9]We use the SVM^{perf} package, freely available for non-commercial use from http://svmlight.joachims.org; we use the default settings in v2.00, except for the regularization parameter (set in 10-fold cross-validation).

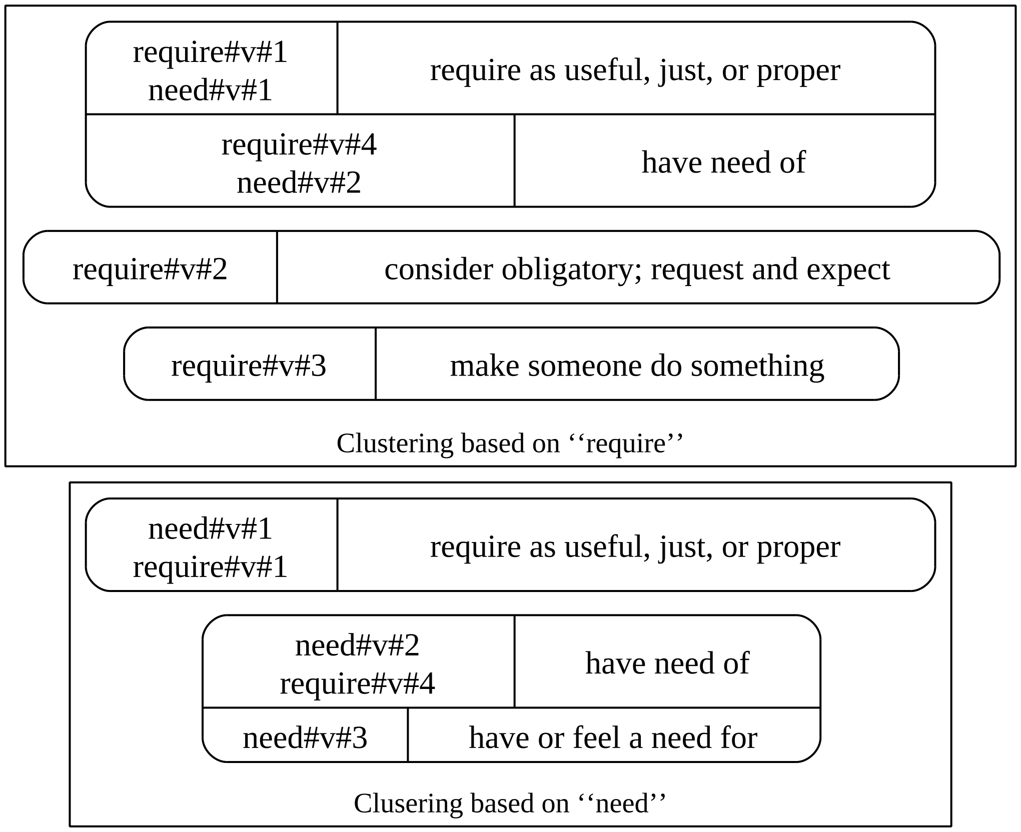

require#v#1
need#v#1
require as useful, just, or proper
require#v#4
need#v#2
have need of
require#v#2
consider obligatory; request and expect
require#v#3
make someone do something
Clustering based on ''require''
need#v#1
require#v#1
require as useful, just, or proper
need#v#2
require#v#4
have need of
need#v#3
have or feel a need for
Clusering based on ''need''

Figure 5.7: Inconsistent sense clusters for the verbs *require* and *need* from SENSEVAL-2 judgments

postulate, *need*, *take*, *involve*, *call for*, *demand*}, and gloss "require as useful, just, or proper"; and the fourth has synonyms {*want*, *need*}, and gloss "have need of."

Within the word *require*, the SENSEVAL-2 dataset clusters senses 1 and 4, leaving the rest unclustered. In order to make a consistent clustering with respect to the sense inventory, however, we must enforce the transitive closure by merging the synset corresponding to the first sense (*necessitate*, *ask*, *need* etc.), with the senses of *want* and *need* in the fourth sense. In particular, these two senses correspond to WordNet 2.1 senses *need#v#1* and *need#v#2*, respectively, which are **not** clustered according to the SENSEVAL-2 word-specific labeling for *need*—*need#v#1* is listed as a singleton (i.e., unclustered) sense, though *need#v#2* is clustered with *need#v#3*, "have or feel a need for."

While one might hope that such disagreements between sense clusterings are rare, we found 178 such transitive closure disagreements in the SENSEVAL-2 data. The ONTONOTES data is much cleaner in this respect, most likely due to the stricter annotation standard [64]; we found only one transitive closure disagreement in the OntoNotes data, specifically WordNet 2.1 synsets (*head#n#2*, *lead#n#7*: "be in charge of") and (*head#n#3*, *lead#v#4*: "travel in front of") are clustered under *head* but not under *lead*.

5.4.2 Sense clustering as maximizing taxonomy likelihood

As a solution to the challenges listed previously, we formulate the problem of clustering a sense hierarchy in the probabilistic framework for taxonomy induction described in Chapter 3. Whereas the use of the model in Chapter 4 considers the problem of combining multiple pairwise classifiers in order to predict whether and where a novel synset should be added to the hypernym hierarchy (using the ADD-CONCEPT operation), we build upon that work to use an additional MERGE-CONCEPTS operation, as described in Section 3.5. Additionally, we show that under certain assumptions a simple model for maximizing the resulting taxonomy likelihood is equivalent to performing agglomerative clustering using pairwise sense-merge probabilities as similarity scores.

First, we define the merge event M_{ij} as clustering together two synsets c_i and c_j within a taxonomy. We use the feature space and gold standard data described in Section 3 as evidence E^M_{ij} for predicting the merging of any pair of synsets that share a word. In order to produce pairwise probability estimates of the form $P(M_{ij}|E^M_{ij})$, we may convert the pairwise outputs of our SVM classifier into probabilistic judgments about whether two senses should be merged using the method described in [116].

As in Chapter 3 we consider a model for taxonomy likelihood, with our goal being to find the taxonomy $\hat{\mathbf{T}}$ that maximizes the conditional probability of our observations

$\mathbf{E}$ given the existing taxonomy $\mathbf{T}$, i.e., to find

$$\hat{\mathbf{T}} = \arg\max_{\mathbf{T}} P(\mathbf{E}|\mathbf{T}).$$

Our model for taxonomy induction is simpler than that used in Chapter 4 for two reasons: first, our evidence is already grounded to sense-disambiguated synsets rather than words, and second in the absence of additional sources of information we may cluster synsets without considering other probabilistic effects on the taxonomy.

We define the multiplicative change $\Delta_{\mathbf{T}}(M_{ij})$ to the conditional probability $P(\mathbf{E}|\mathbf{T})$, resulting in a new taxonomy $\mathbf{T}'$ after the merge action M_{ij} as follows:

$$\begin{aligned}\Delta_{\mathbf{T}}(M_{ij}) &= P(\mathbf{E}|\mathbf{T}')/P(\mathbf{E}|\mathbf{T}) \\ &= \frac{P(M_{ij}|E^M_{ij})P(E^M_{ij})}{P(\neg M_{ij}|E^M_{ij})P(E^M_{ij})} \cdot \frac{P(\neg M_{ij})}{P(M_{ij})} \\ &= k\left(\frac{P\left(M_{ij}|E^M_{ij}\right)}{1-P\left(M_{ij}|E^M_{ij}\right)}\right).\end{aligned}$$

Here k is the inverse odds of the prior on the merge event M_{ij}; by varying the prior distribution over the number of senses for a given word as a free parameter, potentially conditioned upon the initial number of senses within a taxonomy, this formulation can be used to produce sense-clustered semantic taxonomies of arbitrarily coarse sense granularity. We propose the corresponding best-first search algorithm for incremental synset merging:

```
WHILE $\max_{M_{ij} \notin \mathbf{T}} \Delta_{\mathbf{T}}(M_{ij}) > 1$
    $\mathbf{T} \leftarrow \arg\max_{M_{ij} \notin \mathbf{T}} \Delta_{\mathbf{T}}(M_{ij})$
```

The primary benefit of this formulation is that additional sources of evidence for sense relationships can directly be incorporated into the likelihood computation, for example distributional similarity for predicting coordinate terms, or lexico-syntactic patterns for predicting hypernyms. As shown in Chapter 4, robust prediction of relationships in semantic taxonomies can greatly benefit from the use of multiple interrelated sources of evidence.

One drawback to this model is that it requires an expensive recalculation of synset features after each pair of senses is merged. Rather than recomputing each feature and performing a new set of pairwise classifications after each merge operation, we instead approximate the pairwise similarity score between a given synset with a merged sense as the average of the similarity scores between the given synset and clustered sense's component synsets. Using this approximation, and without using additional sources of evidence, the resulting algorithm for incrementally maximizing the taxonomy likelihood is equivalent to average-link agglomerative clustering. This may be seen to be true by observing that after each successive merge event, only the pairwise probability judgments concerning the merged pair need to be updated; without taking into account additional sources of evidence, the likelihood of other relations or merge events remain unaffected.

Without exploiting additional hypernym or coordinate-term evidence, our algorithm does not distinguish between judgments about which hypernym ancestry or other structured relationships to keep or remove upon merging two synsets. In lieu of additional evidence, for our experiments we choose to keep a single hypernym ancestry, specifically the sense with the highest frequency in SemCor; we break frequency ties by choosing the base sense as the earliest-listed sense in WordNet. For other relationships (meronyms, entailments, pertainyms, etc.) we simply add each of these relationships to the new merged sense (except in the rare case where adding a relation would cause a cycle in acyclic relations like hypernymy or holonymy, in which case we omit it). Also, for the purpose of sense clustering we assume a zero sense similarity score between synsets with no intersecting words. Using this approach to taxonomy clustering we have produced several sense-clustered WordNets of varying sense granularity, available at:

`http://ai.stanford.edu/~rion/swn`

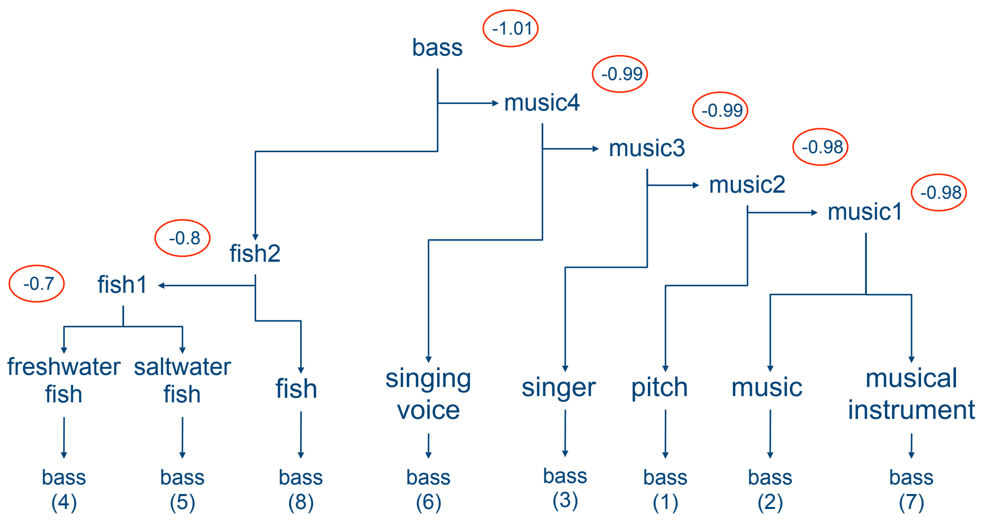


Figure 5.8: Full sense clustering dendrogram for the noun *bass*

For the example of clustering the senses of the noun *bass*, the resulting dendrogram is shown in Figure 5.8. Here we see that first the two senses of *bass* as a fish are clustered together; then, with lower probability, the musical senses are clustered.

As a second example, consider the senses of the verb *oppose*. In the gold standard datasets we observed that OntoNotes and Senseval-2 clustered this verb at two resolutions; we see that our automatic clustering is capable of capturing both of these resolutions, in addition to more. The full dendrogram captured by our algorithm is displayed in Figure 5.9.

5.5 Evaluation

We evaluate our classifier in a comparison with thirteen previously proposed similarity measures and automatic methods for sense clustering. We conduct a feature ablation study to explore the relevance of the different features in our system. Finally, we evaluate the sense-clustered taxonomies we create on the problem of providing improved coarse-grained sense distinctions for WSD evaluation.

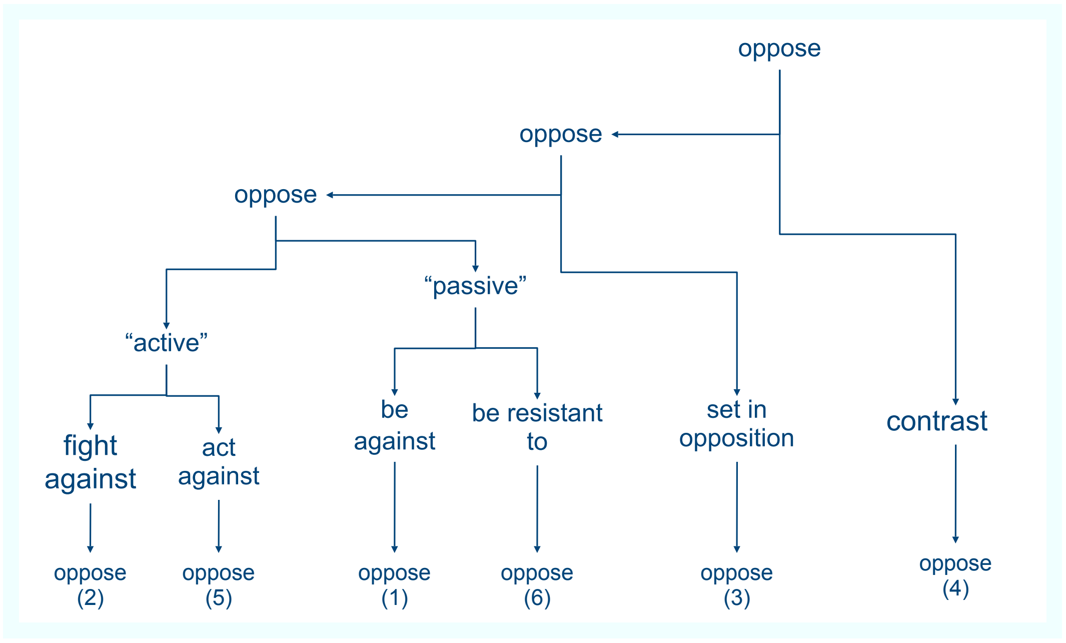


Figure 5.9: Full sense clustering dendrogram for the verb *oppose*

5.5.1 Evaluation of automatic sense merging

We evaluate our classifier on two held-out test sets; first, a 30% sample of the sense judgments from the merged gold standard dataset consisting of both the Senseval-2 and OntoNotes sense judgments; and, second, a test set consisting of only the OntoNotes subset of our first held-out test set. For comparison we implement thirteen of the methods discussed in Section 5.1. First, we evaluate each of the eight WordNet::Similarity measures individually. Next, we implement cosine similarity of topic signatures (TopSig) built from monosemous relatives [3], which provides a real-valued similarity score for noun synset pairs.

Additionally, we implement the two methods proposed in [114], namely using metonymy clusters (MetClust) and generalization clusters (GenClust) based on the Cousin relationship in WordNet. While [114] only considers four cousin pairs, we re-implement their method for general purpose sense clustering by using all 226 cousin pairs defined in WordNet 1.6, mapped to WordNet 2.1 synsets. These methods each provide a single clustering of noun synsets.

	Senseval-2 + OntoNotes			OntoNotes	
Method	Nouns	Verbs	Adj	Nouns	Verbs
SVM	**0.4228**	**0.4319**	**0.4727**	**0.3698**	**0.4545**
RES	0.3817	0.2703	—	0.2807	0.3156
WUP	0.3763	0.2782	—	0.3036	0.3451
LCH	0.3700	0.2440	—	0.2857	0.3396
OED	0.3310	0.2878	0.3712	0.2183	0.3962
Lesk	0.3174	0.2956	0.4323	0.2914	0.3774
HSO	0.3090	0.2784	0.4312	0.3025	0.3156
TopSig	0.3072	—	—	0.2581	—
VEC	0.2960	0.2315	0.4321	0.2454	0.3420
JCN	0.2818	0.2292	—	0.2222	0.3156
Lin	0.2759	0.2464	—	0.2056	0.3471
Baseline	0.2587	0.2072	0.4312	0.1488	0.3156
MiMo	0.0989	0.2142	0.0759	0.1833	0.2157
GenClust	0.0973	—	—	0.0264	—
MetClust	0.0876	—	—	0.0377	—

Table 5.3: F-score sense merging evaluation on hand-labeled testsets

Next, we implement the set of semantic rules described in [93] (MiMo); this algorithm for merging senses is based on 6 semantic rules, in effect using a subset of the Twin, MaxMN, Pertainym, Antonym, and VerbGroup features; in our implementation we set the parameter for when to cluster based on number of twins to $K = 2$; this results in a single clustering for each of nouns, verbs, and adjectives. Finally, we compare against the mapping from WordNet to the Oxford English Dictionary constructed in [99], equivalent to clustering based solely on the OED feature.

Considering merging senses as a binary classification task, Table 5.3 gives the F-score performance of our classifier vs. the thirteen other tested classifiers along with an uninformed "merge all synsets" baseline on our held-out gold standard test set. This table shows that our SVM classifier outperforms all implemented methods on the basis of F-score on both datasets for all parts of speech. In Figure 5.10 we give a precision/recall plot for noun sense merge judgments for the Senseval-2 + OntoNotes dataset. For sake of simplicity we plot only the two best measures

(RES and WUP) of the eight WordNet-based similarity measures; we see that our classifier, RES, and WUP each have higher precision all levels of recall compared to the other tested measures.

Of the methods we compare against, only the WordNet-based similarity measures, [93], and [99] provide a method for predicting verb similarities; our learned measure widely outperforms these methods, achieving a 13.6% F-score improvement over the Lesk similarity measure. In Figure 5.11 we give a precision/recall plot for verb sense merge judgments, plotting the performance of the three best WordNet-based similarity measures; here we see that our classifier has significantly higher precision than all other tested measures at nearly every level of recall.

Only the measures provided by Lesk, HSO, VEC, [93], and [99] provide a method for predicting adjective similarities; of these, only Lesk and Vec outperform the uninformed baseline on adjectives, while our learned measure achieves a 4.0% improvement over the Lesk measure on adjectives.

5.5.2 Feature analysis

Next we analyze our feature space. Table 5.4 gives the ablation analysis for all features used in our system as evaluated on our held-out test set; here the quantity listed in the table is the F-score loss obtained by removing that single feature from our feature space, and retraining and retesting our classifiers, keeping everything else the same. Here negative scores correspond to an *improvement* in classifier performance with the removal of the feature.

For noun classification, the three features that yield the highest gain in testset F-score are the topic signature, OED, and derivational link features, yielding a 4.0%, 3.6%, and 3.5% gain, respectively.

For verb classification, we find that three features yield more than a 5% F-score gain; by far the largest single-feature performance gain for verb classification found in our ablation study was the Deriv feature, i.e., the count of shared derivational links between the two synsets; this single feature improves our maximum F-score by 9.8% on the testset. This is a particularly interesting discovery, as none of the

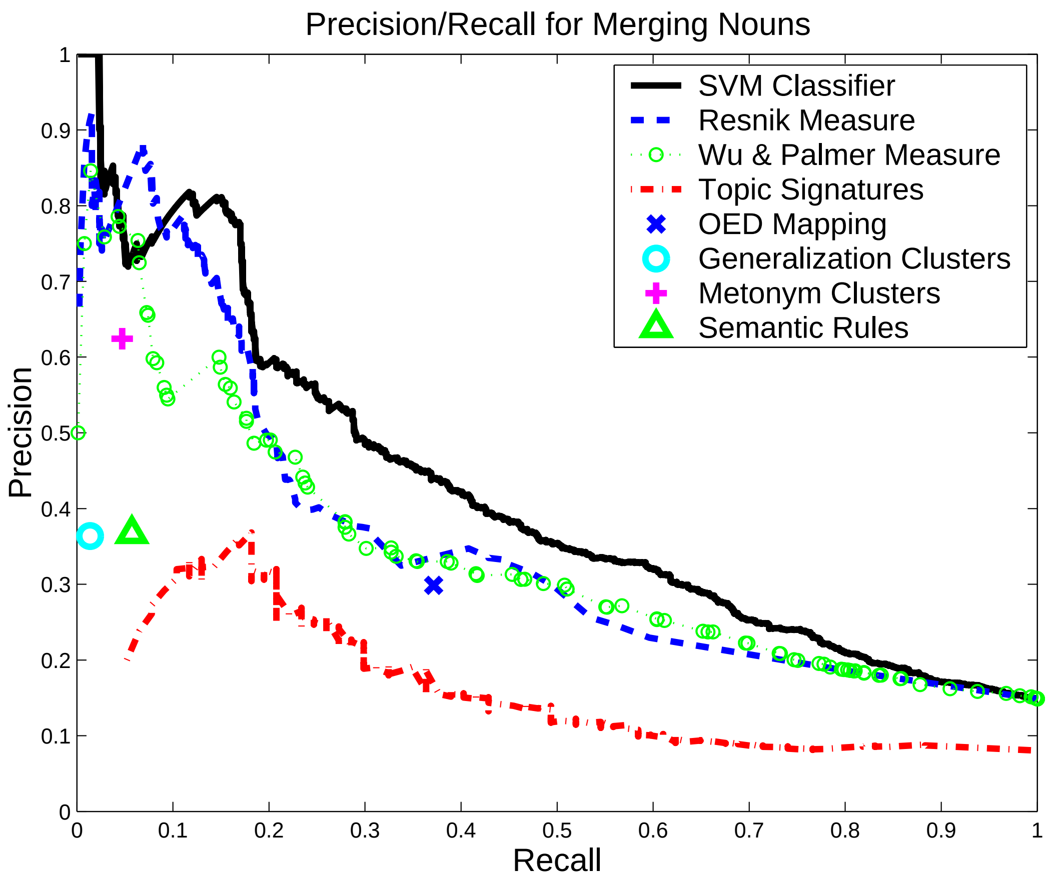


Figure 5.10: Precision/Recall plot for noun sense merge judgments

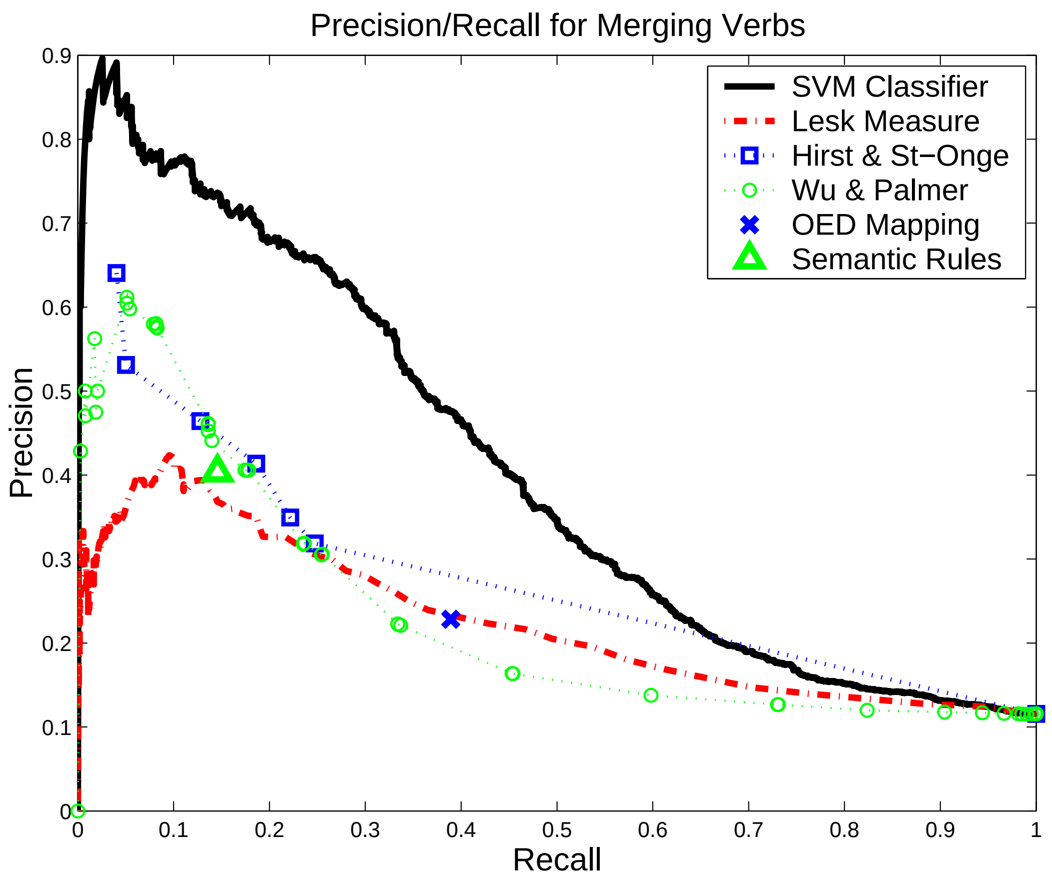


Figure 5.11: Precision/Recall plot for verb sense merge judgments

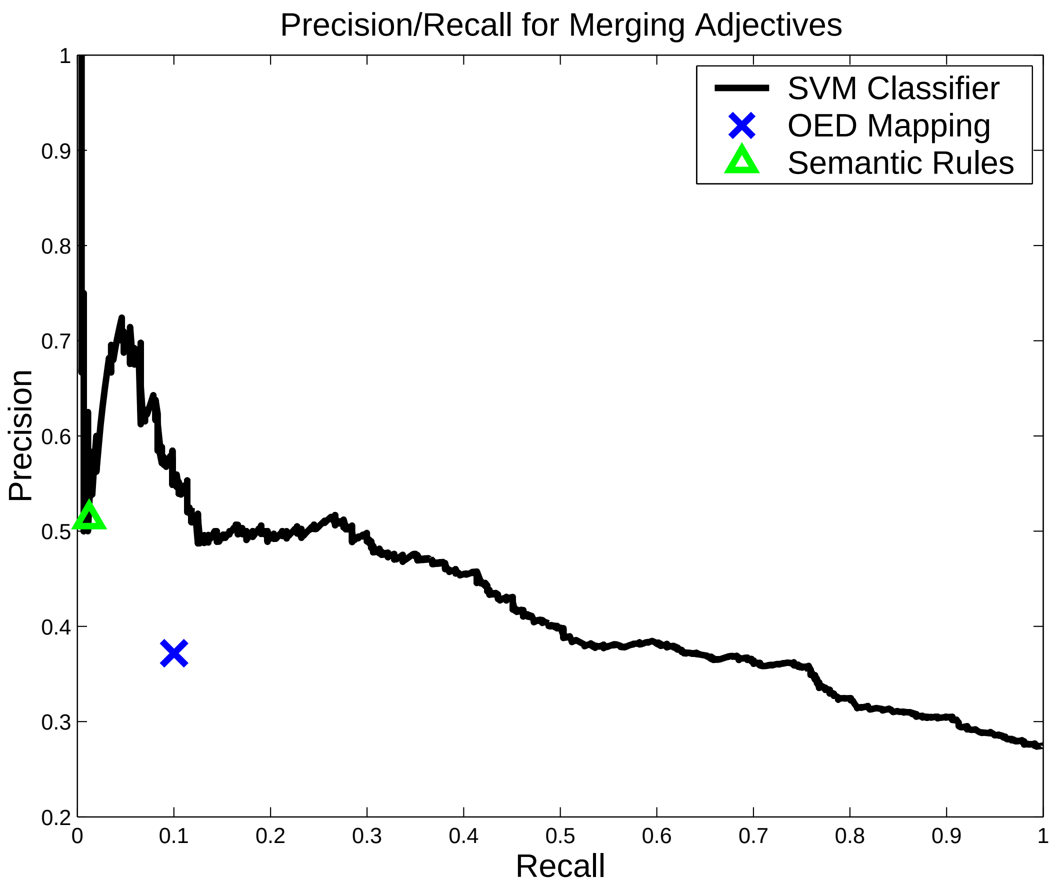


Figure 5.12: Precision/Recall plot for adjective sense merge judgments

	Nouns	Verbs	Adjectives
F-SCORE	0.4228	0.4319	0.4727
Feature	F-Score Ablation Difference		
TOPSIG	0.0403	—	—
OED	0.0355	0.0126	-0.0124
DERIV	0.0351	0.0977	0.0352
RES	0.0287	0.0147	—
TWIN	0.0285	0.0109	-0.0130
MN	0.0188	0.0358	—
LESK	0.0183	0.0541	-0.0250
SENSENUM	0.0155	0.0146	-0.0147
SENSECNT	0.0121	0.0160	0.0168
DOMAIN	0.0119	0.0082	-0.0265
LCH	0.0099	0.0068	—
WUP	0.0036	0.0168	—
JCN	0.0025	0.0190	—
ANTONYM	0.0000	0.0295	0.0000
MAXMN	-0.0013	0.0179	—
VEC	-0.0024	0.0371	-0.0062
HSO	-0.0073	0.0112	-0.0246
LIN	-0.0086	0.0742	—
COUSIN	-0.0094	—	—
VERBGRP	—	0.0327	—
VERBFRM	—	0.0102	—
PERTAINYM	—	—	-0.0029

Table 5.4: Feature ablation study: F-score difference obtained by removal of the single feature

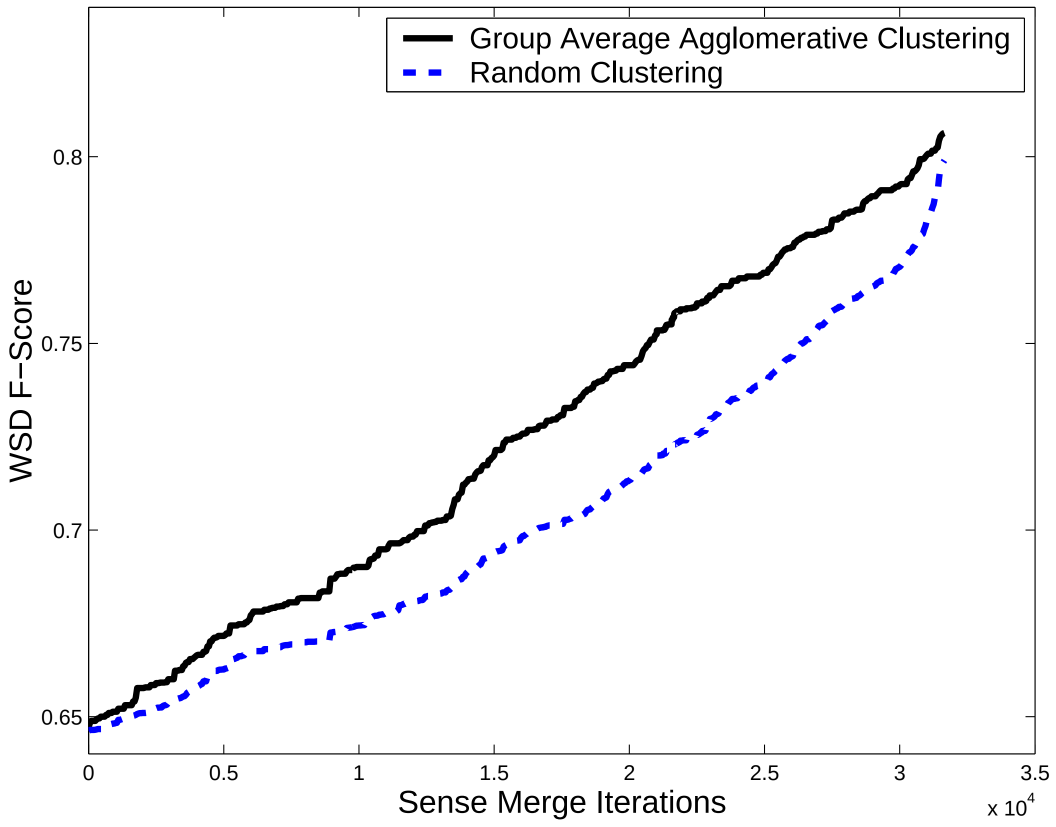


Figure 5.13: WSD Improvement with coarse-grained sense hierarchies

referenced automatic techniques for sense clustering presently make use of this very useful feature. We also achieve large gains with the LIN and LESK similarity features, with F-score improvement of 7.4% and 5.4% gain respectively.

For adjective classification again the DERIV feature proved very helpful, with a 3.5% gain on the testset. Interestingly, only the DERIV feature and the SENSECNT features helped across all parts of speech; in many cases a feature which proved to be very helpful for one part of speech actually hurt performance on another part of speech (e.g., LIN on nouns and OED on adjectives).

5.5.3 Evaluation of sense-clustered Wordnets

Our goal in clustering a sense taxonomy is to produce fully sense-clustered WordNets, and to be able to produce coarse-grained Wordnets at many different levels of resolution. In order to evaluate the entire sense-clustered taxonomy, we have employed an evaluation method inspired by Word Sense Disambiguation (this is similar to an evaluation used in [99], however we do not remove monosemous clusters). Given past system responses in the SENSEVAL-3 English all-words task, we can evaluate past systems on the same corpus, but using the coarse-grained sense hierarchy provided by our sense-clustered taxonomy. We may then compare the scores of each system on the coarse-grained task against their scores given a random clustering at the same resolution. Our expectation is that, if our sense clustering is much better than a random sense clustering (and, of course, that the WSD algorithms perform better than random guessing), we will see a marked improvement in the performance of WSD algorithms using our coarse-grained sense hierarchy.

We consider the outputs of the top 3 all-words WSD systems that participated in SENSEVAL-3: Gambl [44], SenseLearner [92], and KOC University [159]. A guess by a system is given full credit if it was either the correct answer or if it was in the same cluster as the correct answer.

Clearly any amount of clustering will only increase WSD performance. Therefore, to account for this natural improvement and consider only the effect of our particular clustering, we also calculate the expected score for a random clustering of the same granularity, as follows: Let C represent the set of clusters over the possible N synsets containing a given word; we then calculate the expectation that an incorrectly-chosen sense and the actual correct sense would be clustered together in the random clustering as $\frac{\sum_{c \in C} |c|(|c|-1)}{N(N-1)}$.

Our sense clustering algorithm provides little improvement over random clustering when too few or too many clusters are chosen; however, with an appropriate threshold for average-link clustering we find a maximum of 3.55% F-score improvement in WSD over random clustering (averaged over the decisions of the top 3 WSD algorithms). Table 5.5 shows the improvement of the three top WSD algorithms given a sense clustering created by our algorithm vs. a random clustering at the same granularity.

System	F-score	Avg-link	Random	Impr.
Gambl	0.6516	0.7702	0.7346	0.0356
SenseLearner	0.6458	0.7536	0.7195	0.0341
KOC Univ.	0.6414	0.7521	0.7153	0.0368

Table 5.5: Improvement in Senseval-3 WSD performance using our average-link agglomerative clustering vs. random clustering at the same granularity

5.6 Discussion

We have presented a classifier for automatic sense merging that significantly outperforms previously proposed automatic methods. In addition to its novel use of supervised learning and the integration of many previously proposed features, it is interesting that one of our new features, the Deriv count of shared derivational links between two synsets, proved an extraordinarily useful new cue for sense-merging, particularly for verbs.

We also show how to integrate this sense-merging algorithm into a model for clustering these senses of taxonomies like WordNet, with the additional capability of incorporating taxonomic constraints such as the transitive effects of merging synsets. Using this model, we have produced several WordNet taxonomies of various sense granularities; we hope these new lexical resources will be useful for NLP applications that require a coarser-grained sense hierarchy than that already found in WordNet.

Chapter 6

Applications of Semantic Taxonomy Induction

6.1 Introduction

A wide variety of lexical taxonomies are available for use in the natural language processing community; WordNet [50] is the most commonly used, and a number of taxonomies with a similar construction have been created [23, 29, 143]. While much work has focused on the automatic creation or augmentation of lexical taxonomies, the usefulness and application of these resources is not well understood. It has been claimed that these taxonomies might eventually contribute to significantly improved performance on a wide range of NLP tasks, as they can provide relational knowledge between words that might be difficult to obtain or represent through other means.

Due to the lack of comparative evaluations of these taxonomies in previous work, our understanding of how effectively different taxonomies may be applied to various problems is extremely limited. A wide variety of research has focused on a multitude of different relations, further complicating attempts to compare taxonomies; for example, if one algorithm induces hypernym (IS-A) relations, and another induces synonym relations, it is difficult to say which algorithm is better. Further, even if two algorithms both only find hypernym relations, an algorithm that achieves higher precision and recall as evaluated by the hypernyms it infers may not necessarily be the superior one if the relations it finds are less applicable to a particular problem domain, e.g., if all the relations it finds are of infrequently occurring terms.

While some previous comparisons of lexical resources have been conducted, there have been only a small number of application-based tasks proposed for comparison of lexical taxonomies. Exceptions may be found in the work of [38] and [39]; there they compare several sense-tagged lexical resources in an application to word sense disambiguation. While this comparison is interesting, it does not evaluate the hierarchical organization of the taxonomies it evaluates; in fact, no use is made of the hypernym relation at all.

In this chapter we discuss the topic of applying automatically inferred taxonomic relations to applied tasks in natural language processing. In Section 6.2 we discuss the application of our inferred taxonomies to the problem of question answering in the QACTIS system; in Section 6.3 we apply our inferred taxonomies to the problem of text categorization; in Section 6.4 we discuss the application of our model to domain-specific taxonomy induction; in Section 6.5 we consider the problem of applying our framework to inferring foreign language taxonomies; finally, in Section 6.6 we summarize our applications and conclude.

6.2 Question answering

To correctly extract answers, modern question answering systems depend on matching words between questions and retrieved passages containing answers. We consider the problem of applying augmented taxonomies to question answering because we believe that the learned hyponym relations can be exploited to improve question answering performance on a significant class of questions.

For example, consider the following questions:

- What island produces Blue Mountain coffee?
- In which game show do participants compete based on their knowledge of consumer prices?
- What villain is the nemesis of Dudley Do-Right?

Knowledge that *Jamaica* is an island, that *The Price is Right* is a game show, and that *Snidely Whiplash* is a villain, is crucial to answering these questions.

Sometimes these relations are evident in the same context as answers to questions, for example, in *"The island of Jamaica is the only producer of Blue Mountain coffee"*; however, *"Jamaica is the only producer of Blue Mountain coffee"* should be sufficient, despite the fact that it is not directly inferable from the sentence that Jamaica is an island.

The dynamic nature of named entities (NEs) makes it difficult to enumerate all of their evolving properties; thus manual creation and curation of this information in a lexical resource such as WordNet [50] is problematic. Pasca and Harabagiu discuss how insufficient coverage of named entities impairs QA [109]. They write:

> "Because WordNet was not designed as an encyclopedia, the hyponyms of concepts such as *composer* or *poet* are illustrations rather than an exhaustive list of instances. For example, only twelve composer names specialize the concept *composer* ... Consequently, the enhancement of WordNet with NE information could help QA."

The chief contribution of this study is demonstrating that an automatically mined knowledge base, which naturally contains errors as well as correctly distilled knowledge, can be used to improve QA performance. In our work we have demonstrated how our s can be tailored to identify named entity hyponyms, and how as a result, superior question answering performance can be obtained. We ranked candidate hyponyms on 75 categories of named entities and attained 53% mean average precision. On TREC QA data our method produces a 9% improvement in performance.

One area where we have seen gains has been the application of automatically inferred taxonomies to improving the QACTIS question answering system on the TREC 2005-2006 QA task [91]. This work has shown that large-scale, weakly-supervised hyponym learning is capable of producing improvements in an end-to-end QA system. This is an important contribution, as previous studies have often proposed the intuitive notion that discovering is-a relations for entities would improve factoid QA, but have failed to demonstrate gains in a realistic application. In this work we have

	Size	**Sentences**	**Genre**
TREC Disks 4,5	81 MB	0.70 M	Newswire
AQUAINT	1464 MB	12.17 M	Newswire
Wikipedia (4/04)	357 MB	3.27 M	Encyclopedia

Table 6.1: Sources used for training named entity hyponym classifiers

shown that using automatically inferred hyponyms can generate substantial performance gains over WordNet for question answering on certain query types.

The acquisition of novel named entity hyponyms can be viewed as solving the Named Entity Recognition problem as a subtask; rather than simply categorizing named entities into a small number of categories (e.g., *person*, *group*, *location*, *quantity*), grounding the novel hyponyms to a semantic taxonomy returns a much finer-grained classification. For example, using this approach someone might be classified under the *author* or *president* synset, or a company might be classified as an *airline* or *bank* rather than simply as a *person* or *organization*, respectively.

6.2.1 Experimental design

We rely on large amounts of text; in all our experiments we worked with a corpus from the sources given in Table 6.1. Sentences that presented difficulties in parsing were removed and those remaining were parsed with Minipar [81]. We extracted 17.3 million noun pairs that co-occurred in at least one sentence. All pairs were viewed as potential hypernym/hyponym pairs.

The experimental conditions for our three classifiers are summarized in Table 6.2. The baseline model used 71,068 pairs as training data; it is comparable to the distantly-supervised hypernym classifier discussed in Chapter 2 which used only dependency parse features, although here the corpus is larger. The entity-enriched data extended the baseline training set by adding positive examples. The +Feat model uses additional features besides dependency paths.

	Pos. Pairs	**Neg. Pairs**	**Total Features**
Baseline	7975	63093	162528
+NE	9331	63093	164298
+Feat	7975	63093	162804

Table 6.2: Characteristics of training sets

Feature	**Comment**
Hypernym contained in hyponym	*Sands Hotel* is-a *hotel*
Length in chars / words	Chars: 1-4, 5-8, 9-16, 17+ Words: 1, 2, 3, 4, 5, 6, 7+
Has preposition	*Treaty of Paris*; *Statue of Liberty*
Common suffixes	*-ation*, *-ment*, *-ology*, etc...
Figurative term	Such as *goal*, *basis*, or *problem*
Abstract category	Like *person*, *location*, *amount*
Contains digits	Usually not a good hyponym
Day of week; month of year	Indiscriminately co-occurs with many nouns.
Presence and depth in WordNet graph	Shallow hypernyms are unlikely to have entity hyponyms. Presence in WN suggests word is not an entity.
Lexname of 1st synset in WordNet	Root classes like *person*, *location*, *quantity*, and *process*.
Capitalization	Helps identify entities.
Binned document frequency	Partitioned by base 10 logs

Table 6.3: Features considered for +Feat model

Additional Features

The +Feat model incorporates an additional 276 binary features which are listed in Table 6.3.

To evaluate the usefulness of our learned NE hyponyms for question answering, we used the QACTIS system [133]. QACTIS was fielded at the 2004-2006 TREC QA evaluations and placed fifth at the 2005 workshop. We worked with a version of the software from July 2005.

QACTIS uses WordNet to improve matching of question and document words,

	Classes	**Class Instances**
Baseline	76	11,066
SFD	1,140	75,647
SWN	7,327	458,370
+Feat	44,703	1,868,393

Table 6.4: Additional knowledge sources by size

and a resource, the Semantic Forest Dictionary (SFD), which contains many hypernym/hyponym pairs. The SFD was populated through both automatic and manual means [133], and was updated based on questions asked in TREC evaluations through 2004.

We used factoid questions from the TREC 2005-2006 QA evaluations [149] and measured performance with mean reciprocal rank (MRR) and percent correct at rank 1. All runs made use of WordNet 2.0, and we examined several other sources of hypernym knowledge. The baseline condition added a small subset of the Semantic Forest Dictionary consisting of 76 classes seen in earlier TREC test sets (e.g., nationalities, occupations, presidents). We also tested: (1) the full SFD; (2) the Stanford Wordnet with 30,000 additional nodes discussed in Section 4.6.5; and, (3) the +Feat model. The number of classes and entries of each is given in Table 6.4.

6.2.2 Evaluation

To compare our different models we created a test set of 75 categories. The classes are diverse and include personal, corporate, geographic, political, artistic, abstract, and consumer product entities. From the top 100 responses of the different learners, a pool of candidate hyponyms was created, randomly reordered, and manually judged. To assess the quality of purported hyponyms we used average precision, a measure in ranked information retrieval evaluation, which combines precision and recall.

Table 6.5 gives average precision values for the three models on 15 classes of mixed difficulty. Performance varies considerably based on the hypernym category, and for a given category, by classifier. *N* is the number of known correct instances found in the pool that belong to a given category. Aggregate performance, as mean average

	N	**Baseline**	**+NE**	**+Feat**
chemical element	78	0.9096	0.9781	0.8057
african country	48	0.8581	0.8521	0.4294
prep school	26	0.6990	0.7098	0.7924
oil company	132	0.6406	0.6342	0.7808
boxer	109	0.6249	0.6487	0.6773
sculptor	95	0.6108	0.6375	0.8634
cartoonist	58	0.5988	0.6109	0.7097
volcano	119	0.5687	0.5516	0.7722
horse race	23	0.4837	0.4962	0.7322
musical	80	0.4827	0.4270	0.3690
astronaut	114	0.4723	0.5912	0.5738
word processor	26	0.4437	0.4426	0.6207
chief justice	115	0.4029	0.4630	0.5955
perfume	43	0.2482	0.2400	0.5231
pirate	10	0.1885	0.3070	0.2282

Table 6.5: Average precision on 15 categories

	Baseline	**+NE**	**+Feat**
MAP	0.4801	0.5001 (+4.2%)	0.5320 (+10.8%)

Table 6.6: Mean average precision over 75 categories

precision, was computed over all 75 categories and is given in Table 6.6. Both the +NE and +Feat models yielded improvements that were statistically significant at a 99% confidence level. The +Feat model gained 11% over the baseline condition. The maximum F-score for +Feat is 0.55 at 70% recall.

Mean average precision emphasizes precision at low ranks, so to capture the error characteristics at multiple operating points we present a precision-recall graph in Figure 6.1.The +NE and +Feat models both attain superior performance at all but the lowest recall levels. For question answering this is important because it is not known which entities will be the focus of a question, so the ability to deeply mine various entity classes is important.

Table 6.7 lists top responses for four categories. Helpful examples of named entity

	Sculptor	**Horse Race**	**Astronaut**	**Perfume**
1	Evelyn Beatrice Longman	Tevis Cup	Mark L Polansky	* Avishag
2	Nancy Schon	Kenilworth Park Gold Cup	Richard O Covey	Ptisenbon
3	Phidias	Cox Plate	George D Nelson	Poeme
4	Stanley Brandon Kearl	Grosser Bugatti Preis	Guion Bluford Jr	Parfums International
5	Andy Galsworthy	Melbourne Cup	Stephen S Oswald	Topper Schroeder
6	Alexander Collin	* Great Budda Hall	Eileen Collins	* Baccarin
7	Rachel Feinstein	Travers Stakes	Leopold Eyharts	Pink Lady
8	Zurab K Tsereteli	English Derby	Daniel M Tani	Blue Waltz
9	Bertel Thorvaldsen	* Contrade	Ronald Grabe	WCW Nitro
10	Cildo Meireles	Palio	* Frank Poole	Jicky
11	Victor Salmones	Bastille Stakes	Toktar Aubakirov	* Popular Muppet
12	Nico Yektai	Kentucky Derby	Wilson Kime	* Mish Mosh
13	Grace Weir	* El Nuevo Comandante	William Readdy	Bulgari
14	Alexander Munro	* Jax Jacobsen	Ellison Onizuka	Kenzo
15	Ludwig Gies	St. Leger Stakes	Alan B Shepard Jr	* Jalapa

Table 6.7: Top responses for four categories using the +Feat model, starred entries were judged incorrect

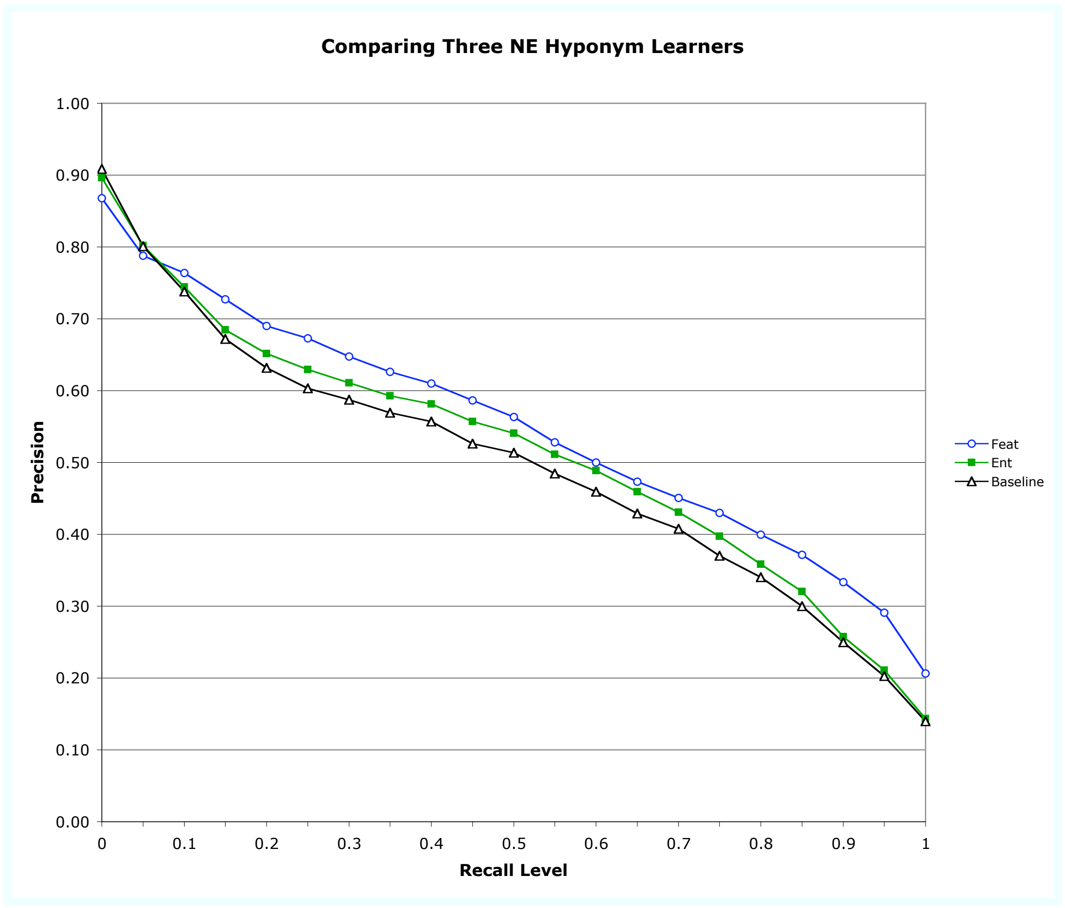


Figure 6.1: Precision-recall graph for three hyponym/hypernym classifiers

hyponyms include learning a meteorite (Q84.1), a university (Q93.3), a chief operating officer (Q108.3), a political party (Q183.3), a pyramid (Q186.4), and a movie (Q211.5).

In Table 6.8 we compare performance on questions from the 2005 and 2006 test sets. We assessed performance primarily on test questions that were deemed likely to benefit from hyponym knowledge—questions that had a readily discernible category (e.g., *"What film ..."*, *"In what country ..."*)—but we also give results on the entire test set. The WordNet-only run suffers a large decrease compared to the baseline. This is expected because WordNet lacks coverage of entities and the baseline condition specifically populates common categories of entities that have been observed in prior

	Hyponym-Relevant Subset (242)		**All Questions (734)**	
	MRR	**% Correct**	**MRR**	**% Correct**
WN-alone	0.189 (-45.6%)	12.8 (-51.6%)	0.243 (-29.0%)	18.26 (-30.9%)
Baseline	0.348	26.4	0.342	26.4
SFD	0.405 (+16.5%)	31.0 (+17.2%)	0.362 (+5.6%)	27.9 (+5.7%)
SWN	0.351 (+1.0%)	26.9 (+1.6%)	0.343 (+0.3%)	26.6 (+0.5%)
Feat	0.373 (+7.4%)	28.9 (+9.4%)	0.351 (+2.5%)	27.3 (+3.1%)

Table 6.8: QA Performance on TREC 2005 & 2006 Data

TREC evaluations. Nonetheless, WordNet is useful to the system because it addresses lexical mismatch that does not involve entities.

The full SFD, the SWN, and the +Feat model achieved 17%, 2%, and 9% improvements in answer correctness, respectively. While no model had exposure to the 2005-2006 TREC questions, the SFD database was manually updated based on training on the TREC-8 through TREC-2004 data sets. It approximates an upper bound on gains attributable to addition of hyponym knowledge: it has an unfair advantage over the other models because recent question sets use similar categories to those in earlier TRECs. Our +Feat model, which has no bias towards TREC questions, realizes larger gains than the SWN. This is probably at least in part because it produced a more diverse set of classes and a significantly larger number of class instances. Compared to the baseline condition the +Feat model sees a 7% improvement in mean reciprocal rank and a 9% improvement in correct first answers; both results represent a doubling of performance compared to the use of WordNet alone. We believe that these results illustrate clear improvement attributable to automatically learned hyponyms.

6.3 Text classification

This section describes the application of taxonomic information to the problem of text categorization. We specifically focus on the use of noun synonym and hypernym information within taxonomies, as this is among the most commonly applied and inferred form of taxonomic knowledge. We choose the text categorization task because it is a well-studied task with clear practical benefit. We perform our comparison using two canonical text categorization datasets: the REUTERS-21578 collection[1] and the 20-NEWSGROUPS dataset[2] [76]. We expect the effect of lexical taxonomies to be most pronounced in the setting where there are relatively few training examples, and where additional information supplied by the taxonomy therefore has the greatest relative effect; thus it is also in this setting that we expect that the relative improvement conferred upon a text classification algorithm can be most easily studied.

Text categorization has been widely studied, though most work in text classification has typically assumed that the categorization algorithm lacks any source of world knowledge or external semantic features. An excellent summary of text classification excluding the use of world knowledge may be found in [135]. While most work in text classification has eschewed the use of external lexical resources, a few studies have considered the effect of using lexical resources. These studies have come to mixed conclusions; [134] found small but statistically significant improvements using WordNet hypernyms on the USENET dataset and a dataset of folk lyrics, but no improvement on the Reuters dataset; [42] found large gains on the REUTERS-21578 dataset using synonym expansion from manually disambiguated datasets, and [87] found no positive effect from a variety of features derived from WordNet as input to Naive Bayes and SVM classifiers on any of four major document collections.

A related area where the effect of lexical taxonomies has been studied is that of information retrieval; while initial studies showed no improvement for detailed queries, and relatively small gains for shorter queries [150], more recent studies have shown that more refined techniques for query expansion can significantly improve performance on standard TREC collections [84, 49, 97].

Throughout the course of these experiments the primary lexical resource evaluated has been WordNet, and there has so far been little comparison between WordNet and other similarly structured resources. The recent introduction of multiple WordNet-like resources raises the question of whether these new resources can significantly

[1]The REUTERS-21578 collection is available at:
`http://www.daviddlewis.com/resources`.

[2]We use the dataset provided at:
`http://people.csail.mit.edu/jrennie/20Newsgroups`.

increase the performance of existing systems above and beyond what has already been found using WordNet; in our evaluation we address this question by providing a comparison of text categorization using WordNet, YAGO, a taxonomy built from the Fleischman instances, and two of the Wordnets we have constructed using our framework for semantic taxonomy induction: the Stanford Wordnet with 40,000 automatically-inferred synsets, and the Stanford Wordnet with 450,000 automatically-inferred synsets.

In our application to text classification we begin with a simple baseline Naive Bayes text classification procedure. We then augment this baseline with a straightforward query expansion procedure that depends only on the synonym and hypernym links within a taxonomy, and is otherwise independent of the particular taxonomy involved. For each of the taxonomies we wish to evaluate, we thus perform a simple feature expansion, retrain the Naive Bayes classifier, and evaluate the change in performance on the text classification task.

6.3.1 Experimental design

We perform text categorization experiments over both the Reuters-21578 and 20-Newsgroups datasets. For the Reuters-21578 dataset we hold out a test set of 1,535 documents across the 18 most populated categories. We then run a series of experiments with an increasing number of training examples in the range [1, 2, 5, 10, 15, 20].

For the 20-Newsgroups set we use the dataset sorted by date, with duplicate articles removed, and all newsgroup-identifying information removed[3]. Here we have a test set of 7,532 documents across 20 different categories, and we similarly stage a set of experiments using an increasing number of training examples in the range [1, 2, 5, 10, 15, 20].

[3] We use the '*bydate*' dataset available at
`http://people.csail.mit.edu/jrennie/20Newsgroups`.

Document classification with Naive Bayes

Here we describe our baseline model for document classification, including our tokenization, part-of-speech tagging, lemmatization, phrase recognition, and stopword removal.

For each document we tokenize and part-of-speech tag the sentence using the Stanford Parser[4]. Since we are comparing the noun hypernym relations inferred in the different taxonomies, we remove any token from our representation of the document that is not tagged as a noun.

After tokenization and part-of-speech tagging we apply the WordNet MORPHY function to each lexeme; this applies a set of heuristics in order to determine the possible lemmas that a particular lexeme might be an inflected form of. If more than one lemma known to WordNet is found by MORPHY, we choose the lemma that has the highest frequency count in the SemCor corpus [95].

Next we apply the specific taxonomy $\mathbf{T}$ currently being evaluated to recognize potential noun multiword entities in the sentence using the following greedy approach: starting from the first token in each sentence we progress until we find the first noun n_1; then, we identify the longest noun phrase p in the sentence beginning with n_1 that is also in $\mathbf{T}$; we then chunk this as a phrase and repeat this process starting with the next noun in the sentence not already contained in p. Finally, for all single tokens that have not been chunked into a larger multiword phrase, we perform stopword removal[5]. Given the remaining lemmas and phrases W, our baseline model then performs text categorization by assigning the category label according to the highest probability in a Naive Bayes model over the bag-of-lemmas. Specifically, we choose the class C that maximizes the function:

$$C' = \arg\max_c P(c) \prod_{w_i \in W} P(w_i|c)$$

[4]The Stanford Parser is available for download at `http://nlp.stanford.edu/software`.

[5]We use a list based on the one at `http://www.d.umn.edu/~tpederse/Group01/WordNet/words.txt`.

Query expansion using taxonomies

We extend our baseline model using query expansion in the following simple way: for each noun lemma or multiword noun phrase in each document, if that noun is recognized in a given taxonomy, we selectively add observations of other related nouns from the taxonomy. We perform a variety of possible expansions for each taxonomy. The parameters we vary are as follows: for a lemma w in the sentence, we choose the maximum number of senses s to expand; we choose the total number of synonyms from each sense to use; finally, we choose the maximum number of ancestors to add.

Specifically, we explore the addition of terms in a synonymy and hypernymy relation. This effectively results in augmenting the feature vector for each training and test document with additional lemmas corresponding to strongly-related concepts for each noun lemma.

For example, in the REUTERS-21578 dataset article we have an article belonging to the ACQ or "acquisition" topic which discusses *American Express*; here our taxonomy identifies the term *American_Express* as a hyponym of the WordNet synset *company#n#1*; if this article were in the training set, than this query expansion would lead the Naive Bayes classifier to increase the weighting of an observed hypernym synset *company#n#1* with the "acquisition" topic.

Similarly, in the 20-NEWSGROUPS dataset, an article in the *comp.graphics* contains the term *Adobe Illustrator*. Since our augmented taxonomy identifies this term as a hyponym of the synset *software#n#1*, then it will similarly associate this synset with the *comp.graphics* topic.

6.3.2 Evaluation

Here we consider the comparative evaluation of several taxonomies for use in the text classification task.

Comparison of taxonomies

In our evaluation of text classification we compare the Stanford Wordnets that we have constructed to three previously proposed taxonomies and to a Naive Bayes baseline

Taxonomy	Noun synsets
Baseline	0
WordNet	81426
Stanford Wordnet + 40K	121426
YAGO	376919
Stanford Wordnet + 450K	531426
Fleischman	783928

Table 6.9: Summary of the sizes of the taxonomies in our comparison

that does not use a taxonomy.

[52] proposes an instance discovery method for automatically inferring novel instances from text, and published a list of over 2,000,000 instances derived with this method; while these instances have not been sense-disambiguated, we can nonetheless consider them in a taxonomy by supposing they have been added to the first sense of the hypernym. We give a brief summary of the sizes of each of the taxonomies we compare in Table 6.9.

Taxonomy selection

In our experiments using different taxonomies we explored a wide parameter space in order to maximize the effectiveness of the taxonomies. These parameters include: the number of distinct senses of a word to use, the number of synonyms to use, and the number of steps to expand the query up the hypernym chain. In each case we performed parameter selection based on the best performance of WordNet 2.1; thus we are giving WordNet the best chance it can get, and using the same parameters for the rest of the taxonomies. The best parameters for the Reuters-21578 corpus was to use only a single sense (the first listed sense in WordNet), up to 5 synonyms, and expand up to 2 steps in the hypernym chain, while the best parameters for the 20-Newsgroups data was to use only a single sense (again, the first listed sense in WordNet), up to 2 synonyms, and expand up to 2 steps in the hypernym chain.

System	R-1	R-2	R-5	R-10	R-15	R-20
Baseline	0.358	0.371	0.501	0.586	**0.666**	**0.676**
WordNet	0.359	0.423	0.515	**0.608**	0.638	0.650
Fleischman	0.455	0.444	0.551	0.607	0.638	0.650
YAGO	0.455	0.444	0.550	0.608	0.638	0.650
SWN-40K	**0.479**	0.442	**0.553**	0.606	0.635	0.650
SWN-450K	0.469	**0.448**	0.551	0.604	0.636	0.652

Table 6.10: Accuracy of Naive Bayes text categorization using different taxonomies in the Reuters-21578 dataset for different training set sizes

The Reuters-21578 dataset

As shown in Table 6.10, the text categorization performance using taxonomies all track very closely. In particular, the performance using the taxonomies derived from YAGO and from the Fleischman instances are nearly indistinguishable to themselves or to the original WordNet.

The 20-Newsgroups dataset

As shown in Table 6.11 we found that, as in the Reuters-21578 datase, the set of augmented taxonomies perform very similarly. Indeed, the taxonomies based on YAGO and the Fleischman instance data are nearly indistinguishable from using WordNet alone, with our augmented taxonomy marginally better for up to two training examples. Also, as in the Reuters data, we observe that after a small number of training examples (in this case 5 training examples) the Naive Bayes baseline without a taxonomy is superior compared to using any of the taxonomies.

Our results suggest that taxonomies are indeed helpful in the text categorization task, but only when the amount of training data is very small; using a taxonomy actually hurts performance after there are more than 5 examples (in the case of the 20-Newgroups data) or more than 10 examples (in the case of the Reuters-21578 data). Further, our comparison suggests that acquiring more training examples is almost always a better choice than using a taxonomy; in almost all cases the baseline algorithm with more examples performs as well or better than an algorithm using a

System	N-1	N-2	N-5	N-10	N-15	N-20
Baseline	0.352	0.399	0.458	**0.520**	**0.550**	**0.579**
WordNet	0.367	0.401	0.460	0.500	0.523	0.544
Fleischman	0.367	0.401	0.460	0.500	0.522	0.544
YAGO	0.367	0.401	0.460	0.500	0.523	0.544
SWN-40K	0.366	0.400	**0.463**	0.499	0.525	0.545
SWN-450K	**0.371**	**0.405**	0.462	0.498	0.520	0.540

Table 6.11: Accuracy of Naive Bayes text categorization using different taxonomies in the 20-Newsgroups dataset for different training set sizes

taxonomy with fewer examples.

6.4 Inferring domain-specific taxonomies

In addition to inferring general purpose knowledge, our probabilistic taxonomy framework offers opportunities for creating domain-specific ontologies. One area that is particularly relevant for the application of our framework is the biomedical research literature domain. Text drawn from the biomedical research literature is particularly promising for taxonomy induction as it has a large vocabulary of domain-specific terminology and a wide range of typed entities (e.g., cells, proteins, hormones, etc.) and structured relations between entities (e.g., *X regulates Y*, *X phosphorylates Y*, etc.). Further, just as WordNet provided a seed set of relationship tuples for general-purpose taxonomy induction, there already exist a large number of rich, hand-curated databases explicitly stating known relationships between biological entities, such as the Gene Ontology [6] and the Open Biomedical Ontologies[6].

In the same way that WordNet provided the initial seed data for eventually extending WordNet, we can directly apply the same relationship extraction techniques and taxonomy induction formulation to augment these biological ontologies using biomedical text. A consequence of our method also allows us to decompose the input articles from biomedical literature into the constituent structured relationships that

[6]The Open Biomedical Ontologies are available at `http://obofoundry.org`.

those articles give evidence for; this leads to a promising new approach for representing articles in the biomedical literature in the form of structured, machine-readable relationships. The general framework for taxonomy induction we have described here has been applied to the problem of automatically extracting structured representations of more than one million full-text articles in the biomedical research domain in the form of "Structured Digital Abstracts" [142].

In this application we have demonstrated that, using the Gene Ontology as an initial seed set, our algorithm is capable of extracting dozens of different kinds of ontological relationships from free text, ranging from gene localizations to chemical modifications and anatomical structures. Moreover, we have shown that our algorithm is able to expand upon the set of known ontological relationships in the Gene Ontology by an order of magnitude. We anticipate that such techniques will have significant impact in the area of machine-readable content summary for intelligent search and indexing.

6.5 Augmenting foreign-language taxonomies

A major advantage of our probabilistic formulation for taxonomy likelihood is that it can be applied in any language. In this section we address the task of automated taxonomy mapping and taxonomy translation. Lexical resources are very scarce in the vast majority of languages. A number of languages have thesauruses like WordNet or projects to create them, but many do not. Automated techniques which could induce a foreign-language Wordnet via machine translation from the English thesaurus could vastly speed the process of creating thesauruses in many languages.

Consider the task of mapping from the English synsets S_e in WordNet to a sense taxonomy using the target language words F as nodes. For a particular lexicon of words F in the target language , for any particular mapping M of $F \rightarrow S_e$ (English WordNet synsets) we get a corresponding map N of foreign language words to foreign synsets $F \rightarrow S_f$. We can find the best taxonomy by ranking the hierarchies via global inference using multiple sources of evidence. Here we use the following definitions in our framework for automatically inferring foreign-language taxonomies:

translation probability $p(f|e)$**:** the probability of a word e in English mapping to a foreign word f. This can be derived from a parallel corpus using standard techniques in statistical machine learning (perhaps even simplified techniques like IBM Model 1 [16]), or from off-the-shelf engines.

foreign word similarity $S(w_1, w_2)$**:** a similarity function between two words in the foreign language. This can be derived from unsupervised distributional information in the foreign language [106].

taxonomy node distance $P(C_{mn}(w_1, w_2)|S(w_1, w_2))$**:** The probability of two words being a particular distance apart in the hierarchy, i.e., (m, n)-cousins C_{mn}, i.e. being m steps up and n steps down in the hypernym hierarchy from each other, given the similarity function $S(w_1, w_2)$. We have shown how to compute this function in Section 4.4.2.

Given this knowledge we can formulate a taxonomy likelihood as:

$$P(T_f|T_e) \propto \prod_{w_f} P(w_f|M(w_f)) \times \prod_{w_i, w_j \in F \text{ s.t. } C_{mn}(N(w_i), N(w_j)) \in T_f} P(C_{mn}(w_i, w_j)|S(w_i, w_j))$$

Our goal is then to find the mapping M' that will maximize this taxonomy likelihood, i.e.,

$$M' = \arg\max_M \prod_{w_f} P(w_f|M(w_f)) \times \prod_{w_i, w_j \in F \text{ s.t. } C_{mn}(N(w_i), N(w_j)) \in T_f} P(C_{mn}(w_i, w_j)|S(w_i, w_j))$$

Given this formulation, various search algorithms may be explored to find mappings which increase the conditional taxonomy likelihood. This approach can also be extended with additional knowledge, such as a mapping from the English gloss, a hypernym classifier or other relationship classifier in the target language, and other sources of evidence.

6.6 Discussion

In our experiments we have shown that our framework for taxonomy induction may be applied to the problems of domain-specific taxonomy induction, foreign-language taxonomy induction, question answering, and text categorization. In our experiments with question answering we have demonstrated that automatically inferred hyponyms can significantly improve the performance of a question answering system. In our experiments with text categorization we have provided an evaluation methodology for augmented taxonomies for the application of text categorization with various amount of training examples. We have further provided a comparison of several previously proposed lexical taxonomies.

Chapter 7

Conclusion

Bridging the gap between structured knowledge and unstructured text is a long-held goal of artificial intelligence and natural language processing. We have approached this problem by addressing the acquisition and organization of structured representations of binary semantic relations. We have shown that we can use existing knowledge bases and arbitrarily large unannotated bodies of text in order to train probabilistic relation classifiers using *distant supervision*. Our experiments have demonstrated that, given a sufficiently large unlabeled corpus, our automatic relation extraction algorithms trained with distant supervision can outperform WordNet for the identification of hypernym relations and coordinate term relations in newswire text.

Our formulation of taxonomy likelihood provides a framework allowing predictions from multiple classifiers to be combined. We have shown how this framework can be used to implicitly resolve lexical ambiguity and discover the appropriate mapping between words and the concepts within an existing structured representation of knowledge. We have discussed several operations that may be applied to a taxonomy in order to create modified versions of that taxonomy including adding and removing relations, and adding, removing, and merging concepts. Using this framework we have created several augmented Wordnets, and shown that these automatically-constructed resources outperform the manually-constructed WordNet on a variety of tasks.

We have also created a supervised algorithm for measuring sense similarity within a Wordnet, and shown that our algorithm performs favorably compared to previously proposed methods for measuring sense similarity based on a gold-standard testset. Using this algorithm we have constructed several sense-clustered Wordnets of varying degrees of sense granularity, and we have publicly released these lexical resources for use in natural language applications. Finally, we have discussed a variety of applications of our framework and lexical resources, including applications to domain-specific taxonomy induction, the extension of foreign-language Wordnets, question answering, and text categorization. Many open avenues of research follow from this work. Open questions remain in each of the areas we have discussed, particularly in the areas of relation extraction, taxonomy augmentation, and the use of taxonomies in applied tasks.

Our distantly-supervised relation extraction classifiers could be improved with the use of techniques previously proposed in the literature, as discussed in Section 1.1. The feature space we use in distantly-supervised relation extraction could be greatly expanded by the use of kernel methods. Further, our work could be extended to use the bootstrapping framework explored by Dipre, Snowball, and others. The taxonomy constraints we discuss in Section 3.2 might be employed to filter the possible terms used in successive iterations within a learning framework using bootstrapping.

An additional area we have not yet explored is the interaction between relation extraction and taxonomy induction. A promising extension of the bootstrapping framework would be to retrain a relation extraction classifier with the additional concepts and relations inferred in the process of taxonomy induction, then again use the classifier in a second round of taxonomy induction, and so forth. Such a framework suggests the idea of jointly maximizing the likelihood of the taxonomy and the likelihood functions of the base relation classifiers.

Our work in taxonomy augmentation has so far focused only on inferring leaf noun hypernyms corresponding to words that have no current representation within the existing taxonomy. Our primary focus has been the extension of large, previously existing taxonomies. Our framework could be extended to include the addition of intermediate nodes in the noun hypernym hierarchy, or adding nodes in the verb hypernym hierarchy. Further, there is a great deal of promise in using this framework for inferring cross-part-of-speech relations. Possible relations include, for instance, discovering the likely noun arguments of verbs, or the adjective classes likely to describe certain nouns, and so forth. There is also the possibility of inferring taxonomies from scratch, rather than merely augmenting existing taxonomies. These explorations would likely require a more nuanced search algorithm than the simple local search that we propose. Whereas the local search we propose works well when applied to a large, preexisting taxonomy, the problem of encountering pipeline errors becomes much more critical when inferring a taxonomy from scratch.

Finally, our model for semantic taxonomy induction has laid the groundwork for a wide variety of applications in computational semantics. We have only explored a fraction of these. In this work we have considered the problems of augmenting a taxonomy and merging the senses of a taxonomy; other possible applications include the possibility of pruning a taxonomy to make it relevant for a specific corpus. Such a modification might be helpful for domain-specific taxonomy induction; having senses in a taxonomy that are irrelevant to the domain of interest can only reduce the effectiveness of the taxonomy. We have given one framework for mapping between taxonomies of different languages in Section 6.5; a similar approach might be extended to consider mapping between taxonomies in the same language, but that use different terms to represent overlapping sets of information. Finally there is broad potential application of our framework for summarization and search, as touched upon in Section 6.4.

Bridging the gap between unstructured text and structured representations of knowledge is a crucial step towards the computational understanding of language. In this work we have presented several new powerful techniques for acquiring and organizing structured knowledge. We hope that the algorithms and resources that we have presented will provide a foundation for many future advances in acquiring and organizing machine-readable knowledge from unstructured text.

Bibliography

[1] Steven P. Abney. Understanding the Yarowsky algorithm. *Computational Linguistics*, 30(3):365–395, 2004.

[2] Eugene Agichtein and Luis Gravano. Snowball: Extracting relations from large plain-text collections. In *Proceedings of the 5th ACM International Conference on Digital Libraries*, 2000.

[3] Eneko Agirre and Oier Lopez de Lacalle. Clustering WordNet word senses. In *Proceedings of RANLP-03*, 2003.

[4] Eneko Agirre and Oier Lopez de Lacalle. Publicly available topic signatures for all WordNet nominal senses. In *Proceedings of LREC-04*, 2004.

[5] Hiyan Alshawi. Processing dictionary definitions with phrasal pattern hierarchies. *Computational Linguistics*, 13:13–3, 1987.

[6] M. Ashburner, C. A. Ball, J. A. Blake, D. Botstein, H. Butler, J. M. Cherry, A. P. Davis, K. Dolinski, S. S. Dwight, J. T. Eppig, M. A. Harris, D. P. Hill, L. Issel-Tarver, A. Kasarskis, S. Lewis, J. C. Matese, J. E. Richardson, M. Ringwald, G. M. Rubin, and G. Sherlock. Gene ontology: Tool for the unification of biology. *Nature Genetics 2000*, 2000.

[7] Satanjeev Banerjee and Ted Pedersen. Extended gloss overlaps as a measure of semantic relatedness. In *Proceedings of IJCAI-03*, 2003.

[8] Michele Banko, Michael J. Cafarella, Stephen Soderland, Matthew Broadhead, and Oren Etzioni. Open information extraction from the web. In *Proceedings of IJCAI-07*, 2007.

[9] Michele Banko and Oren Etzioni. The tradeoffs between traditional and open relation extraction. In *Proceedings of ACL-08*, 2008.

[10] Lisa Bentivogli, Pamela Forner, Bernardo Magnini, and Emanuele Pianta. Revising the WordNet domains hierarchy: Semantics, coverage, and balancing. In *Proceedings of COLING Workshop on Multilingual Linguistic Resources*, 2004.

[11] Matthew Berland and Eugene Charniak. Finding parts in very large corpora. In *Proceedings of ACL-99*, 1999.

[12] Daniel M. Bikel, Richard Schwartz, and Ralph M. Weischedel. An algorithm that learns what's in a name. *Machine Learning*, 34(1-3):211–231, 1999.

[13] Patrick Blackburn and Johan Bos. Computational semantics. *Theoria*, 18(46):27–45, 2003.

[14] Patrick Blackburn and Johan Bos. *Representation and Inference for Natural Language. A First Course in Computational Semantics.* CSLI, 2005.

[15] Sergei Brin. Extracting patterns and relations from the World Wide Web. In *Proceedings World Wide Web and Databases International Workshop, Number 1590 in LNCS*, pages 172–183. Springer, 1998.

[16] Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–311, 1993.

[17] Paul Buitelaar. *CoreLex: systematic polysemy and underspecification.* PhD thesis, Brandeis University, 1998.

[18] Paul Buitelaar, Philipp Cimiano, and Bernardo Magnini. *Ontology Learning from Text: Methods, Evaluation and Applications.* IOS Press, 2005.

[19] Razvan C. Bunescu and Raymond J. Mooney. Relational Markov Networks for collective information extraction. In *Proceedings of ICML-04 Workshop on Statistical Relational Learning*, 2004.

[20] Razvan C. Bunescu and Raymond J. Mooney. A shortest path dependency kernel for relation extraction. In *Proceedings of HLT-EMNLP-05*, 2005.

[21] Razvan C. Bunescu and Raymond J. Mooney. Subsequence kernels for relation extraction. In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Advances in Neural Information Processing Systems 18*, pages 171–178. MIT Press, Cambridge, MA, 2006.

[22] Michael J. Cafarella, Alon Halevy, Daisy Zhe Wang, Eugene Wu, and Yang Zhang. WebTables: exploring the power of tables on the web. *Proceedings of VLDB-08*, 2008.

[23] Sharon A. Caraballo. *Automatic Acquisition of a Hypernym-Labeled Noun Hierarchy from Text*. PhD thesis, Brown University, 2001.

[24] Andrew Carlson, Justin Betteridge, Estevam R. Hruschka Jr., and Tom M. Mitchell. Coupling semi-supervised learning of categories and relations. In *Proceedings of the NAACL-HLT 2009 Workshop on Semi-supervised Learning for Natural Language Processing*, 2009.

[25] John Carroll, Edward Briscoe, and Antonio Sanlippo. Parser evaluation: A survey and a new proposal. In *Proceedings of LREC-98*, 1998.

[26] Scott Cederberg and Dominic Widdows. Using LSA and noun coordination information to improve the precision and recall of automatic hyponymy extraction. In *Proceedings of CoNLL-03*, 2003.

[27] Nancy Chinchor, editor. *Proceedings of MUC-7*, 1998.

[28] Timothy Chklovski and Rada Mihalcea. Exploiting agreement and disagreement of human annotators for word sense disambiguation. In *Proceedings of RANLP-03*, 2003.

[29] Timothy Chklovski and Patrick Pantel. VerbOcean: Mining the web for fine-grained semantic verb relations. In *Proceedings of EMNLP-04*, 2004.

[30] Irina Chugur, Julio Gonzalo, and Felisa Verdejo. Polysemy and sense proximity in the Senseval-2 test suite. In *Proceedings of ACL-02 WSD Workshop*, 2002.

[31] Massimiliano Ciaramita and Mark Johnson. Supersense tagging of unknown nouns in WordNet. In *Proceedings of EMNLP-03*, 2003.

[32] Philipp Cimiano, Aleksander Pivk, Lars Schmidt-Thieme, and Steffen Staab. Learning taxonomic relations from heterogeneous evidence. In *Proceedings of ECAI-04 Workshop on Ontology Learning and Population*, 2004.

[33] Michael Collins. *Head-Driven Statistical Models for Natural Language Parsing*. PhD thesis, University of Pennsylvania, 1999.

[34] David P. A. Corney, Bernard F. Buxton, William B. Langdon, and David T. Jones. Biorat: extracting biological information from full-length papers. *Bioinformatics*, 20(17):3206–3213, 2004.

[35] Corinna Cortes and Vladimir Vapnik. Support-vector networks. In *Machine Learning*, pages 273–297, 1995.

[36] Mark Craven and Johan Kumlien. Constructing biological knowledge bases by extracting information from text sources. In *ISMB*, pages 77–86. AAAI, 1999.

[37] Nello Cristianini and John Shawe-Taylor. *An Introduction to Support Vector Machines (and Other Kernel- Based Learning Methods)*. Cambridge University Press, 2000.

[38] Montse Cuadros and German Rigau. Assessment of large scale knowledge resources. In *Proceedings of EMNLP-06*, 2006.

[39] Montse Cuadros and German Rigau. SemEval-2007 Task 16: Evaluation of wide coverage knowledge resources. In *Proceedings of SemEval-07*, 2007.

[40] Aron Culotta and Jeffrey Sorensen. Dependency tree kernels for relation extraction. In *Proceedings of ACL-04*, 2004.

[41] David Day, John Aberdeen, Lynette Hirschman, Robyn Kozierok, Patricia Robinson, and Marc Vilain. Mixed-initiative development of language processing systems. In *Proceedings of the Fifth Conference on Applied Natural Language Processing*, 1997.

[42] Manuel de Buenaga Rodríguez, José María Gómez-Hidalgo, and Belén Díaz-Agudo. Using WordNet to complement training information in text categorization. In *Proceedings of RANLP-97*, 1997.

[43] Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. Generating typed dependency parses from phrase structure parses. In *Proceedings of LREC-06*, 2006.

[44] Bart Decadt, Veronique Hoste, Walter Daelemans, and Antal van den Bosch. Gambl, genetic algorithm optimization of memory-based WSD. In *Proceedings of ACL/SIGLEX Senseval-3*, 2004.

[45] George Doddington, Alexis Mitchell, Mark Przybocki, Lance Ramshaw, Stephanie Strassel, and Ralph Weischedel. The Automatic Content Extraction (ACE) Program–Tasks, Data, and Evaluation. In *Proceedings of LREC-04*, 2004.

[46] William Dolan. Word sense ambiguation: Clustering related senses. In *Proceedings of ACL-94*, 1994.

[47] Jason M. Eisner. Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of the COLING-96*, 1996.

[48] Oren Etzioni, Michael Cafarella, Doug Downey, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S. Weld, and Alexander Yates. Unsupervised named-entity extraction from the web: An experimental study. *Artificial Intelligence*, 165(1):91–134, 2005.

[49] Hui Fang. A re-examination of query expansion using lexical resources. In *Proceedings of ACL-08*, 2008.

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

[51] Michael Fleischman and Eduard H. Hovy. Fine grained classification of named entities. In *Proceedings of COLING-02*, 2002.

[52] Michael Fleischman, Eduard H. Hovy, and Abdessamad Echihabi. Offline strategies for online question answering: Answering questions before they are asked. In *Proceedings of ACL-03*, 2003.

[53] Katrin Fundel, Robert Küffner, and Ralf Zimmer. Relex—relation extraction using dependency parse trees. *Bioinformatics*, 23(3):365–371, 2007.

[54] Roxana Girju, Adriana Badulescu, and Dan Moldovan. Automatic discovery of part-whole relations. *Computational Linguistics*, 31(1), 2006.

[55] Roxana C. Girju, Adriana Badulescu, and Dan Moldovan. Learning semantic constraints for the automatic discovery of part-whole relations. In *Proceedings of HLT-03*, 2003.

[56] Claudio Giuliano, Alberto Lavelli, and Lorenza Romano. Exploiting shallow linguistic information for relation extraction from biomedical literature. In *Proceedings of EACL-06*, 2006.

[57] Julio Gonzalo, Felia Verdejo, Irina Chugur, and Juan Cigarran. Indexing with WordNet synsets can improve text retrieval. In *Proceedings of COLING-ACL-98 Workshop on WordNet in NLP Systems*, 1998.

[58] Patrick Hanks. Do word meanings exist? *Computers and the Humanities,* **34***(1-2): 171-177*, 2000.

[59] Donna Harman. The DARPA TIPSTER Project. *ACM SIGIR Forum*, 1992.

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

[61] Marti Hearst. Automated discovery of WordNet relations. In *WordNet: An Electronic Lexical Database*, pages 131–153. MIT Press, 1998.

[62] Don Hindle. Noun classification from predicate-argument structures. In *Proceedings of ACL-90*, 1990.

[63] Graeme Hirst and David St-Onge. Lexical chains as representations of context for the detection and correction of malapropisms. In *WordNet: An Electronic Lexical Database*, pages 305–332. MIT Press, 1998.

[64] Eduard H. Hovy, Mitchell P. Marcus, Martha Palmer, Lance A. Ramshaw, and Ralph Weischedel. Ontonotes: The 90% solution. In *HLT-NAACL-06*, 2006.

[65] Grolier Incorporated. *The Academic American Encyclopedia*. Grolier Electronic Publishing, 1990.

[66] Jay J. Jiang and David W. Conrath. Semantic similarity based on corpus statistics and lexical taxonomy. In *Proceedings of the International Conference on Research in Computational Linguistics*, 1997.

[67] Jing Jiang and ChengXiang Zhai. A systematic exploration of the feature space for relation extraction. In *Proceedings of HLT-NAACL-07*, 2007.

[68] Thorsten Joachims. *Learning to Classify Text Using Support Vector Machines*. PhD thesis, University of Dortmund, 2002.

[69] Rosie Jones, Andrew Mccallum, Kamal Nigam, and Ellen Riloff. Bootstrapping for text learning tasks. In *Proceedings of IJCAI-99 Workshop on Text Mining: Foundations, Techniques and Applications*, 1999.

[70] Nanda Kambhatla. Combining lexical, syntactic, and semantic features with maximum entropy models for extracting relations. In *Proceedings of ACL-04 interactive poster and demonstration sessions*, 2004.

[71] Adam Kilgarriff. I don't believe in word senses. *Computers and the Humanities*, 1997.

[72] Adam Kilgarriff. English lexical sample task description. In *Proceedings of the Senseval-2 workshop*, 2001.

[73] Dan Klein and Christopher D. Manning. Accurate unlexicalized parsing. In *Proceedings of HLT-NAACL-03*, 2003.

[74] Nicholas Kushmerick, Daniel S. Weld, and Robert Doorenbos. Wrapper induction for information extraction. In *Proceedings of IJCAI-97*, 1997.

[75] John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of ICML-01*, 2001.

[76] Ken Lang. Newsweeder: Learning to filter netnews. In *Proceedings of ML-95*, 1995.

[77] Claudia Leacock and Martin Chodorow. Combining local context and WordNet similarity for word sense identification. In *WordNet: An Electronic Lexical Database*, pages 265–283. MIT Press, 1998.

[78] Douglas B. Lenat. Cyc: A large-scale investment in knowledge infrastructure. *Communications of the ACM*, 1995.

[79] Michael Lesk. Automatic sense disambiguation using machine readable dictionaries: How to tell a pine cone from an ice cream cone. In *Proceedings of SIGDOC-86*, 1986.

[80] Beth Levin. English verb classes and alternations: A preliminary investigation. *University of Chicago Press, Chicago, IL*, 1993.

[81] Dekang Lin. Dependency-based evaluation of MINIPAR. In *Proceedings of Workshop on the Evaluation of Parsing Systems, Granada, Spain*, 1998.

[82] Dekang Lin. An information-theoretic definition of similarity. In *Proceedings of ICML-98*, 1998.

[83] Dekang Lin and Patrick Pantel. Discovery of inference rules for question-answering. *Natural Language Engineering*, 7(4):343–360, 2001.

[84] Shuang Liu, Fang Liu, Clement Yu, and Weiyi Meng. An effective approach to document retrieval via utilizing WordNet and recognizing phrases. In *Proceedings of SIGIR-04*, 2004.

[85] Bernardo Magnini and Gabriela Cavaglia. Integrating subject field codes into WordNet. In *Proceedings of LREC-00*, 2000.

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

[87] Trevor Mansuy and Robert J. Hilderman. A characterization of WordNet features in Boolean models for text classification. In *Proceedings of AusDM-06*, 2006.

[88] Diana McCarthy. Relating WordNet senses for word sense disambiguation. In *Proceedings of ACL Workshop on Making Sense of Sense*, 2006.

[89] Ryan McDonald. Extracting relations from unstructured text. UPenn CIS Technical Report: MS-CIS-05-06, 2005.

[90] Ryan McDonald, Fernando Pereira, Seth Kulick, Scott Winters, Yang Jin, and Pete White. Simple algorithms for complex relation extraction with applications to biomedical IE. In *Proceedings of ACL-05*, 2005.

[91] Paul McNamee, Rion Snow, Patrick Schone, and James Mayfield. Learning named entity hyponyms for question answering. In *Proceedings of IJCNLP-08*, 2008.

[92] Rada Mihalcea and Ehsanul Faruque. Senselearner: Minimally supervised word sense disambiguation for all words in open text. In *Proceedings of ACL/SIGLEX Senseval-3*, 2004.

[93] Rada Mihalcea and Dan I. Moldovan. Automatic generation of a coarse grained WordNet. In *Proceedings of NAACL Workshop on WordNet and Other Lexical Resources*, 2001.

[94] George A. Miller and Florentina Hristea. WordNet nouns: Classes and instances. *Computational Linguistics*, 32(1):1–3, March 2006.

[95] George A. Miller, Claudia Leacock, Randee Tengi, and Ross T. Bunke. A semantic concordance. In *Proceedings of HLT-93*, 1993.

[96] Mike Mintz, Steven Bills, Rion Snow, and Daniel Jurafsky. Distant supervision for relation extraction without labeled data. In *Proceedings of ACL-09*, 2009.

[97] Dan I. Moldovan and Rada Mihalcea. Using WordNet and lexical operators to improve internet searches. *IEEE Internet Computing, 4(1):34-43*, 2000.

[98] Alexander A. Morgan, Lynette Hirschman, Marc Colosimo, Alexander S. Yeh, and Jeff B. Colombe. Gene name identification and normalization using a model organism database. *Journal of Biomedical Informatics*, 37(6):396–410, December 2004.

[99] Roberto Navigli. Meaningful clustering of senses helps boost word sense disambiguation performance. In *Proceedings of COLING-ACL-06*, 2006.

[100] Joakim Nivre. *Inductive Dependency Parsing*. Springer, 2006.

[101] Joakim Nivre and Mario Scholz. Deterministic dependency parsing of English text. In *Proceedings of COLING-04*, 2004.

[102] Toshihide Ono, Haretsugu Hishigaki, Akira Tanigami, and Toshihisa Takagi. Automated extraction of information on protein-protein interactions from the biological literature. *Bioinformatics*, 17(1):155–161, 2001.

[103] Martha Palmer, Olga Babko-Malaya, and Hoa Trang Dang. Different sense granularities for different applications. In *Proceedings of Workshop on Scalable Natural Language Understanding*, 2004.

[104] Martha Palmer, Hoa Trang Dang, and Christiane Fellbaum. Making fine-grained and coarse-grained sense distinctions. *Journal of Natural Language Engineering*, 2005.

[105] Patrick Pantel. *Clustering by Committee*. PhD thesis, University of Alberta, 2003.

[106] Patrick Pantel and Dekang Lin. Discovering word senses from text. In *Proceedings of KDD-02*, pages 613–619, 2002.

[107] Patrick Pantel and Marco Pennacchiotti. Espresso: leveraging generic patterns for automatically harvesting semantic relations. In *Proceedings of ACL-06*, 2006.

[108] Patrick Pantel and Deepak Ravichandran. Automatically labeling semantic classes. In *Proceedings of HLT-NAACL-04*, 2004.

[109] Marius Pasca and Sanda M. Harabagiu. The informative role of WordNet in open-domain question answering. In *Proceedings of the NAACL-01 Workshop on WordNet and Other Lexical Resources*, 2001.

[110] Siddharth Patwardhan. Incorporating dictionary and corpus information into a context vector measure of semantic relatedness. Master's thesis, University of Minnesota, Duluth, 2003.

[111] Siddharth Patwardhan and Ellen Riloff. Learning domain-specific information extraction patterns from the web. In *Proceedings of the ACL-06 Workshop on Information Extraction Beyond the Document*, 2006.

[112] Ted Pedersen, Siddharth Patwardhan, and Jason Michelizzi. WordNet::Similarity – measuring the relatedness of concepts. In *Proceedings of NAACL-04*, 2004.

[113] Fernando Pereira, Naftali Tishby, and Lillian Lee. Distributional clustering of English words. In *Proceedings of ACL-93*, 1993.

[114] Wim Peters, Ivonne Peters, and Piek Vossen. Automatic sense clustering in EuroWordNet. In *Proceedings of LREC-98*, 1998.

[115] Andrew Philpot, Eduard H. Hovy, and Patrick Pantel. The Omega Ontology. In *Proceedings of the ONTOLEX Workshop at IJCNLP-05*, 2005.

[116] John C. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *Advances in Large Margin Classifiers*, pages 61–74. MIT Press, 1999.

[117] Hoifung Poon and Pedro Domingos. Joint unsupervised coreference resolution with Markov logic. In *Proceedings of EMNLP-08*, 2008.

[118] Deepak Ravichandran and Eduard H. Hovy. Learning surface text patterns for a question answering system. In *Proceedings of ACL-02*, 2002.

[119] Deepak Ravichandran, Patrick Pantel, and Eduard H. Hovy. Randomized algorithms and NLP: Using locality sensitive hash function for high speed noun clustering. In *Proceedings of ACL-05*, 2005.

[120] Philip Resnik. Using information content to evaluate semantic similarity in a taxonomy. In *Proceedings of IJCAI-95*, 1995.

[121] Philip Resnik and David Yarowsky. Distinguishing systems and distinguishing senses: New evaluation methods for word sense disambiguation. *Natural Language Engineering*, 1999.

[122] Stephen D. Richardson, William B. Dolan, and Lucy Vanderwende. Mindnet: acquiring and structuring semantic information from text. In *Proceedings of COLING-98*, 1998.

[123] Ellen Riloff. Automatically constructing a dictionary for information extraction tasks. In *Proceedings of AAAI-93*, 1993.

[124] Ellen Riloff. Automatically generating extraction patterns from untagged text. In *Proceedings of AAAI-96*, 1996.

[125] Ellen Riloff and Jessica Shepherd. A corpus-based approach for building semantic lexicons. In *Proceedings of EMNLP-97*, 1997.

[126] Thomas C. Rindflesch, Lorraine Tanabe, John N . Weinstein, and Lawrence Hunter. Edgar: Extraction of drugs, genes and relations from the biomedical literature. In *Proceedings of the Pacific Symposium on Biocomputing*, 2000.

[127] Alan Ritter, Stephen Soderland, Doug Downey, and Oren Etzioni. It's a contradiction - no, it's not: A case study using functional relations. In *Proceedings of EMNLP-08*, 2008.

[128] Alan Ritter, Stephen Soderland, and Oren Etzioni. What is this, anyway: Automatic hypernym discovery. In *Proceedings of the 2009 AAAI Spring Symposium on Learning by Reading and Learning to Read*, 2009.

[129] Brian Roark and Eugene Charniak. Noun-phrase co-occurrence statistics for semi-automatic semantic lexicon construction. In *Proceedings of ACL-98*, 1998.

[130] Barbara Rosario and Marti A. Hearst. Multi-way relation classification: Application to protein-protein interactions. In *Proceedings of HLT-EMNLP-05*, 2005.

[131] Gerard Salton and Michael J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, New York, NY, 1983.

[132] Ryan McDonald Sandra Kübler and Joakim Nivre. *Dependency Parsing*. Morgan & Claypool Publishers, 2009.

[133] Patrick Schone, Gary Ciany, Paul McNamee, James Mayfield, and Thomas Smith. QACTIS-based Question Answering at TREC 2005. In *Proceedings of the 14th Text REtrieval Conference*, 2005.

[134] Sam Scott and Stan Matwin. Text classification using WordNet hypernyms. In *Proceedings of COLING-ACL-98 Workshop on Usage of WordNet in Natural Language Processing Systems*, 1998.

[135] Fabrizio Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys*, 2002.

[136] Yusuke Shinyama and Satoshi Sekine. Preemptive information extraction using unrestricted relation discovery. In *Proceedings of HLT-NAACL-06*, 2006.

[137] Rion Snow, Daniel Jurafsky, and Andrew Y. Ng. Learning syntactic patterns for automatic hypernym discovery. In Lawrence K. Saul, Yair Weiss, and Léon Bottou, editors, *NIPS 17*, pages 1297–1304. MIT Press, 2005.

[138] Rion Snow, Daniel Jurafsky, and Andrew Y. Ng. Semantic taxonomy induction from heterogenous evidence. In *Proceedings of COLING-ACL-06*, 2006.

[139] Rion Snow, Sushant Prakash, Daniel Jurafsky, and Andrew Y. Ng. Learning to merge word senses. In *Proceedings of EMNLP-07*, 2007.

[140] Benjamin Snyder and Martha Palmer. The English all-words task. In *Proceedings of the ACL-04 Senseval-3 Workshop*, 2004.

[141] Stephen Soderland and Bhushan Mandhani. Moving from textual relations to ontologized relations. In *AAAI Spring Symposium on Machine Reading*, 2007.

[142] Balaji Srinivasan, Rion Snow, Kim Branson, Vijay S. Pande, Andrew Ng, Russ Altman, and Serafim Batzoglou. Automatic generation of one million structured digital abstracts. In *Proceedings of BCATS-07*, 2007.

[143] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. YAGO - a core of semantic knowledge. In *Proceedings of WWW-07*, 2007.

[144] Beth Sundheim, editor. *Proceedings of MUC-4*, 1992.

[145] Noriko Tomuro. Tree-cut and a lexicon based on systematic polysemy. In *Proceedings of NAACL-01*, 2001.

[146] Peter D. Turney. Mining the web for synonyms: PMI-IR versus LSA on TOEFL. In *Proceedings of ECML-01*, 2001.

[147] Peter D. Turney and Michael L. Littman. Corpus-based learning of analogies and semantic relations. *Machine Learning*, 60(1-3):251–278, 2005.

[148] Vladimir Vapnik. *Statistical Learning Theory.* John Wiley, 1998.

[149] Ellen Voorhees and Hoa Trang Dang. Overview of the TREC 2005 question answering track. In *Notebook Proceedings of the Fourteenth Text REtrieval Conference*, 2005.

[150] Ellen M. Voorhees. Query expansion using lexical-semantic relations. In *Proceedings of SIGIR-94*, 1994.

[151] Mengqiu Wang. A re-examination of dependency path kernels for relation extraction. In *Proceedings of IJCNLP-08*, 2008.

[152] Ben Wellner, Andrew McCallum, Fuchun Peng, and Michael Hay. An integrated, conditional model of information extraction and coreference with application to citation matching. In *Proceedings of AUAI-04*, 2004.

[153] Fei Wu and Daniel S. Weld. Autonomously semantifying Wikipedia. In *Proceedings of CIKM-07*, 2007.

[154] Fei Wu and Daniel S Weld. Automatically refining the Wikipedia infobox ontology. In *Proceedings of WWW-08*, 2008.

[155] Zhibiao Wu and Martha Palmer. Verb semantics and lexical selection. In *Proceedings of ACL-94*, 1994.

[156] David Yarowsky. Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of ACL-95*, 1995.

[157] Alexander Yates and Oren Etzioni. Unsupervised resolution of objects and relations on the web. In *Proceedings of HLT-NAACL-07*, 2007.

[158] Alexander Yates and Oren Etzioni. Unsupervised methods for determining object and relation synonyms on the web. *Journal of Artificial Intelligence Research*, 34:255–296, March 2009.

[159] Deniz Yuret. Some experiments with a Naive Bayes WSD system. In *Proceedings of ACL/SIGLEX Senseval-3*, 2004.

[160] Dmitry Zelenko, Chinatsu Aone, and Anthony Richardella. Kernel methods for relation extraction. *Journal of Machine Learning Research*, 3:1083–1106, 2003.

[161] Min Zhang, Jie Zhang, Jian Su, and Guodong Zhou. A composite kernel to extract relations between entities with both flat and structured features. In *Proceedings of ACL-06*, 2006.

[162] Guodong Zhou, Jian Su, Jie Zhang, and Min Zhang. Exploring various knowledge in relation extraction. In *Proceedings of ACL-05*, 2005.

[163] Guodong Zhou, Jian Su, and Min Zhang. Modeling commonality among related classes in relation extraction. In *Proceedings of ACL-06*, 2006.