Surface Web Semantics for Structured Natural Language Processing

Mohit Bansal

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2015-20
http://www.eecs.berkeley.edu/Pubs/TechRpts/2015/EECS-2015-20.html

May 1, 2015

Surface Web Semantics for Structured Natural Language Processing

by

Mohit Bansal

A dissertation submitted in partial satisfaction of the
requirements for the degree of
Doctor of Philosophy

in

Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Dan Klein, Chair
Professor Marti Hearst
Professor Line Mikkelsen
Professor Nelson Morgan

Fall 2013

Surface Web Semantics for Structured Natural Language Processing

Abstract

Surface Web Semantics for Structured Natural Language Processing

by

Mohit Bansal

Doctor of Philosophy in Computer Science

University of California, Berkeley

Professor Dan Klein, Chair

Research in natural language processing (NLP) is aimed at making machines automatically understand natural language. Incorporating world knowledge semantics is becoming an increasingly crucial requirement for resolving deep, complex decisions in most NLP tasks today, e.g., question answering, syntactic parsing, coreference resolution, and relation extraction. Structured NLP corpora such as treebanks are too small to encode much of this knowledge, so instead, we turn to the vast Web, and access its information via a diverse collection of Web n-gram counts (of size 4 billion, and $\sim$500x larger than Wikipedia). Shallow cues from this large n-gram dataset, when harnessed in a structured learning setting, help reveal deep semantics.

In this thesis, we address various important facets of the semantics problem – from indirect semantics for sentence-level syntactic ambiguities, and semantics as specific knowledge for discourse-level coreference ambiguities, to structured acquisition of semantic taxonomies from text, and fine-grained semantics such as intensity order. These facets represent structured NLP tasks which have a combinatorially large decision space. Hence, in general, we adopt a structured learning approach, incorporating surface Web-based semantic cues as intuitive features on the full space of decisions. The feature weights are then learned automatically based on a discriminative training approach. Empirically, for each facet, we see significant improvements over the corresponding state-of-the-art.

In the first part of this thesis, we show how Web-based features can be powerful cues to resolving complex syntactic ambiguities. We develop surface n-gram features over the full range of syntactic attachments, encoding both lexical affinities as well as paraphrase-based cues to syntactic structure. These features, when encoded into full-scale, discriminative dependency and constituent parsers, correct a range of error types.

In the next part, we address semantic ambiguities in discourse-level coreference resolution, again using Web n-gram features that capture a range of world knowledge cues to hypernymy, semantic compatibility, and semantic context, as well as general lexical co-occurrence. When added to a state-of-the-art coreference baseline, these Web features provide significant improvements on multiple datasets and metrics.

In the third part, we acquire the semantics itself via structured learning of hypernymy taxonomies. We adopt a probabilistic graphical model formulation which incorporates heterogeneous relational evidence about both hypernymy and siblinghood, captured by surface features based on patterns and statistics from Web n-grams and Wikipedia abstracts. Inference is based on loopy belief propagation and spanning tree algorithms. The system is discriminatively trained on WordNet sub-structures using adaptive subgradient stochastic optimization. On the task of reproducing sub-hierarchies of WordNet, this approach achieves substantial error reductions over previous work.

Finally, we discuss a fine-grained semantic facet – intensity order, where the relative ranks of near-synonyms such as *good*, *great*, and *excellent* are predicted using Web statistics of phrases like *good but not excellent*. We employ linear programming to jointly infer the positions on a single scale, such that individual decisions benefit from global information. When ranking English near-synonymous adjectives, this global approach gets substantial improvements over previous work on both pairwise and rank correlation metrics.

To my family, for always being there for me and believing in me.

Contents

List of Figures

List of Tables

Acknowledgments

During my PhD and the process of this thesis, I spent five amazing years in Berkeley and I would like to thank the various folks who made it an enriching and enjoyable journey for me. The list to be thanked is huge, and cannot possibly be covered here completely, but I will try to mention as many people as I can.

I will start with my adviser Dan Klein, the person who shaped me as a researcher and made this thesis possible. His combined expertise in the fields of artificial intelligence, linguistics, and algorithms makes him one of the smartest researchers in the field of natural language processing, and I have been incredibly fortunate to be advised by him. The passion with which he conducts research and inspires his students to do the same made my gradate school experience a fulfilling one. He taught me how to effectively and efficiently formulate, investigate, and evaluate NLP problems, quickly discard an idea if needed, and think about high-impact and long-term research. I also benefitted from his extraordinarily high standards on writing papers and giving presentations. Moreover, Dan is an exceptional teacher, and his classes and students are one of his biggest priorities. I was lucky enough to experience and cultivate the joy of teaching when I was a graduate student instructor for two of his classes. However, I will most admiringly remember Dan for being an all-round mentor – he was always available for healthily discussing, sharing his views, and giving advice on any topic that might concern me from any phase of life.

Having Dan as a mentor had the added bonus of getting to be a part of an awesome research group – the Berkeley NLP group. I learned a lot from the outstanding colleagues in this group, via research collaborations, lively discussions on every possible agenda under the sun, and various fun outings. Taylor Berg-Kirkpatrick, John Blitzer, Alexandre Bouchard-Côté, David Burkett, John DeNero, Greg Durrett, Dave Golland, Aria Haghighi, David Hall, Jonathan Kummerfeld, Percy Liang, Adam Pauls, and Slav Petrov – I would like to earnestly thank each and every one of them for the positive impact on my graduate life.

I also had the privilege of collaborating with some brilliant external researchers – I did my first industrial research internship at Microsoft Research with Chris Quirk and Bob Moore, and it was a rewarding experience. My second internship experience at Google Research, with John DeNero (which fortunately gave me a chance to continue working with him after he graduated) and Dekang Lin, was also fabulous. The fellow interns and other researchers at these places added much joy to the great summers. I also enjoyed collaborating with Gerard de Melo at ICSI Berkeley. Marti Hearst, Line Mikkelsen, and Nelson Morgan deserve a special acknowledgment for serving on my qualifier and dissertation committees, and providing valuable advice and feedback at various stages of my graduate career.

My friends outside work have been a constant source of support and refreshment in my graduate life – Abhijeet, Mobin, Kavitha, Anuj, Debanjan, Aditya, Brinda, Sharanya, Prateek, and the list goes on. Berkeley (and the surrounding San Francisco region), I believe, is one of the best places to pursue an arduous task such as a PhD. This city has played a significant role in this dissertation by keeping me stimulated via a ton of fun activities, delectable cuisines, and scenic places to visit, and I am grateful for this.

Last but definitely one of the most, I would like to acknowledge a deep gratitude towards my family. My parents and my sister have always had unwavering faith in me and provided endless motivation and support throughout this process. My wife, Shikha, has been the most patient, loving, enthusiastic, and supportive companion I could ever hope for on this journey, and she saw me through the thick and thin of the entire process. I cannot imagine having successfully (and more importantly, happily) completed my PhD without her by my side.

Again, I would like to wholeheartedly thank every person who made these five years such a worthwhile experience for me. I will always cherish your support.

Chapter 1

Introduction

1.1 Current State of NLP

Research in natural language processing (NLP) is aimed at making machines understand and generate natural language. There has been a lot of progress in this field. One of the latest successes is the IBM Watson question answering system (Ferrucci et al., 2010), that recently won the game of 'Jeopardy' against the best human players. Search engines today also use similar NLP techniques to return the best results for a query. There have also been great advances in the tasks of machine translation (automatically converting text in one language to another) and speech recognition (automatically converting human speech to text), both of which have become accurate enough for day to day usage on our mobile phones.

We have come a long way as a field, but NLP is still not accurate enough for us to have open-ended, fluent conversations with machines. One of the biggest challenges is resolving the many deep and complex semantic ambiguities that are an inherent property of natural language, and prevalent in most standard NLP tasks such as question answering, syntactic parsing, coreference resolution, relation extraction, textual entailment, etc. For example, the phrase *spent time with family* is ambiguous and has two very different syntactic analyses (or bracketings), and hence, two very different meanings. Similarly, consider the following document:

> *Obama met Jobs to discuss the economy, technology, and education. After the meeting, he signed a bill to introduce [...]*

Here, we need to automatically predict that the correct referent of the pronoun *he* is *Obama* and not *Jobs*, which in turn would need the world knowledge that presidents sign bills, and not CEOs.

Predicting the semantic relationship between the entities themselves, e.g., *rat is a rodent*, also requires resolving various semantic ambiguities. Automatic detection and generation of non-literal language constructions such as metaphors and idioms present us with even bigger challenges. Incorporating missing contextual world knowledge can help resolve many such intricate NLP decisions and this is the main goal of this thesis.

1.2 Incorporating World Knowledge

World knowledge can be a deep and subtle thing, but, in practice, shallow surface cues can go a long way towards robustly resolving various complex semantic ambiguities – if you have enough data and if you mine it well. Structured NLP corpora such as parsing treebanks are too small to encode much of this knowledge, so instead, we turn to the vast Web which contains large amounts of information on almost every topic in the world.

Previous work has effectively demonstrated that counts from large corpora and the Web can help correct errors in a variety of NLP problems. One of the earliest uses of such large corpus counts was for the problem of prepositional phrase (PP) attachment disambiguation (Hindle and Rooth, 1993; Volk, 2001). They formulated the problem as a binary decision task with the configuration where a verb takes a noun complement that is followed by a prepositional-phrase (PP), i.e., *verb* $noun_1$ *preposition* $noun_2$. The binary decision to make is whether the PP *preposition* $noun_2$ attaches to *verb* or $noun_1$. For this, they compared surface co-occurrence counts and probabilities of the competing tuples. For example, to resolve the ambiguity in the tuple *spent time with family*, we compare $P(\textit{with}|\textit{spent})$ versus $P(\textit{with}|\textit{time})$ based on counts in a large corpus, and the former is higher than the latter, which signals that the preposition *with* attaches to the verb *spent*. While this 'affinity' is only a surface correlation, (Volk, 2001) showed that comparing such counts can often correctly resolve tricky PP attachments.

This basic idea led to a great deal of successful work on disambiguating binary PP attachments. Nakov and Hearst (2005b) showed that Web-based *paraphrase* and context cues can further improve PP attachment accuracy in the tuples described above. For example, the existence of reworded phrases like *spent it with* on the Web further add evidence of a verbal attachment for the preposition *with* in the previous example. Other work has exploited Web counts for similar isolated syntactic ambiguities such as those in noun phrase (NP) coordination (Nakov and Hearst, 2005b) and noun compound bracketing (Nakov and Hearst, 2005a; Pitler et al., 2010). In NP coordination, previous work considered configurations of the form $noun_1$ *and/or* $noun_2$ *head-noun*, e.g., *president and chief executive*, where the ambiguity is between the two bracketing's *((president and chief) executive)* versus *(president and (chief executive))*. In noun compound bracketing, a sequence of nouns is considered and the task is to disambiguate the bracketing options, for example, '*liver cell antibody*' can be bracketed as *((liver cell) antibody)* or *(liver (cell antibody))*.

In addition to the syntactic ambiguities above, counts from large corpora have also been successfully used for various other tasks such as machine translation candidate selection, spelling correction, adjective ordering (Lapata and Keller, 2004; Bergsma et al., 2010), multiple problems and subtasks in coreference resolution, e.g., bridging anaphora, other anaphora, definite NP reference, pronoun resolution (Poesio et al., 2004; Markert and Nissim, 2005; Yang et al., 2005; Daumé III and Marcu, 2005; Bergsma and Lin, 2006; Yang and Su, 2007; Kobdani et al., 2011; Rahman and Ng, 2011), relation extraction and taxonomy induction (Lin and Pantel, 2002; Pasca, 2004; Davidov and Rappoport, 2006; Pantel and Pennacchiotti, 2006; Snow et al., 2006; Yamada et al., 2009; Ritter et al., 2009; Hovy et al., 2009; Kozareva and Hovy, 2010; Navigli et al., 2011), etc.[1]

[1]Details of previous work are discussed in the corresponding subsequent chapters.

As an example of using surface Web counts for pronoun resolution, the count of *Obama signed a bill* being higher than that of *Jobs signed a bill* indicates that the pronoun *he* in our example above refers to *Obama*.

1.3 This Thesis: Surface Web Semantics in Structured Learning

Most previous work has either used hand-designed templates for extracting Web statistics, or applied the Web statistics to isolated subtasks or tasks with a relatively small and simple decision space (e.g., PP attachment, NP coordination). The primary contribution and direction of this thesis is to automatically learn all the useful Web-based surface cues, and to use them to resolve complex, deep decisions in end-to-end, full-scale, structured NLP tasks such as syntactic parsing (dependency and constituent), coreference resolution and taxonomy induction. Such tasks involve a combinatorial number of decisions. Hence, in general, we adopt a structured learning approach which allows us to directly work with the full combinatorial space appropriate for the particular problem. The surface Web-based information is incorporated in the learning paradigm as intuitive features on the full space of decisions, and their weights are learned automatically based on a discriminative training approach.

Moreover, in order to harness the information on the large Web without presupposing a deep understanding of all Web text, we use a diverse collection of Web n-gram counts (of size 4 billion, and $\sim$500x larger than Wikipedia). A majority of previous approaches have used search engines to collect counts or the number of page hits (Lapata and Keller, 2004; Nakov and Hearst, 2005b), but these will lead to unstable and irreproducible statistics and results (Kilgarriff, 2007)). Other approaches (e.g., Nakov and Hearst (2008)) adopt post-processing of the top result snippets of a search engine. However, this method also has multiple issues such as daily query limits, speed, quality of post-processed search results, etc. This is why we use an offline, static Web corpus such as the Google n-grams dataset (Brants and Franz, 2006) – which contains English n-grams (n = 1 to 5) and their observed frequency counts, generated from nearly 1 trillion word tokens and 95 billion sentences. This corpus allows us to efficiently access huge amounts of Web-derived information in a compressed way, though in the process it limits us to local, short-distance queries. For this reason, we also use other resources, e.g., Wikipedia abstracts, where essential.

Empirically, we find that surface cues from the large Web n-gram dataset, when harnessed as features in a structured learning setting, help reveal deep semantics and achieve substantial improvements over the previous state-of-the-art.

1.4 Outline and Contributions

The outline of this thesis is as follows. In Chapter 2, we resolve sentence-level syntactic ambiguities via Web n-gram features, based on Bansal and Klein (2011). In Chapter 3, we develop Web n-gram features to address document-level coreference resolution ambiguities, based on Bansal and Klein (2012). Chapter 4 describes structured taxonomy induction with Web-based hypernymy and siblinghood features, based on Bansal et al. (2013). Finally, in Chapter 5, we discuss a fine-grained semantic facet – intensity order, based on De Melo and Bansal (2013).

The specific contributions of this thesis are:

- Web n-gram features (over the full range of syntactic attachments) that exploit affinity and paraphrase cues to address syntactic ambiguities.
- Integration of these features into full-scale, state-of-the-art constituent and dependency parsers via discriminative learning.
- Web n-gram features that exploit co-occurrence, hypernymy, compatibility, and context cues to address discourse-level coreference ambiguities.
- Integration of these features into a full-scale, state-of-the-art coreference resolution system via decision tree classification.
- Acquiring hypernymy relations (as a taxonomy) via a probabilistic graphical model that incorporates heterogeneous relational evidence about hypernymy and siblinghood.
- Web n-gram and Wikipedia features, based on patterns and their statistics, to capture the hypernymy and siblinghood evidence.
- Loopy belief propagation and directed spanning tree algorithms for efficient inference over taxonomy structures, and adaptive subgradient stochastic optimization for discriminative learning.
- Web n-gram intensity patterns (e.g., *good but not great*) and their statistics to compute pairwise intensity ordering scores for near-synonyms such as *good*, *great*, and *excellent*.
- Incorporating these pairwise scores into a mixed integer linear program to globally infer the collective ranks of near-synonyms on a continuous scale.
- Thorough empirical investigation via all the appropriate datasets, metrics, baselines and previous systems for each semantic facet.
- Substantial empirical improvements over the corresponding state-of-the-art previous work for each semantic facet.
- Analysis of the errors resolved, the remaining errors, the features assigned the highest weights by the learning mechanism, etc. for each semantic facet.
- Efficient extraction of all types of statistics from the large Web n-grams corpus using a trie-based batch approach for each semantic facet.

Chapter 2

Semantics for Syntactic Ambiguities

In this chapter, we focus on developing Web-based surface cues for resolving complex, sentence-level syntactic ambiguities.[1] Current state-of-the art syntactic parsers have achieved accuracies in the range of 90% F1 on the standard Penn Treebank (Marcus et al., 1993), but a range of errors remain. From a dependency viewpoint, structural errors can be cast as incorrect attachments, even for constituent (phrase-structure) parsers. For example, in the Berkeley parser (Petrov et al., 2006), about 20% of the errors are prepositional phrase (PP) attachment errors, where a preposition-headed (IN) phrase was assigned an incorrect parent in the implied dependency tree. Figure 2.1 illustrates an example of the classic (canonical) PP attachment ambiguities. Here, the Berkeley parser (solid blue edges) incorrectly attaches *from debt* to the noun phrase *$ 30 billion* whereas the correct attachment (dashed gold edges) is to the verb *raising*. However, there are a range of error types, as shown in Figure 2.2. Here, (a) is a non-canonical PP attachment ambiguity where *by yesterday afternoon* should attach to *had already*, (b) is an NP-internal ambiguity where *half a* should attach to *dozen* and not to *newspapers*, and (c) is an adverb attachment ambiguity, where *just* should modify *fine* and not the verb *'s*.

Resolving many of these errors requires information that is simply not present in the approximately 1M words on which the parser was trained. One way to access more information is to exploit surface counts from large corpora like the Web (Volk, 2001; Lapata and Keller, 2004). For example, the phrase *raising from* is much more frequent on the Web than *$ x billion from*. While this co-occurrence count or 'affinity' is only a surface correlation, Volk (2001) showed that comparing such counts can often correctly resolve tricky PP attachments. This basic idea has led to a good deal of successful work on disambiguating isolated, binary PP attachments. For example, Nakov and Hearst (2005b) showed that looking for *paraphrase* counts, i.e., counts of special context cues, can further improve PP resolution. In this case, the existence of reworded phrases like *raising it from* on the Web also imply a verbal attachment. Still other work has exploited Web counts for other isolated ambiguities, such as NP coordination (Nakov and Hearst, 2005b) and noun-sequence bracketing (Nakov and Hearst, 2005a; Pitler et al., 2010). For example, in (b), *half dozen* is more frequent than *half newspapers*.

[1]The work described in this chapter was originally presented at ACL in 2011 (Bansal and Klein, 2011).

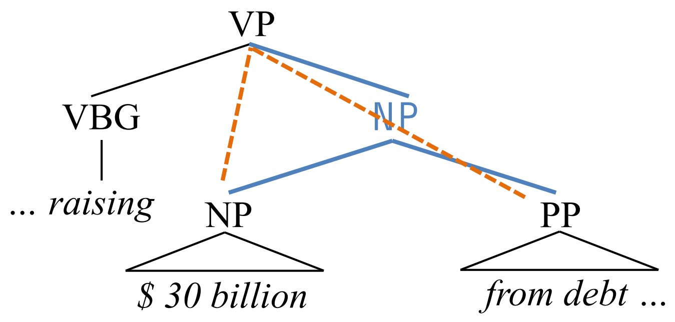


Figure 2.1: A prepositional phrase (PP) attachment error in the parse output of the Berkeley parser (on Penn Treebank). Guess edges are in solid blue, gold edges are in dashed gold and edges common in guess and gold parses are in black.

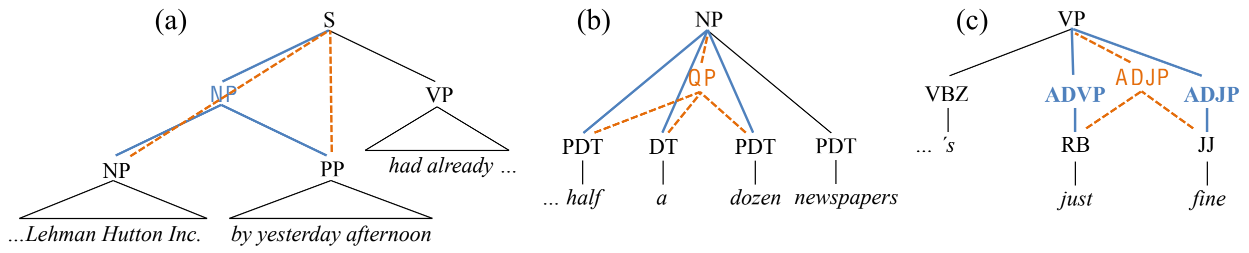


Figure 2.2: Different kinds of attachment errors in the parse output of the Berkeley parser (on Penn Treebank). Guess edges are in solid blue, gold edges are in dashed gold and edges common in guess and gold parses are in black.

In our work, we show how to apply these ideas to all attachments in full-scale parsing. Doing so requires three main issues to be addressed. First, we show how features can be generated for arbitrary head-argument configurations. Affinity features are relatively straightforward, but paraphrase features, which have been hand-developed in the past, are more complex. Second, we integrate our features into full-scale parsing systems. For dependency parsing, we augment the features in the second-order parser of McDonald and Pereira (2006). For constituent parsing, we rerank the output of the Berkeley parser (Petrov et al., 2006). Third, past systems have usually gotten their counts from Web search APIs, which does not scale to quadratically-many attachments in each sentence. Instead, we consider how to efficiently mine a large, static Web-scale corpus –

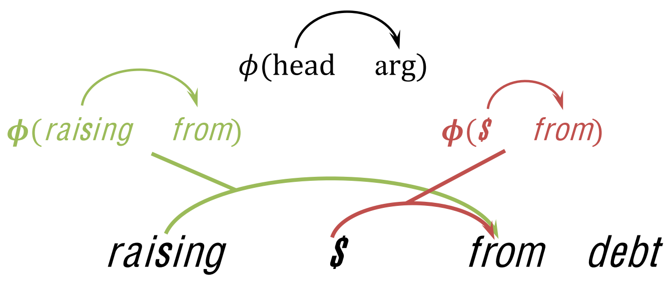


Figure 2.3: Features factored over head-argument pairs or dependency attachments.

the Google n-grams corpus (Brants and Franz, 2006).

Given the success of Web counts for isolated ambiguities, there is relatively little previous research in this direction. The most similar work is Pitler et al. (2010), which use Web-scale n-gram counts for multi-way noun bracketing decisions, though that work considers only sequences of nouns and uses only affinity-based Web features. Yates et al. (2006) use Web counts to filter out certain 'semantically bad' parses from extraction candidate sets but are not concerned with distinguishing amongst top parses. In an important contrast, Koo et al. (2008) smooth the sparseness of lexical features in a discriminative dependency parser by using cluster-based word-senses as intermediate abstractions in addition to POS tags (also see Finkel et al. (2008)). Their work also gives a way to tap into corpora beyond the training data, through cluster membership rather than explicit corpus counts and paraphrases.

This work uses a large Web-scale corpus (Google n-grams) to compute features for the full parsing task. To show end-to-end effectiveness, we incorporate our features into state-of-the-art dependency and constituent parsers. For the dependency case, we can integrate them into the dynamic program of a base parser; we use the discriminatively-trained MST (maximum spanning tree) dependency parser (McDonald et al., 2005; McDonald and Pereira, 2006). Our first-order Web-features give 7.0% relative error reduction over the second-order dependency baseline of McDonald and Pereira (2006). For constituent parsing, we use a reranking framework (Charniak and Johnson, 2005; Collins and Koo, 2005; Collins, 2000) and show 9.2% relative error reduction over the Berkeley parser baseline. In the same framework, we also achieve 3.4% error reduction over the non-local syntactic features used in Huang (2008). Our Web-scale features reduce errors for a range of attachment types. On analyzing the influential features, we find that we not only reproduce features suggested in previous work but also discover a range of new ones.

2.1 Web Features

Structural errors in the output of state-of-the-art parsers, constituent or dependency, can be viewed as attachment errors, examples of which are Figure 2.1 and Figure 2.2.[2] One way to address attachment errors is through features which factor over head-argument pairs, as is standard in the dependency parsing literature (see Figure 2.3). Here, we discuss which Web-count based features $\phi(h, a)$ should fire (i.e., get activated) over a given head-argument pair (we consider the words h and a to be indexed, and so features can be sensitive to their order and distance, as is also standard).

2.1.1 Affinity Features

Affinity statistics, such as lexical co-occurrence counts from large corpora, have been used previously for resolving individual attachments at least as far back as Lauer (1995) for noun-compound bracketing, and later for PP attachment (Volk, 2001; Lapata and Keller, 2004) and coordination ambiguity (Nakov and Hearst, 2005b). The approach of Lauer (1995), for example, would be to take an ambiguous noun sequence like *hydrogen ion exchange* and compare the various counts (or associated conditional probabilities) of n-grams like *hydrogen ion* and *hydrogen exchange*. The attachment with the greater score is chosen. More recently, Pitler et al. (2010) use Web-scale n-grams to compute similar association statistics for longer sequences of nouns.

Our *affinity features* closely follow this basic idea of association statistics. However, because a real parser will not have access to gold-standard knowledge of the competing attachment sites (see Atterer and Schutze (2007)'s criticism of previous work), we must instead compute features for all possible head-argument pairs from our Web corpus. Moreover, when there are only two competing attachment options, one can do things like directly compare two count-based heuristics and choose the larger. Integration into a parser requires features to be functions of single attachments, not pairwise comparisons between alternatives. A learning algorithm can then assign weights to features so that they compare appropriately across parses.

When designing the affinity features, one can use varying specificity. The basic feature is the core adjacency count feature ADJ, which fires for all (h, a) pairs. What is specific to a particular (h, a) is the value of the feature, not its identity. For example, in a naive approach, the value of the ADJ feature might be the count of the query issued to the Web corpus – the 2-gram $q = ha$ or $q = ah$ depending on the order of h and a in the sentence. However, it turns out that there are several problems with this approach. First, rather than a single all-purpose feature like ADJ, the utility of such query counts will vary according to aspects like the parts-of-speech of h and a (because a high adjacency count is not equally informative for all kinds of attachments). Hence, we found that it is better to use more refined affinity features that are specific to each pair of POS tags, i.e. ADJ $\wedge$ POS(h) $\wedge$ POS(a). The values of these POS-specific features, however, are still derived from the same queries as before. Second, using real-valued features did not perform as accurately as first taking the log-value and binning the query-counts[3] and then firing indicator

[2]For constituent parsers, there can be minor tree variations which can result in the same set of induced dependencies, but these are rare in comparison.

[3]For binning, we placed the log-counts into size-5 bins: $b = floor(\log_r(\text{count})/5) * 5$.

features ADJ $\wedge$ POS(h) $\wedge$ POS(a) $\wedge$ b for values of b defined by the query count. Finally, we also conjoin to the preceding features the order of the words h and a as they occur in the sentence, and the (binned) distance between them. These highest specificity features are the final features used in our experiments. Also, for these distance-marked features, wildcards ($\star$) are used in the query $q = h \star a$, where the number of wildcards allowed in the query is proportional to the binned distance between h and a in the sentence. We also include unigram variants of the above features, which are sensitive to only one of the head or argument. Moreover, we add cumulative variants of all the features used, where indicators are fired for all count bins b' up to query count bin b.

2.1.2 Paraphrase Features

In addition to measuring counts of the words present in the sentence, there exist clever ways in which paraphrases and other accidental indicators (e.g., surface-level markers such as capitalization and punctuation) can help resolve specific ambiguities, some of which are discussed in Nakov and Hearst (2005a), Nakov and Hearst (2005b). For example, finding attestations of *eat : spaghetti with sauce* suggests a nominal attachment in *Jean ate spaghetti with sauce*, because separation by a punctuation cue (colon) suggests that *spaghetti with sauce* is a constituent to be bracketed, which in turn means that *with* modifies *spaghetti*. As another example, one clue that the example in Figure 2.1 is a verbal attachment is that the proform paraphrase *raising it from* is commonly attested. This is a clue because unlike common nouns, pronouns cannot take prepositional dependents and hence, the preposition *from* must modify the verb *raising*. Similarly, the attestation of *be noun preposition* suggests nominal attachment.

These paraphrase features hint at the correct attachment decision by looking for Web n-grams with special contexts that reveal syntax superficially. Again, while effective in their isolated disambiguation tasks, past work has been limited by both the range of attachments considered and the need to intuit these special contexts. For instance, frequency of the pattern *The noun preposition* suggests noun attachment and of the pattern *verb adverb preposition* suggests verb attachment for the preposition in the phrase *verb noun preposition*, but these features were not in the manually brainstormed list.

In this work, we automatically generate a large number of paraphrase-style features for arbitrary attachment ambiguities. To induce our list of features, we first mine useful context words. We take each (correct) training dependency relation (h, a) and consider Web n-grams of the form cha, hca, and hac. Aggregating over all h and a (of a given POS pair), we determine which context words c are most frequent in each position. For example, for h = *raising* and a = *from* (see Figure 2.1), we look at Web n-grams of the form *raising c from* and see that one of the most frequent values of c on the Web turns out to be the word *it*.

Once we have collected context words (for each position p in $\{$BEFORE, MIDDLE, AFTER$\}$), we turn each context word c into a collection of features of the form PARA $\wedge$ POS(h) $\wedge$ POS(a) $\wedge$ c $\wedge$ p $\wedge$ dir, where dir is the linear order of the attachment in the sentence. Note that h and a are head and argument words and so actually occur in the sentence, but c is a context word that generally does not. For such features, the queries that determine their values are then of the form cha, hca, and so on. Continuing the previous example, if the test set has a possible attachment of two words

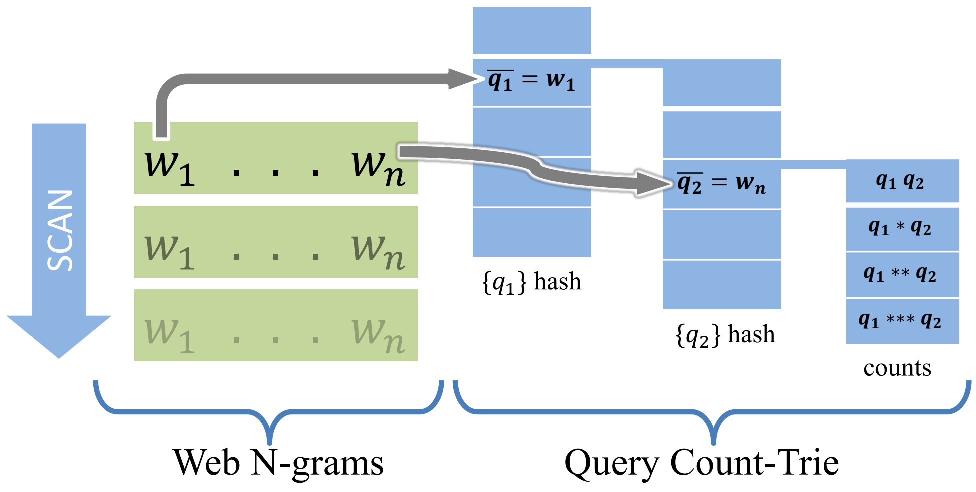


Figure 2.4: Efficient mining of Web n-gram statistics via a trie-based batch approach.

like h = *lowering* and a = *with*, we will fire a feature PARA $\wedge$ VBG $\wedge$ IN $\wedge$ *it* $\wedge$ MIDDLE $\wedge \rightarrow$ with value (indicator bins) set according to the results of the query *lowering it with*. The idea is that if frequent occurrences of *raising it from* indicated a correct attachment between *raising* and *from*, frequent occurrences of *lowering it with* will indicate the correctness of an attachment between *lowering* and *with*. Finally, to handle the cases where no induced context word is helpful, we also construct abstracted versions of these paraphrase features where the context words c are collapsed to their parts-of-speech POS(c), obtained using a unigram-tagger trained on the parser training set. As discussed in Section 2.4, the top features learned by our learning algorithm duplicate the hand-crafted configurations used in previous work (Nakov and Hearst, 2005b) but also add numerous others, and, of course, apply to many more attachment types.

2.2 Efficiently Working with Web n-Grams

Previous approaches have generally used search engines to collect count statistics (Lapata and Keller, 2004; Nakov and Hearst, 2005b; Nakov and Hearst, 2008). Lapata and Keller (2004) uses the number of page hits as the Web-count of the queried n-gram (which is problematic according to Kilgarriff (2007)). Nakov and Hearst (2008) post-processes the first 1000 result snippets. One challenge with this approach is that an external search API is now embedded into the parser, raising issues of both speed and daily query limits, especially if all possible attachments trigger queries. Such methods also create a dependence on the quality and post-processing of the search results, limitations of the query process (for instance, search engines can ignore punctuation (Nakov and Hearst, 2005b)).

Rather than working through a search API (or scraper), we use an offline Web corpus – the Google n-gram corpus (Brants and Franz, 2006) – which contains English n-grams (n = 1 to 5) and their observed frequency counts, generated from nearly 1 trillion word tokens and 95 billion sentences. This corpus allows us to efficiently access huge amounts of Web-derived information in a compressed way, though in the process it limits us to local queries. In particular, we only use counts of n-grams of the form $x \star y$ where the gap length is ≤ 3.

Our system requires the counts from a large collection of these n-gram queries (around 4.5 million). The most basic queries are counts of head-argument pairs in contiguous *h a* and gapped *h* $\star$ *a* configurations.[4] Here, we describe how we process queries of the form (q_1, q_2) with some number of wildcards in between. We first collect all such queries over all trees in preprocessing (so a new test set requires a new query-extraction phase). Next, we exploit a simple but efficient trie-based hashing algorithm to efficiently answer all of them in one pass over the n-grams corpus.

Consider Figure 2.4, which illustrates the data structure which holds our queries. We first create a trie of the queries in the form of a nested hashmap. The key of the outer hashmap is the first word q_1 of the query. The entry for q_1 points to an inner hashmap whose key is the final word q_2 of the query bigram. The values of the inner map is an array of 4 counts, to accumulate each of $(q_1 q_2)$, $(q_1 \star q_2)$, $(q_1 \star\star q_2)$, and $(q_1 \star\star\star q_2)$, respectively. We use k-grams to collect counts of $(q_1 ... q_2)$ with gap length = $k - 2$, i.e. 2-grams to get $count(q_1 q_2)$, 3-grams to get $count(q_1 \star q_2)$ and so on. With this representation of our collection of queries, we go through the Web n-grams (n = 2 to 5) one by one. For an n-gram $w_1 ... w_n$, if the first n-gram word w_1 doesn't occur in the outer hashmap, we move on. If it does match (say $\bar{q}_1 = w_1$), then we look into the inner map for $\bar{q}_1$ and check for the final word w_n. If we have a match, we increment the appropriate query's result value.

In similar ways, we also mine the most frequent words that occur before, in between and after the head and argument query pairs. For example, to collect mid words, we go through the 3-grams $w_1 w_2 w_3$; if w_1 matches $\bar{q}_1$ in the outer hashmap and w_3 occurs in the inner hashmap for $\bar{q}_1$, then we store w_2 and the count of the 3-gram. After the sweep, we sort the context words in decreasing order of count. We also collect unigram counts of the head and argument words by sweeping over the unigrams once.

In this way, our work is linear in the size of the n-gram corpus, but essentially constant in the number of queries. Of course, if the number of queries is expected to be small, such as for a one-off parse of a single sentence, other solutions might be more appropriate; in our case, a large-batch setting, the number of queries was such that this formulation was chosen. Our main experiments (with no parallelization) took 115 minutes to sweep over the 3.8 billion n-grams (n = 1 to 5) to compute the answers to 4.5 million queries, much less than the time required to train the baseline parsers.

2.3 Experiments

Our features are designed to be used in full-sentence parsing rather than for limited decisions about isolated ambiguities. We first integrate our features into a discriminatively-trained dependency

[4]Paraphrase features give situations where we query $\star$ *h a* and *h a* $\star$; these are handled similarly.

	Order 2 Baseline	+ Web features	Relative Error Reduction
Dev (sec 22)	92.1	**92.7**	7.6%
Test (sec 23)	91.4	**92.0**	7.0%

Table 2.1: UAS results for English WSJ dependency parsing. Dev is WSJ section 22 (all sentences) and Test is WSJ section 23 (all sentences). The order 2 baseline represents McDonald and Pereira (2006).

parser, where the integration is more natural and pushes all the way into the underlying dynamic program. We then add them to a constituent parser in a discriminative reranking approach. We also verify that our features contribute on top of standard reranking features. All reported experiments are run on *all* sentences, i.e. without any length limit.

2.3.1 Dependency Parsing

For dependency parsing, we use the discriminatively-trained MSTParser[5], an implementation of first and second order MST (maximum spanning tree) parsing models of McDonald et al. (2005) and McDonald and Pereira (2006). We use the standard splits of Penn Treebank into training (sections 2-21), development (section 22) and test (section 23). We used the 'pennconverter'[6] tool to convert Penn trees from constituent format to dependency format. Following Koo et al. (2008), we used the MXPOST tagger (Ratnaparkhi, 1996) trained on the full training data to provide part-of-speech tags for the development and the test set, and we used 10-way jackknifing to generate tags for the training set.

We added our first-order Web-scale features to the MSTParser system to evaluate improvement over the results of McDonald and Pereira (2006).[7] Table 2.1 shows unlabeled attachments scores (UAS) for their second-order projective parser and the improved numbers resulting from the addition of our Web-scale features. Our first-order Web-scale features show significant improvement even over their non-local *second*-order features. Additionally, our Web-scale features are at least an order of magnitude fewer in number than even their first-order base features. Work such as Smith and Eisner (2008), Martins et al. (2009), Koo and Collins (2010) has been exploring more non-local features for dependency parsing. It will be interesting to see how these features interact with our Web features.

[5]http://sourceforge.net/projects/mstparser

[6]This supersedes 'Penn2Malt' and is available at http://nlp.cs.lth.se/software/treebank_converter. We follow its recommendation to patch WSJ data with NP bracketing by Vadas and Curran (2007).

[7]Their README specifies 'training-k:5 iters:10 loss-type:nopunc decode-type:proj', which we used for all final experiments; we used the faster 'training-k:1 iters:5' setting for most development experiments.

	$k = 1$	$k = 2$	$k = 10$	$k = 25$	$k = 50$	$k = 100$
Dev (sec 22)	90.6	92.3	95.1	95.8	96.2	96.5
Test (sec 23)	90.2	91.8	94.7	95.6	96.1	96.4

Table 2.2: Oracle F1-scores for k-best lists output by Berkeley parser for English WSJ parsing (Dev is section 22 and Test is section 23, all lengths).

2.3.2 Constituent Parsing

We also evaluate the utility of Web-scale features on top of a state-of-the-art constituent parser – the Berkeley parser (Petrov et al., 2006), an unlexicalized phrase-structure parser. Because the underlying parser does not factor along lexical attachments, we instead adopt the discriminative reranking framework, where we generate the top-k candidates from the baseline system and then rerank this k-best list using (generally non-local) features.

Our baseline system is the Berkeley parser, from which we obtain k-best lists for the development set (WSJ section 22) and test set (WSJ section 23) using a grammar trained on all the training data (WSJ sections 2-21).[8] To get k-best lists for the training set, we use 3-fold jackknifing where we train a grammar on 2 folds to get parses for the third fold.[9] The oracle scores of the k-best lists (for different values of k) for the development and test sets are shown in Table 2.2. Based on these results, we used 50-best lists in our experiments. For discriminative learning, we used the averaged perceptron (Collins, 2002; Huang, 2008).

Our core feature is the log conditional likelihood of the underlying parser.[10] All other features are indicator features. First, we add all the Web-scale features as defined above. These features alone achieve a 9.2% relative error reduction. The affinity and paraphrase features contribute about two-fifths and three-fifths of this improvement, respectively. Next, we rerank with only the features (both local and non-local) from Huang (2008), a simplified merge of Charniak and Johnson (2005) and Collins (2000) (here *configurational*). These features alone achieve around the same improvements over the baseline as our Web-scale features, even though they are highly non-local and extensive. Finally, we rerank with both our Web-scale features and the configurational features. When combined, our Web-scale features give a further error reduction of 3.4% over the configurational reranker (and a combined error reduction of 12.2%). All results are shown in Table 2.3. Note that the head words are found using the lexical head rules defined in Collins (1999) and Collins (2003).

[8]Settings: 6 iterations of split and merge with smoothing.

[9]Default: we ran the Berkeley parser in its default 'fast' mode; the output k-best lists are ordered by max-rule-score.

[10]This is output by the flag -confidence. Note that baseline results with just this feature are slightly worse than 1-best results because the k-best lists are generated by max-rule-score. We report both numbers in Table 2.3.

	Dev (sec 22)		Test (sec 23)	
Parsing Model	F1	EX	F1	EX
Baseline (1-best)	90.6	39.4	90.2	37.3
log $p(t\|w)$	90.4	38.9	89.9	37.3
+ Web features	91.6	42.5	91.1	40.6
+ Configurational features	91.8	43.8	91.1	40.6
+ Web + Configurational features	**92.1**	**44.0**	**91.4**	**41.4**

Table 2.3: Parsing results for reranking 50-best lists of Berkeley parser (Dev is WSJ section 22 and Test is WSJ section 23, all lengths).

2.4 Analysis

Table 2.4 shows error counts and relative reductions that our Web features provide over the 2nd-order dependency baseline. While we do see substantial gains for classic prepositional (IN) attachment cases, we see equal or greater error reductions for a range of attachment types. Further, Table 2.5 shows how the total errors break down by gold head. For example, the 12.1% total error reduction for attachments of an IN argument (which includes PPs as well as complementized SBARs) includes many errors where the gold attachments are to both noun and verb heads. Similarly, for an NN-headed argument, the major corrections are for attachments to noun and verb heads, which includes both object-attachment ambiguities and coordination ambiguities.

We next investigate the features that were given high weight by our learning algorithm (in the constituent parsing case). We first threshold features by a minimum training count of 400 to focus on frequently-firing ones (recall that our features are not bilexical indicators and so are quite a bit more frequent). We then sort them by descending (signed) weight.

Table 2.6 shows which affinity features received the highest weights, as well as examples of training set attachments for which the feature fired (for concreteness), suppressing both features involving punctuation and the features' count and distance bins. With the standard caveats that interpreting feature weights in isolation is always to be taken for what it is, the first feature (RB$\rightarrow$IN) indicates that high counts for an adverb occurring adjacent to a preposition (like *back into the spotlight*) is a useful indicator that the adverb actually modifies that preposition. The second row (NN$\rightarrow$IN) indicates that whether a preposition is appropriate to attach to a noun is well captured by how often that preposition follows that noun. The fifth row (VB$\rightarrow$NN) indicates that when considering an NP as the object of a verb, it is a good sign if that NP's head frequently occurs immediately following that verb. All of these features essentially state cases where local surface counts are good indicators of (possibly non-adjacent) attachments.

A subset of paraphrase features, which in the automatically-extracted case don't really correspond to paraphrases at all, are shown in Table 2.7. Here we show features for verbal heads and

Argument Tag	# Attachments	# Correct (Baseline)	# Correct (This Work)	Rel. Error Redn.
NN	5725	5387	5429	12.4
NNP	4043	3780	3804	9.1
IN	4026	3416	3490	12.1
DT	3511	3424	3429	5.8
NNS	2504	2319	2348	15.7
JJ	2472	2310	2329	11.7
CD	1845	1739	1738	-0.9
VBD	1705	1571	1580	6.7
RB	1308	1097	1100	1.4
CC	1000	855	854	-0.7
VB	983	940	945	11.6
TO	868	761	776	14.0
VBN	850	776	786	13.5
VBZ	705	633	629	-5.6
PRP	612	603	606	33.3

Table 2.4: Error reduction for attachments of various child (argument) categories. The columns depict the tag, its total attachments as argument, number of correct ones in baseline (McDonald and Pereira, 2006) and this work, and the relative error reduction. Results are for dependency parsing on the dev set for *iters:5,training-k:1*.

Argument Tag	Relative Error Reduction (%) for Various Parent Tags				
NN	IN: 18,	NN: 23,	VB: 30,	NNP:20,	VBN: 33
IN	NN: 11,	VBD: 11,	NNS: 20,	VB:18,	VBG: 23
NNS	IN: 9,	VBD: 29,	VBP: 21,	VB:15,	CC: 33

Table 2.5: Error reduction for each type of parent attachment for a given child in Table 2.4.

IN arguments. The mid-words m which rank highly are those where the occurrence of hma as an n-gram is a good indicator that a attaches to h (m of course does not have to actually occur in the sentence). Interestingly, the top such features capture exactly the intuition from Nakov and Hearst (2005b), namely that if the verb h and the preposition a occur with a pronoun in between, we have evidence that a attaches to h (it certainly can't attach to the pronoun). However, we also see other indicators that the preposition is selected for by the verb, such as adverbs like *directly*.

As another example of known useful features being learned automatically, Table 2.8 shows

POS$_{head}$	POS$_{arg}$	Example (*head, arg*)
RB	IN	*back* $\rightarrow$ *into*
NN	IN	*review* $\rightarrow$ *of*
NN	DT	*The* $\leftarrow$ *rate*
NNP	IN	*Regulation* $\rightarrow$ *of*
VB	NN	*limit* $\rightarrow$ *access*
VBD	NN	*government* $\leftarrow$ *cleared*
NNP	NNP	*Dean* $\leftarrow$ *Inc*
NN	TO	*ability* $\rightarrow$ *to*
JJ	IN	*active* $\rightarrow$ *for*
NNS	TO	*reasons* $\rightarrow$ *to*
IN	NN	*under* $\rightarrow$ *pressure*
NNS	IN	*reports* $\rightarrow$ *on*
NN	NNP	*Warner* $\leftarrow$ *studio*
NNS	JJ	*few* $\leftarrow$ *plants*

Table 2.6: The highest-weight features (thresholded at a count of 400) of the affinity schema. We list only the head and argument POS and the direction (arrow from head to arg). We omit features involving punctuation.

the previous-context-word paraphrase features for a noun head and preposition argument (N $\rightarrow$ IN). Nakov and Hearst (2005b) suggested that the attestation of *be* N IN is a good indicator of attachment to the noun (the IN cannot generally attach to forms of auxiliaries). One such feature occurs on this top list – for the context word *have* – and others occur farther down. We also find their surface marker / punctuation cues of *:* and *,* preceding the noun. However, we additionally find other cues, most notably that if the N IN sequence occurs following a capitalized determiner, it tends to indicate a nominal attachment (in the n-gram, the preposition cannot attach leftward to anything else because of the beginning of the sentence).

In Table 2.9, we see the top-weight paraphrase features that had a conjunction as a middle-word cue. These features essentially say that if two heads w_1 and w_2 occur in the direct coordination n-gram w_1 *and* w_2, then they are good heads to coordinate (coordination unfortunately looks the same as complementation or modification to a basic dependency model). These features are relevant to a range of coordination ambiguities.

Finally, Table 2.10 depicts the high-weight, high-count general paraphrase-cue features for arbitrary head and argument categories, with those shown in previous tables suppressed. Again, many interpretable features appear. For example, the top entry (*the* JJ NNS) shows that when considering attaching an adjective a to a noun h, it is a good sign if the trigram *the a h* is frequent – in that trigram, the adjective attaches to the noun. The second entry (NN - NN) shows that one noun

POS$_{head}$	middle-word	POS$_{arg}$	Example (*head, arg*)
VBN	*this*	IN	*learned, from*
VB	*this*	IN	*publish, in*
VBG	*him*	IN	*using, as*
VBG	*them*	IN	*joining, in*
VBD	*directly*	IN	*converted, into*
VBD	*held*	IN	*was, in*
VBN	*jointly*	IN	*offered, by*
VBZ	*it*	IN	*passes, in*
VBG	*only*	IN	*consisting, of*
VBN	*primarily*	IN	*developed, for*
VB	*us*	IN	*exempt, from*
VBG	*this*	IN	*using, as*
VBD	*more*	IN	*looked, like*
VB	*here*	IN	*stay, for*
VBN	*themselves*	IN	*launched, into*
VBG	*down*	IN	*lying, on*

Table 2.7: The highest-weight features (thresholded at a count of 400) of the mid-word schema for a verb head and preposition argument (with head on left of argument).

is a good modifier of another if they frequently appear together hyphenated (another punctuation-based cue mentioned in previous work on noun bracketing, see Nakov and Hearst (2005a)). While they were motivated on separate grounds, these features can also compensate for inapplicability of the affinity features. For example, the third entry (VBD *this* NN) is a case where even if the head (a VBD like *adopted*) actually selects strongly for the argument (a NN like *plan*), the bigram *adopted plan* may not be as frequent as expected, because it requires a determiner in its minimal analogous form *adopted the plan*.

before-word	POS$_{head}$	POS$_{arg}$	Example (*head, arg*)
second	NN	IN	*season, in*
The	NN	IN	*role, of*
strong	NN	IN	*background, in*
our	NNS	IN	*representatives, in*
any	NNS	IN	*rights, against*
A	NN	IN	*review, of*
:	NNS	IN	*Results, in*
three	NNS	IN	*years, in*
In	NN	IN	*return, for*
no	NN	IN	*argument, about*
current	NN	IN	*head, of*
no	NNS	IN	*plans, for*
public	NN	IN	*appearance, at*
from	NNS	IN	*sales, of*
net	NN	IN	*revenue, of*
,	NNS	IN	*names, of*
you	NN	IN	*leave, in*
have	NN	IN	*time, for*
some	NN	IN	*money, for*
annual	NNS	IN	*reports, on*

Table 2.8: The highest-weight features (thresholded at a count of 400) of the before-word schema for a noun head and preposition argument (with head on left of argument).

POS$_{head}$	middle-CC	POS$_{arg}$	Example (*head, arg*)
NNS	*and*	NNS	*purchases, sales*
VB	*and*	VB	*buy, sell*
NN	*and*	NN	*president, officer*
NN	*and*	NNS	*public, media*
VBD	*and*	VBD	*said, added*
VBZ	*and*	VBZ	*makes, distributes*
JJ	*and*	JJ	*deep, lasting*
IN	*and*	IN	*before, during*
VBD	*and*	RB	*named, now*
VBP	*and*	VBP	*offer, need*

Table 2.9: The highest-weight features (thresholded at a count of 400) of the mid-word schema where the mid-word was a conjunction. For variety, for a given head-argument POS pair, we only list features corresponding to the *and* conjunction and $h \rightarrow a$ direction.

POS$_h$	POS$_a$	middle/before-word		Example (*h, a*)
NNS	JJ	b =	*the*	*other* $\leftarrow$ *things*
NN	NN	m =	-	*auto* $\leftarrow$ *maker*
VBD	NN	m =	*this*	*adopted* $\rightarrow$ *plan*
NNS	NN	b =	*of*	*computer* $\leftarrow$ *products*
NN	DT	m =	*current*	*the* $\leftarrow$ *proposal*
VBG	IN	b =	*of*	*going* $\rightarrow$ *into*
NNS	IN	m =	”	*clusters* $\rightarrow$ *of*
IN	NN	m =	*your*	*In* $\rightarrow$ *review*
TO	VB	b =	*used*	*to* $\rightarrow$ *ease*
VBZ	NN	m =	*that*	*issue* $\leftarrow$ *has*
IN	NNS	m =	*two*	*than* $\rightarrow$ *minutes*
IN	NN	b =	*used*	*as* $\rightarrow$ *tool*
IN	VBD	m =	*they*	*since* $\rightarrow$ *were*
VB	TO	b =	*will*	*fail* $\rightarrow$ *to*

Table 2.10: The high-weight high-count (thresholded at a count of 2000) general features of the mid and before paraphrase schema (examples show head and arg in linear order with arrow from head to arg).

Chapter 3

Semantics for Coreference Ambiguities

In this chapter, we shift our focus from developing Web-based surface cues for sentence-level ambiguities to cues for document-level, discourse-based ambiguities in coreference resolution.[1] Many of the most difficult ambiguities in coreference resolution are semantic in nature. For instance, consider the following example:

When Obama met Jobs, the president discussed the economy, technology, and education. His election campaign is expected to [...]

For resolving coreference in this example, a system would benefit from the world knowledge that *Obama* is *the president*. Also, to resolve the pronoun *his* to the correct antecedent *Obama*, we can use the knowledge that *Obama* has an *election campaign* while *Jobs* does not. Such ambiguities are difficult to resolve on purely syntactic or configurational grounds.

There have been multiple previous systems that incorporate some form of world knowledge in coreference resolution tasks. Most work (Poesio et al., 2004; Markert and Nissim, 2005; Yang et al., 2005; Bergsma and Lin, 2006) addresses special cases and subtasks such as bridging anaphora, other anaphora, definite NP reference, and pronoun resolution, computing semantic compatibility via Web-hits and counts from large corpora. There is also work on end-to-end coreference resolution that uses large noun-similarity lists (Daumé III and Marcu, 2005) or structured knowledge bases such as Wikipedia (Yang and Su, 2007; Haghighi and Klein, 2009; Kobdani et al., 2011) and YAGO (Rahman and Ng, 2011). However, such structured knowledge bases are of limited scope, and, while Haghighi and Klein (2010) self-acquires knowledge about coreference, it does so only via reference constructions and on a limited scale.

In our work, we look to the Web for broader if shallower sources of semantics. In order to harness the information on the Web without presupposing a deep understanding of all Web text, we instead turn to a diverse collection of Web n-gram counts (Brants and Franz, 2006) which, in aggregate, contain diffuse and indirect, but often robust, cues to reference. For example, we can collect

[1]The work described in this chapter was originally presented at ACL in 2012 (Bansal and Klein, 2012).

the co-occurrence statistics of an anaphor with various candidate antecedents to judge relative surface affinities (i.e., (*Obama*, *president*) versus (*Jobs*, *president*)). We can also count co-occurrence statistics of competing antecedents (with surrounding context) when placed in the position of an anaphoric pronoun (i.e., *Obama's election campaign* versus *Jobs' election campaign*).

All of our features begin with a pair of head words (corresponding to the lexical heads of constituents in syntax, defined in Collins (1999) and Collins (2003)) from candidate mention pairs and compute statistics derived from various potentially informative queries' counts. We explore five major categories of semantically informative Web features, based on (1) general lexical affinities (via generic co-occurrence statistics), (2) lexical relations via Hearst-style hypernymy patterns (Hearst, 1992), (3) similarity of entity-based context (e.g., common values of y for which *h is a y* is attested), (4) matches of distributional soft cluster ids, and (5) attested substitutions of candidate antecedents in the context of a pronominal anaphor.

We first describe a strong baseline consisting of the mention-pair model of the Reconcile system (Stoyanov et al., 2009; Stoyanov et al., 2010) using a decision tree (DT) as its pairwise classifier. To this baseline system, we add our suite of features in turn, each class of features providing substantial gains. Altogether, our final system achieves state-of-the-art results on end-to-end coreference resolution (with automatically detected system mentions) on multiple data sets (ACE 2004 and ACE 2005) and metrics (MUC and B^3), with significant improvements over the Reconcile DT baseline and over the results of Haghighi and Klein (2010).

3.1 Baseline System

Before describing our semantic Web features, we first describe our baseline. The core inference and features come from the Reconcile package (Stoyanov et al., 2009; Stoyanov et al., 2010), with modifications described below. Our baseline differs most substantially from Stoyanov et al. (2009) in using a decision tree classifier rather than an averaged linear perceptron.

3.1.1 Reconcile

Reconcile is one of the best implementations of the *mention-pair* model (Soon et al., 2001) of coreference resolution. The mention-pair model relies on a pairwise function to determine whether or not two mentions are coreferent. Pairwise predictions are then consolidated by transitive closure (or some other clustering method) to form the final set of coreference clusters (chains). While our Web features could be adapted to entity-mention systems, their current form was most directly applicable to the mention-pair approach, making Reconcile a particularly well-suited platform for this investigation.

The Reconcile system provides baseline features, learning mechanisms, and resolution procedures that already achieve near state-of-the-art results on multiple popular datasets using multiple standard metrics. It includes over 80 core features that exploit various automatically generated annotations such as named entity tags, syntactic parses, and WordNet classes, inspired by Soon et al. (2001), Ng and Cardie (2002), and Bengtson and Roth (2008). The Reconcile system also facilitates standardized empirical evaluation to past work.[2]

In this work, we develop a suite of simple semantic Web features based on pairs of mention headwords which stack with the default Reconcile features to surpass past state-of-the-art results.

3.1.2 Decision Tree Classifier

Among the various learning algorithms that Reconcile supports, we chose the decision tree classifier, available in Weka (Hall et al., 2009) as J48, an open source Java implementation of the C4.5 algorithm of Quinlan (1993).

The C4.5 algorithm builds decision trees by incrementally maximizing information gain. The training data is a set of already classified samples, where each sample is a vector of attributes or features. At each node of the tree, C4.5 splits the data on an attribute that most effectively splits its set of samples into more ordered subsets, and then recurses on these smaller subsets. The decision tree can then be used to classify a new sample by following a path from the root downward based on the attribute values of the sample.

We find the decision tree classifier to work better than the default averaged perceptron (used by Stoyanov et al. (2009)), on multiple datasets using multiple metrics (see Section 3.3.3). Many advantages have been claimed for decision tree classifiers, including interpretability and robustness. However, we suspect that the aspect most relevant to our case is that decision trees can capture non-linear interactions between features. For example, recency is very important for pronoun reference but much less so for nominal reference.

3.2 Web Features

Our Web features for coreference resolution are simple and capture a range of diffuse world knowledge. Given a mention pair, we use the head finder in Reconcile to find the lexical heads of both mentions (for example, the head of *the Palestinian territories* is the word *territories*). Next, we take each headword pair (h_1, h_2) and compute various Web-count functions on it that can signal whether or not this mention pair is coreferent.

As the source of Web information, we use the Google n-grams corpus (Brants and Franz, 2006) which contains English n-grams ($n = 1$ to 5) and their Web frequency counts, derived from nearly 1 trillion word tokens and 95 billion sentences. Because we have many queries that must be run against this corpus, we apply the trie-based hashing algorithm described in Chapter 2 (Section 2.2) to efficiently answer all of them in one pass over it. The features that require word clusters (Section 3.2.4) use the output of Lin et al. (2010).[3]

We describe our five types of features in turn. The first four types are most intuitive for mention pairs where both members are non-pronominal, but, aside from the general co-occurrence

[2]We use the default configuration settings of Reconcile (Stoyanov et al., 2010) unless mentioned otherwise.

[3]These clusters are derived from the V2 Google n-grams corpus. The V2 corpus itself is not publicly available, but the clusters are available at `http://old-site.clsp.jhu.edu/~sbergsma/PhrasalClusters`

group, helped for all mention pair types. The fifth feature group applies only to pairs in which the anaphor is a pronoun but the antecedent is a non-pronoun. Related work for each feature category is discussed inline.

3.2.1 General Co-occurrence

These features capture co-occurrence statistics of the two headwords, i.e., how often h_1 and h_2 are seen adjacent or nearly adjacent on the Web. This count can be a useful coreference signal because, in general, mentions referring to the same entity will co-occur more frequently (in large corpora) than those that do not. Using the n-grams corpus (for n = 1 to 5), we collect co-occurrence Web-counts by allowing a varying number of wildcards between h_1 and h_2 in the query. The co-occurrence value is:

$$bin\left(\log_{10}\left(\frac{c_{12}}{c_1 \cdot c_2}\right)\right)$$

where

$$\begin{aligned} c_{12} &= count(\text{“}h_1 \star h_2\text{”}) \\ &\quad + count(\text{“}h_1 \star \star h_2\text{”}) \\ &\quad + count(\text{“}h_1 \star \star \star h_2\text{”}), \\ c_1 &= count(\text{“}h_1\text{”}), \text{and} \\ c_2 &= count(\text{“}h_2\text{”}). \end{aligned}$$

We normalize the overall co-occurrence count of the headword pair c_{12} by the unigram counts of the individual headwords c_1 and c_2, so that high-frequency headwords do not unfairly get a high feature value (this is similar to computing scaled mutual information MI (Church and Hanks, 1989)).[4] This normalized value is quantized by taking its $\log_{10}$ and binning (by dividing into fixed-sized intervals and rounding off). The actual feature that fires is an indicator of which quantized bin the query produced. As a real example from our development set, the co-occurrence count c_{12} for the headword pair (*leader, president*) is 11383, while it is only 95 for the headword pair (*voter, president*); after normalization and $\log_{10}$, the values are -10.9 and -12.0, respectively.

These kinds of general Web co-occurrence statistics have been previously used for other NLP tasks (via supervised and unsupervised methods) such as adjective ordering, spelling correction, verb part-of-speech disambiguation, prepositional phrase attachment disambiguation, noun compound bracketing, and syntactic parsing (Nakov and Hearst, 2005a; Nakov and Hearst, 2005b; Bergsma et al., 2010; Pitler et al., 2010; Bansal and Klein, 2011). In coreference, similar word-association scores were used by Kobdani et al. (2011), but their scores were computed using Wikipedia and employed in a bootstrapping-based self-training framework.

[4]We also tried adding $count(\text{“}h1\ h2\text{”})$ to c_{12} but this decreases performance, perhaps because truly adjacent occurrences are often not grammatical.

3.2.2 Hypernymy Co-occurrence

These features capture templated co-occurrence of the two headwords h_1 and h_2 in the Web-scale corpus. Here, we only collect statistics of the headwords co-occurring with a generalized Hearst-style hypernymy pattern (Hearst, 1992) in between. Hypernymy patterns capture various lexical semantic relations between items. For example, seeing *X is a Y* or *X and other Y* indicates hypernymy and also tends to cue coreference. The specific patterns we use are:

- h_1 $\{$*is* | *are* | *was* | *were*$\}$ $\{$*a* | *an* | *the*$\}$*?* h_2
- h_1 $\{$*and* | *or*$\}$ $\{$*other* | *the other* | *another*$\}$ h_2
- h_1 *other than* $\{$*a* | *an* | *the*$\}$*?* h_2
- h_1 *such as* $\{$*a* | *an* | *the*$\}$*?* h_2
- h_1 *, including* $\{$*a* | *an* | *the*$\}$*?* h_2
- h_1 *, especially* $\{$*a* | *an* | *the*$\}$*?* h_2
- h_1 *of* $\{$*the*| *all*$\}$*?* h_2

For this feature, we again use a quantized normalized count as in Section 3.2.1, but c_{12} here is restricted to n-grams where one of the above patterns occurs in between the headwords. We did not allow wildcards in between the headwords and the hypernymy patterns because this introduced a significant amount of noise. Also, we do not constrain the order of h_1 and h_2 for these features because the hypernymy patterns can hold for either direction of coreference.[5] As a real example from our development set, the c_{12} count for the headword pair (*leader, president*) is 752, while for (*voter, president*), it is 0.

Hypernymy-based semantic compatibility for coreference is intuitive and has been explored in varying forms by previous work. Modjeska et al. (2003), Markert et al. (2003), Poesio et al. (2004), and Markert and Nissim (2005) employ a subset of the Hearst-style patterns we list above and search engine hits for the subtasks of bridging anaphora, other-anaphora, and definite noun phrase coreference resolution. Others such as Daumé III and Marcu (2005), Haghighi and Klein (2009), and Rahman and Ng (2011) use similar relations, e.g., ISA, appositives, predicate-nominative, TYPE, and MEANS, to extract compatibility statistics from noun-similarity lists (Ravichandran et al., 2005), Wikipedia, YAGO (Suchanek et al., 2007), and FrameNet (Baker et al., 1998). Yang and Su (2007) use Wikipedia and a set of coreferential NP seed pairs to automatically extract semantic patterns via bootstrapping, which are then used as features in a learning setup. Instead of extracting patterns from the training data, we use all the above patterns, which helps us generalize to new datasets for end-to-end coreference resolution (see Section 3.3.3).

[5]Two minor variants not listed above are h_1 *including* h_2 and h_1 *especially* h_2.

3.2.3 Entity-based Context Compatibility

For each headword h, we first collect context *seeds* y using the pattern

- *h* $\{$*is* | *are* | *was* | *were*$\}$ $\{$*a* | *an* | *the*$\}$*? y*

taking seeds y in order of decreasing Web count. The corresponding ordered *seed list* $Y = \{y\}$ gives us useful information about the headword's entity type. For example, for $h = president$, the top 30 seeds (and their parts of speech) include important cues such as *president is elected* (verb), *president is authorized* (verb), *president is responsible* (adjective), *president is the chief* (adjective), *president is above* (preposition), and *president is the head* (noun).

Matches in the seed lists of two headwords can be a strong signal that they are coreferent. For example, in the top 30 seed lists for the headword pair (*leader, president*), we get matches including *elected*, *responsible*, and *expected.* To capture this effect, we create a feature that indicates whether there is a match in the top k seeds of the two headwords (where k is a hyperparameter to tune).

We create another feature that indicates whether the dominant parts of speech in the seed lists matches for the headword pair. We first collect the POS tags (using length 2 character prefixes to indicate coarse parts of speech) of the seeds matched in the top k' seed lists of the two headwords, where k' is another hyperparameter to tune. If the dominant tags match and are in a small list of important tags ({JJ, NN, RB, VB}), we fire an indicator feature specifying the matched tag, otherwise we fire a *no-match* indicator. To obtain POS tags for the seeds, we use a unigram-based POS tagger trained on the WSJ treebank training set.

3.2.4 Distributional Clustering

The distributional hypothesis of Harris (1954) says that words that occur in similar contexts tend to have a similar linguistic behavior. Here, we design features with the idea that this hypothesis extends to reference: mentions occurring in similar contexts in large document sets such as the Web tend to be compatible for coreference. Instead of collecting the contexts of each mention and creating sparse features from them, we use Web-scale distributional clustering to summarize compatibility.

Specifically, we begin with the phrase-based clusters from Lin et al. (2010), which were created using the Google n-grams V2 corpus. These clusters come from distributional K-Means clustering (with $K = 1000$) on phrases, using the n-gram context as features. The cluster data contains almost 10 million phrases and their *soft* cluster memberships. Up to twenty cluster ids with the highest centroid similarities are included for each phrase in this dataset (Lin et al., 2010).

Our cluster-based features assume that if the headwords of the two mentions have matches in their cluster id lists, then they are more compatible for coreference. We check the match of not just the top 1 cluster ids, but also farther down in the 20 sized lists because, as discussed in Lin and Wu (2009), the soft cluster assignments often reveal different senses of a word. However, we also assume that higher-ranked matches tend to imply closer meanings. To this end, we fire a feature indicating the value $bin(i + j)$, where i and j are the *earliest match* positions in the cluster id lists of h_1 and h_2. Binning here means that match positions in a close range generally trigger the same feature.

Recent previous work has used clustering information to improve the performance of supervised NLP tasks such as named entity recognition (NER) and query classification (Lin and Wu, 2009), and dependency-based syntactic parsing (Koo et al., 2008). However, to our knowledge, the only related work in coreference resolution is Daumé III and Marcu (2005), who use class-based features derived from a large Web corpus via a clustering process described in Ravichandran et al. (2005).

3.2.5 Pronoun Context Compatibility

Our last feature category specifically addresses pronoun reference, for cases when the anaphoric mention NP_2 (and hence its headword h_2) is a pronoun, while the candidate antecedent mention NP_1 (and hence its headword h_1) is not. For such a headword pair (h_1, h_2), the idea is to substitute the non-pronoun h_1 into h_2's position and see whether the result is attested on the Web.

If the anaphoric pronominal mention is h_2 and its sentential context is *l' l* h_2 *r r'*, then the substituted phrase will be *l' l* h_1 *r r'*. Possessive pronouns are replaced with an additional apostrophe, i.e., h_1 *'s*. We also use features (e.g., R1Gap) that allow wildcards ($\star$) in between the headword and the context when collecting Web-counts, in order to allow for determiners and other filler words. High Web counts of substituted phrases tend to indicate semantic compatibility. Perhaps unsurprisingly for English, only the right context was useful in this capacity. We chose the following three context types, based on performance on a development set:

- h_1 *r* (R1)
- h_1 *r r'* (R2)
- $h_1 \star$ *r* (R1Gap)

As an example of the R1Gap feature, if the anaphor h_2 + context is *his victory* and one candidate antecedent h_1 is *Bush*, then we compute the normalized value

$$\frac{count(\text{“}Bush\,'s\ \star\ victory\text{”})}{count(\text{“}\star\ 's\ \star\ victory\text{”})count(\text{“}Bush\text{”})}$$

In general, we compute

$$\frac{count(\text{“}h_1\,'s\ \star\ r\text{”})}{count(\text{“}\star\ 's\ \star\ r\text{”})count(\text{“}h_1\text{”})}$$

Dataset	# docs	dev split	test split	# mentions	# chains
ACE04	128	63/27	90/38	3037	1332
ACE05	81	40/17	57/24	1991	775
ACE05-ALL	599	337/145	482/117	9217	3050

Table 3.1: Dataset characteristics – the total number of documents, the train/test split during development, the train/test split during testing, the number of gold mentions in the test split, the number of coreference chains in the test split.

The final feature value is again a normalized count converted to $\log_{10}$ and then binned. Normalization helps us with two kinds of balancing. First, we divide by the count of the antecedent so that when choosing the best antecedent for a fixed anaphor, we are not biased towards more frequently occurring antecedents. Second, we divide by the count of the context so that across anaphora, an anaphor with rarer context does not get smaller values (for all its candidate antecedents) than another anaphor with a more common context. We have three separate features for the R1, R2, and R1Gap context types. We tune a separate bin-size hyperparameter for each of these three features.

These pronoun resolution features are similar to selectional preference work by Yang et al. (2005) and Bergsma and Lin (2006), who compute semantic compatibility for pronouns in specific syntactic relationships such as possessive-noun, subject-verb, and verb-object, and in automatically bootstrapped syntactic paths, respectively. In our case, we directly use the general context of any pronominal anaphor to find its most compatible antecedent.

Note that all our above features are designed to be non-sparse by firing indicators of the quantized Web statistics and not the lexical- or class-based identities of the mention pair. This keeps the total number of features small, which is important for the relatively small datasets used for coreference resolution. We go from around 100 features in the Reconcile baseline to around 165 features after adding all our Web features.

3.3 Experiments

3.3.1 Data

We show results on three popular and large coreference resolution data sets – the ACE04, ACE05, and ACE05-ALL datasets from the ACE Program (NIST, 2004). In ACE04 and ACE05, we have only the newswire portion (of the original ACE 2004 and 2005 training sets) and use the standard train/test splits reported in Stoyanov et al. (2009) and Haghighi and Klein (2010). In ACE05-ALL, we have the full ACE 2005 training set and use the standard train/test splits reported in Rahman and Ng (2009) and Haghighi and Klein (2010). Note that most previous work does not report (or need) a standard development set; hence, for tuning our features and its hyper-parameters, we

	MUC			B^3		
Feature	P	R	F1	P	R	F1
Averaged Perceptron	69.0	63.1	65.9	82.2	69.9	75.5
Decision Tree	**80.9**	61.0	69.5	**89.5**	69.0	77.9
+ General Co-occurence	79.8	62.1	69.8	88.7	69.8	78.1
+ Hypernymy Co-occurence	80.0	62.3	70.0	89.1	70.1	78.5
+ Entity-based Context Compatibility	79.4	63.2	70.4	88.1	70.9	78.6
+ Distributional Clustering	79.5	63.6	70.7	87.9	71.2	78.6
+ Pronoun Context Compatibility	79.9	**64.3**	**71.3**	88.0	**71.6**	**79.0**

Table 3.2: Incremental results for the Web features on the ACE04 development set: the averaged perceptron baseline, the decision tree baseline, and the +Feature rows showing the effect of adding a particular feature incrementally (not in isolation) to the decision tree baseline. The feature categories correspond to those described in Section 3.2.

randomly split the original training data into a training and development set with a 70/30 ratio (and then use the full original training set during testing). Details of the corpora are shown in Table 3.1. Note that the development set is used only for ACE04, because for ACE05, and ACE05-ALL, we directly test using the features tuned on ACE04.

Details of the Web-scale corpora used for extracting features are discussed in Section 3.2.

3.3.2 Evaluation Metrics

We evaluated our work on both MUC (Vilain et al., 1995) and B^3 (Bagga and Baldwin, 1998). Both scorers are available in the Reconcile infrastructure. MUC measures how many predicted clusters need to be merged to cover the true gold clusters. B^3 computes precision and recall for each mention by computing the intersection of its predicted and gold cluster and dividing by the size of the predicted and gold cluster, respectively. It is well known (Recasens and Hovy, 2010; Ng, 2010; Kobdani et al., 2011) that MUC is biased towards large clusters (chains) whereas B^3 is biased towards singleton clusters. Therefore, for a more balanced evaluation, we show improvements on both metrics simultaneously. Note that B^3 has two versions which handle twinless (spurious) mentions in different ways (see Stoyanov et al. (2009) for details). We use the B^3All version, unless mentioned otherwise.

3.3.3 Results

We start with the Reconcile baseline but employ the decision tree (DT) classifier, because it has significantly better performance than the default averaged perceptron classifier used in Stoyanov et al. (2009).[6] Table 3.2 compares the baseline perceptron results to the DT results and then shows the incremental addition of the Web features to the DT baseline (on the ACE04 development set).

The DT classifier, in general, is precision-biased. The Web features somewhat balance this by increasing the recall and decreasing precision to a lesser extent, improving overall F1. Each feature type incrementally increases both MUC and B^3 F1-measures, showing that they are not taking advantage of any bias of either metric. The incremental improvements also show that each Web feature type brings in some additional benefit over the information already present in the Reconcile baseline, which includes alias, animacy, named entity, and WordNet class / sense information.[7]

Table 3.3 shows our primary *test* results on the ACE04, ACE05, and ACE05-ALL datasets, for the MUC and B^3 metrics. All systems reported use automatically detected mentions. We report our results (the 3 rows marked 'This Work') on the perceptron baseline, the DT baseline, and the Web features added to the DT baseline. We also report statistical significance of the improvements from the Web features on the DT baseline. All improvements are significant, except on the small ACE05 dataset with the MUC metric (where it is weak, at $p < 0.12$). However, on the larger version of this dataset, ACE05-ALL, we get improvements which are both larger and more significant (at $p < 0.001$). For significance testing, we use the bootstrap test (Noreen, 1989; Efron and Tibshirani, 1993).

Our main comparison is against Haghighi and Klein (2010), a mostly-unsupervised generative approach that models latent entity types, which generate specific entities that in turn render individual mentions. They learn on large datasets including Wikipedia, and their results are state-of-the-art in coreference resolution. We outperform their system on most datasets and metrics (except on ACE05-ALL for the MUC metric). The other systems we compare to and outperform are the perceptron-based Reconcile system of Stoyanov et al. (2009), the strong deterministic system of Haghighi and Klein (2009), and the cluster-ranking model of Rahman and Ng (2009).

We develop our features and tune their hyper-parameter values on the ACE04 development set and then use these on the ACE04 test set.[8] On the ACE05 and ACE05-ALL datasets, we directly transfer our Web features and their hyper-parameter values from the ACE04 dev-set, without any retuning. The test improvements we get on all the datasets (see Table 3.3) suggest that our features are generally useful across datasets and metrics.[9]

[6]Moreover, a DT classifier takes roughly the same amount of time and memory as a perceptron on our ACE04 development experiments. It is, however, slower and more memory-intensive ($\sim$3x) on the bigger ACE05-ALL dataset.

[7]We also initially experimented with smaller datasets (MUC6 and MUC7) and an averaged perceptron baseline, and we did see similar improvements, arguing that these features are useful independently of the learning algorithm and dataset.

[8]Note that for the ACE04 dataset only, we use the 'SmartInstanceGenerator' (SIG) filter of Reconcile that uses only a filtered set of mention-pairs (based on distance and other properties of the pair) instead of the 'AllPairs' (AP) setting that uses all pairs of mentions, and makes training and tuning very slow.

[9]For the ACE05 and ACE05-ALL datasets, we revert to the 'AllPairs' (AP) setting of Reconcile because this gives us baselines competitive with Haghighi and Klein (2010). Since we did not need to retune on these datasets, training and tuning speed were not a bottleneck. Moreover, the improvements from our Web features are similar even when tried over the SIG baseline; hence, the filter choice doesn't affect the performance gain from the Web features.

3.4 Analysis

In this section, we briefly discuss errors (in the DT baseline) corrected by our Web features, and analyze the decision tree classifier built during training (based on the ACE04 development experiments).

To study error correction, we begin with the mention pairs that are coreferent according to the gold-standard annotation (after matching the system mentions to the gold ones). We consider the pairs that are wrongly predicted to be non-coreferent by the baseline DT system but correctly predicted to be coreferent when we add our Web features. Some examples of such pairs include:

Iran ; the country
the EPA ; the agency
athletic director ; Mulcahy
Democrat Al Gore ; the vice president
Barry Bonds ; the best baseball player
Vojislav Kostunica ; the pro-democracy leader
its closest rival ; the German magazine Das Motorrad
One of those difficult-to-dislodge judges ; John Marshall

These pairs are cases where our features on hypernymy co-occurrence and entity-based context-match are informative and help discriminate in favor of the correct antecedents. One advantage of using Web-based features is that the Web has a surprising amount of information on even rare entities such as proper names. Our features also correct coreference for various cases of pronominal anaphora, but these corrections are harder to convey out of context.

Next, we analyze the decision tree built after training the classifier (with all our Web features included). Nearly 30.4% (213 out of 1023) of the non-terminal decision tree nodes and 32.2% (165 out of 512) of the terminal decision tree leaves correspond to Web features. The average classification error at the decision tree leaves corresponding to Web features is only around 2.5%, suggesting that our features are strongly discriminative for pairwise coreference decisions. When we look for strong decision nodes (and leaves) that best split the training instances at that node into the two coreferent (+) and non-coreferent (-) groups with zero or negligible error, we notice that many such useful nodes and leaves correspond to Web features. Some of the most discriminative nodes correspond to the general co-occurrence feature for most (binned) log-count values, the Hearst-style co-occurrence feature for its zero-count value, the cluster-match feature for its zero-match value, and the R2 pronoun context feature for certain (binned) log-count values (e.g., -3 and -5).

	MUC			B^3		
System	P	R	F1	P	R	F1
ACE04-TEST-RESULTS						
Stoyanov et al. (2009)	-	-	62.0	-	-	76.5
Haghighi and Klein (2009)	67.5	61.6	64.4	77.4	69.4	73.2
Haghighi and Klein (2010)	67.4	**66.6**	67.0	81.2	73.3	77.0
This Work: Perceptron Baseline	65.5	61.9	63.7	84.1	70.9	77.0
This Work: DT Baseline	**76.0**	60.7	67.5	**89.6**	70.3	78.8
This Work: DT + Web Features	74.8	64.2	**69.1**	87.5	**73.7**	**80.0**
This Work: Δ of DT+Web over DT	($p < 0.05$)		1.7	($p < 0.005$)		1.3
ACE05-TEST-RESULTS						
Stoyanov et al. (2009)	-	-	67.4	-	-	73.7
Haghighi and Klein (2009)	73.1	58.8	65.2	82.1	63.9	71.8
Haghighi and Klein (2010)	74.6	62.7	68.1	83.2	68.4	75.1
This Work: Perceptron Baseline	72.2	61.6	66.5	85.0	65.5	73.9
This Work: DT Baseline	**79.6**	59.7	68.2	**89.4**	64.2	74.7
This Work: DT + Web Features	75.0	**64.7**	**69.5**	81.1	**70.8**	**75.6**
This Work: Δ of DT+Web over DT	($p < 0.12$)		1.3	($p < 0.1$)		0.9
ACE05-ALL-TEST-RESULTS						
Rahman and Ng (2009)	75.4	64.1	69.3	54.4	70.5	61.4
Haghighi and Klein (2009)	72.9	60.2	67.0	53.2	73.1	61.6
Haghighi and Klein (2010)	77.0	**66.9**	**71.6**	55.4	**74.8**	63.8
This Work: Perceptron Baseline	68.9	60.4	64.4	80.6	60.5	69.1
This Work: DT Baseline	**78.0**	60.4	68.1	**85.1**	60.4	70.6
This Work: DT + Web Features	77.6	64.0	70.2	80.7	65.9	**72.5**
This Work: Δ of DT+Web over DT	($p < 0.001$)		2.1	($p < 0.001$)		1.9

Table 3.3: Primary test results on the ACE04, ACE05, and ACE05-ALL datasets. All systems reported here use automatically extracted system mentions. B^3 here is the B^3All version of Stoyanov et al. (2009). We also report statistical significance of the improvements from the Web features on the DT baseline, using the bootstrap test (Noreen, 1989; Efron and Tibshirani, 1993). The perceptron baseline in this work (Reconcile settings: 15 iterations, threshold = 0.45, SIG for ACE04 and AP for ACE05, ACE05-ALL) has different results from Stoyanov et al. (2009) because their current publicly available code is different from that used in their paper (*p.c.*). Also, the B^3 variant used by Rahman and Ng (2009) is slightly different from other systems (they remove all and only the singleton twinless system mentions, so it is neither B^3All nor B^3None). For completeness, our (untuned) B^3None results (DT + Web) on the ACE05-ALL dataset are P=69.9|R=65.9|F1=67.8.

Chapter 4

Acquiring Semantics as Taxonomies

In previous chapters, we developed Web-based semantic cues to resolve ambiguities in text. Next, in this chapter, we will acquire the semantics itself from the text, specifically via structured learning of hypernymy taxonomies.[1] Many tasks in natural language understanding, such as question answering, information extraction, and textual entailment, benefit from lexical semantic information in the form of types and hypernyms. A recent example is IBM's Jeopardy! system Watson (Ferrucci et al., 2010), which used type information to restrict the set of answer candidates. Information of this sort is present in term taxonomies (e.g., Figure 4.1), ontologies, and thesauri. However, currently available taxonomies such as WordNet are incomplete in coverage (Pennacchiotti and Pantel, 2006; Hovy et al., 2009), unavailable in many domains and languages, and time-intensive to create or extend manually. There has thus been considerable interest in building lexical taxonomies automatically.

In our work, we focus on the task of taking collections of terms as input and predicting a complete taxonomy structure over them as output. Our model takes a log-linear form and is represented using a factor graph (Kschischang et al., 2001) that includes both 1st-order scoring factors on directed hypernymy edges (a parent and child in the taxonomy) and 2nd-order scoring factors on sibling edge pairs (pairs of hypernym edges with a shared parent), as well as incorporating a global (directed spanning tree) structural constraint.[2] Inference for both learning and decoding uses structured loopy belief propagation or BP (Murphy et al., 1999; MacKay, 2003; Smith and Eisner, 2008), incorporating standard spanning tree algorithms (Chu and Liu, 1965; Edmonds, 1967; Tutte, 1984). The belief propagation approach allows us to efficiently and effectively incorporate heterogeneous relational evidence via hypernymy and siblinghood (e.g., coordination) cues, which we capture by semantic features based on simple surface patterns and statistics from Web n-grams and Wikipedia abstracts. We train our model to maximize the likelihood of example ontologies (such as WordNet) using stochastic optimization, automatically learning the most useful relational patterns for full taxonomy induction.

As an example of the relational patterns that our system exploits, suppose we are interested in

[1]The work described in this chapter corresponds to Bansal et al. (2013).

[2]Here, 1st-order and 2nd-order factors refer to factors connected to a single edge and a pair of edges, respectively. See Figure 4.2.

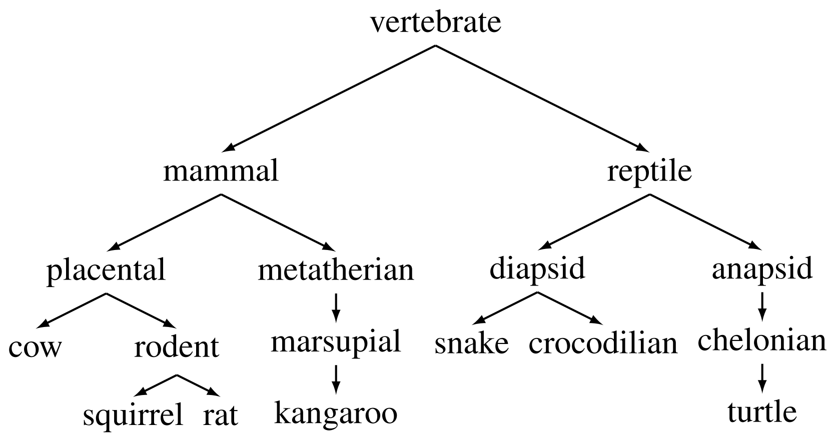


Figure 4.1: An excerpt of WordNet's vertebrates taxonomy.

building a taxonomy for types of mammals (see Figure 4.1). Frequent attestation of hypernymy patterns like *rat is a rodent* in large corpora is a strong signal of the link *rodent* $\rightarrow$ *rat*. Moreover, sibling or coordination cues like *either rats or squirrels* suggest that *rat* is a sibling of *squirrel* and adds evidence for the links *rodent* $\rightarrow$ *rat* and *rodent* $\rightarrow$ *squirrel*. Our model captures exactly these types of intuitions via features on edges and on sibling edge pairs, respectively, and automatically learns not just the weighting but the heterogeneous relational patterns themselves.

Of course, there has been a great deal of previous work on taxonomy induction (see Section 4.3 for a more detailed overview), and some key aspects of our model have appeared in different forms elsewhere. For example, Snow et al. (2006) introduced the use of 2nd-order sibling information to improve hypernymy prediction in an incremental taxonomy induction system. There are also two earlier systems, Kozareva and Hovy (2010) and Navigli et al. (2011), that, like ours, build entire taxonomies from scratch instead of simply augmenting existing taxonomies, and Navigli et al. (2011) also use a maximum spanning tree (MST) algorithm to ensure that the system outputs a coherent taxonomy tree.

The main contribution of this work is that we present the first discriminatively trained, structured probabilistic model over the space of taxonomy trees, using a structured inference procedure through both the learning and decoding phases. Our model is also the first to directly learn relational patterns as part of the process of training an end-to-end taxonomic induction system rather than using patterns that were hand-selected or learned via pairwise classifiers, and it is the first end-to-end (i.e., non-incremental) system to include sibling (e.g., coordination) patterns at all. We also note that our approach falls at a different point in the space of performance trade-offs from past work – by producing complete, highly articulated trees we naturally see a more even balance between precision and recall, while past work generally focused on precision. While different applications will value precision and recall differently, and past work was often intentionally precision-focused, it is certainly the case that an ideal solution would maximize both. To avoid presumption of a single optimal tradeoff, we present results for F1 along with recall (where we see substantial gains) and precision (where we see a drop). Note, however, that our error reduction results are based on F1.

We test our approach in two ways. First, on the task of recreating fragments of WordNet, we achieve a 51% error reduction over a chance baseline, including a 15% error reduction due to the non-hypernym-factored sibling features. Second, we also compare to the results of Kozareva and Hovy (2010) by predicting the large *animal* subtree of WordNet. Here, we get a 28% relative error reduction on ancestor-based F1 and a 12% relative error reduction on parent-based accuracy.

4.1 Structured Taxonomy Induction

Given an input term set $\boldsymbol{x} = \{x_1, x_2, \ldots, x_n\}$, we wish to compute the conditional distribution over taxonomy trees $\boldsymbol{y}$. This distribution $P(\boldsymbol{y}|\boldsymbol{x})$ is represented using the graphical model formulation shown in Figure 4.2. A taxonomy tree $\boldsymbol{y}$ is composed of a set of indicator random variables y_{ij} (circles in Figure 4.2), where y_{ij} = on means that x_i is the parent of x_j in the taxonomy tree (i.e. there exists a directed edge from x_i to x_j). One such variable exists for each pair (i, j) with $0 \leq i \leq n$, $1 \leq j \leq n$, and $i \neq j$. We assume a special dummy root symbol x_0.

In a factor graph formulation, a set of factors (squares and rectangles in Figure 4.2) determines the probability of each possible variable assignment. Each factor F has an associated scoring function ϕ_F, with the probability of a total assignment determined by the product of all these scores:

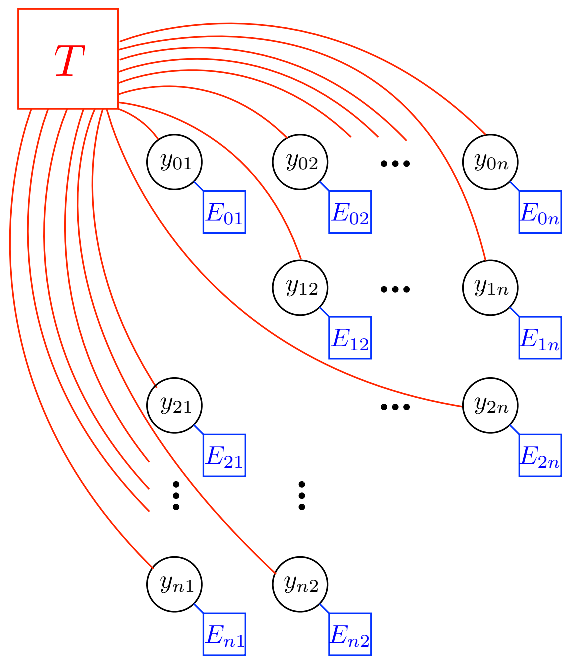
$$P(\boldsymbol{y}|\boldsymbol{x}) \propto \prod_F \phi_F(\boldsymbol{y}) \tag{4.1}$$

4.1.1 Factor Types

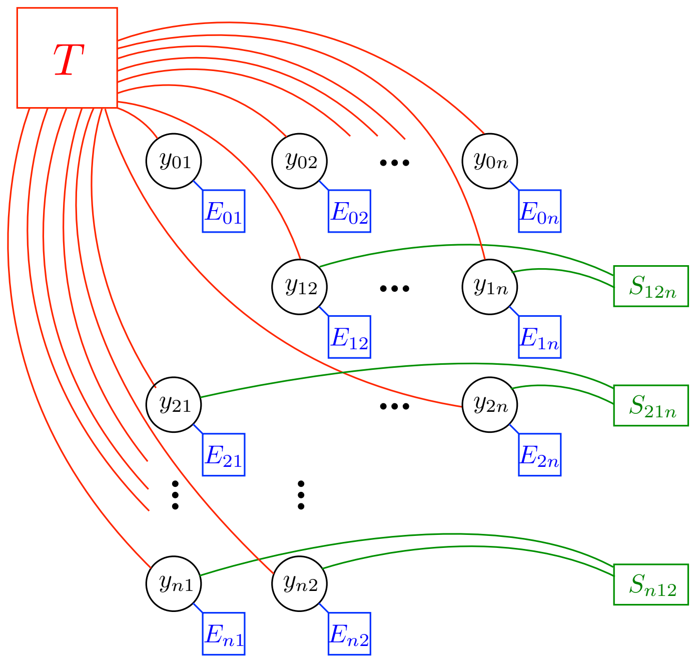
In the models we present here, there are three types of factors: EDGE factors that score individual edges in the taxonomy tree, SIBLING factors that score pairs of edges with a shared parent, and a global TREE factor that imposes the structural constraint that $\boldsymbol{y}$ form a legal taxonomy tree (i.e., a directed spanning tree).

EDGE Factors. For each edge variable y_{ij} in the model, there is a corresponding factor E_{ij} (small blue squares in Figure 4.2) that depends only on y_{ij}. We score each edge by extracting a set of features $\mathbf{f}(x_i, x_j)$ and weighting them by the (learned) weight vector $\mathbf{w}$. So, the factor scoring function is:

$$\phi_{E_{ij}}(y_{ij}) = \begin{cases} \exp(\mathbf{w} \cdot \mathbf{f}(x_i, x_j)) & y_{ij} = \text{on} \\ 1 & y_{ij} = \text{off} \end{cases}$$



(a) Edge Features Only



(b) Full Model

Figure 4.2: Factor graph representation of our model, both without (a) and with (b) SIBLING factors.

SIBLING Factors. Our second model also includes factors that permit 2nd-order features looking at terms that are siblings in the taxonomy tree. For each triple (i, j, k) with $i \neq j$, $i \neq k$, and $j < k$,[3] we have a factor S_{ijk} (green rectangles in Figure 4.2b) that depends on y_{ij} and y_{ik}, and thus can be used to encode features that should be active whenever x_j and x_k share the same parent, x_i. The scoring function is similar to that for the EDGE factors:

$$\phi_{S_{ijk}}(y_{ij}, y_{ik}) = \begin{cases} \exp(\mathbf{w} \cdot \mathbf{f}(x_i, x_j, x_k)) & y_{ij} = y_{ik} = \text{on} \\ 1 & \text{otherwise} \end{cases}$$

TREE Factor. Of course, not all variable assignments $\boldsymbol{y}$ form legal taxonomy trees (i.e., directed spanning trees). For example, the assignment $\forall i, j,\ y_{ij} = \text{on}$ might get a high score, but would not be a valid output of the model. Thus, we need to impose a structural constraint to ensure that such illegal variable assignments are assigned 0 probability by the model. We encode this in our factor graph setting using a single global factor T (shown as a large red square in Figure 4.2) with the following scoring function:

$$\phi_T(\boldsymbol{y}) = \begin{cases} 1 & \boldsymbol{y} \text{ forms a legal taxonomy tree} \\ 0 & \text{otherwise} \end{cases}$$

For a given global assignment $\boldsymbol{y}$, let

$$\mathbf{f}(\boldsymbol{y}) = \sum_{\substack{i,j \\ y_{ij}=\text{on}}} \mathbf{f}(x_i, x_j) + \sum_{\substack{i,j,k \\ y_{ij}=y_{ik}=\text{on}}} \mathbf{f}(x_i, x_j, x_k)$$

Note that by substituting our model's factor scoring functions into Equation 4.1, we get:

$$P(\boldsymbol{y}|\boldsymbol{x}) \propto \begin{cases} \exp(\mathbf{w} \cdot \mathbf{f}(\boldsymbol{y})) & \boldsymbol{y} \text{ is a tree} \\ 0 & \text{otherwise} \end{cases}$$

Thus, our model has the form of a standard log-linear model with feature function $\mathbf{f}$.

4.1.2 Inference via Belief Propagation

With the model defined, there are two main inference tasks we wish to accomplish: computing expected feature counts and selecting a particular taxonomy tree for a given set of input terms

[3]Unlike in dependency parsing, the ordering of the siblings x_j and x_k doesn't matter, so having separate factors for (i, j, k) and (i, k, j) would be redundant.

(decoding). As an initial step to each of these procedures, we wish to compute the marginal probabilities of particular edges (and pairs of edges) being on. In a factor graph, the natural inference procedure for computing marginals is belief propagation. Note that finding taxonomy trees is a structurally identical problem to directed spanning trees (and thereby non-projective dependency parsing), for which belief propagation has previously been worked out in depth (Smith and Eisner, 2008). However, we will briefly sketch the procedure here.

Belief propagation is a general-purpose inference method that computes marginals via messages that are passed between variables and factors in the factor graph. Messages are always between an adjacent variable and factor in the graph, and take the form of (possibly unnormalized) distributions over values of the variable. There are two types of messages, depending on the direction (variable to factor or factor to variable), with mutually recursive definitions.

Let $N(V)$ denote the set of factors neighboring variable V in the factor graph and $N(F)$ the set of variables neighboring F. The message from a variable V to a factor F is fairly simple, as it simply collects the information that variable has received from all of its *other* adjacent factors:

$$m_{V \to F}(v) \propto \prod_{F' \in N(V) \setminus \{F\}} m_{F' \to V}(v)$$

The message from F to V collects information from other adjacent variables, but also includes the factor's own scoring function:

$$m_{F \to V}(v) \propto \sum_{\mathcal{X}_F, \mathcal{X}_F[V]=v} \phi_F(\mathcal{X}_F) \prod_{V' \in N(F) \setminus V} m_{V' \to F}(\mathcal{X}_F[V']) \tag{4.2}$$

where $\mathcal{X}_F$ is a partial assignment of values to just the variables in $N(F)$.

The idea behind Equation 4.2 is that if you're computing the message for a specific value v, you fix that value, then sum over all possible values of every other variable that F touches, multiplying together the factor score (which depends on all these variables) and the messages for each other variable to get the summand for each assignment. For example, if computing a message from a SIBLING factor to an adjacent variable, the equation comes out to:

$$m_{S_{ijk} \to Y_{ij}}(y_{ij}) \propto \sum_{y_{ik}} \phi_{S_{ijk}}(y_{ij}, y_{ik}) m_{Y_{ik} \to S_{ijk}}(y_{ik})$$

While the EDGE and SIBLING factors are simple enough to perform the computation in Equation 4.2 by brute force, performing the sum naïvely for computing messages from the TREE factor would take exponential time. However, due to the structure of that particular factor, all of its outgoing messages can be computed simultaneously in $O(n^3)$ time by using an efficient adaptation of Kirchhoff's Matrix Tree Theorem (Tutte, 1984) which computes partition functions and marginals for directed spanning trees. See Smith and Eisner (2008) for details on how to incorporate the MTT into belief propagation.

Once message passing is completed, marginal beliefs are computed by merely multiplying together all the messages received by a particular variable or factor:

$$b_V(v) \propto \prod_{F \in N(V)} m_{F \to V}(v) \tag{4.3}$$

$$b_F(v_{N(F)}) \propto \phi_F(v_{N(F)}) \prod_{V \in N(F)} m_{V \to F}(v) \tag{4.4}$$

4.1.2.1 Loopy Belief Propagation

Looking closely at Figure 4.2a, one can observe that the factor graph for the first version of our model, containing only EDGE and TREE factors, is acyclic. In this special case, belief propagation is exact: after one round of message passing,[4] the beliefs computed by equations 4.3 and 4.4 will be the true marginal probabilities under the current model. However, in the full model, shown in Figure 4.2b, the SIBLING factors introduce cycles into the factor graph. When a factor graph contains cycles, the messages that are being passed around often depend on each other and so they will change as they are recomputed. The degree of dependency depends on the length of the cycles in the graph and the strength of variable interactions in the factor scoring functions.

The process of iteratively recomputing messages based on earlier messages is known as *loopy* belief propagation. This procedure only finds *approximate* marginal beliefs, and is not actually guaranteed to converge, but in practice can be quite effective for finding workable marginals in models for which exact inference is intractable, as is the case here. All else equal, the more rounds of message passing that are performed, the closer the computed marginal beliefs will be to the true marginals, though in practice, there are usually diminishing returns after the first few iterations. In our experiments, we used a fairly conservative upper bound of 20 iterations, but in most cases, the messages converged much earlier than that.

4.1.3 Training

We used gradient-based maximum likelihood training to learn the model parameters $\mathbf{w}$. Since our model has a log-linear form, the derivative of $\mathbf{w}$ with respect to the likelihood objective is computed by just taking the gold feature vector and subtracting the vector of expected feature counts.

We have features from two types of factors (EDGE and SIBLING), but their expected counts are computed in the same way. First, we run belief propagation until completion (one iteration for the first model, several for the second). Then, for each factor in the model we simply read off

[4]Assuming messages are ordered appropriately along the graph.

the marginal probability of that factor being active[5] according to Equation 4.4, and accumulate a partial count for each feature that's fired by that factor. If the model permits exact inference, then this will yield the exact gradient, whereas if the marginal beliefs are computed using loopy belief propagation then the gradient will be an approximation, though hopefully a reasonable one.

This method of computing the gradient can be plugged into any gradient-based optimizer in order to learn the weights $\mathbf{w}$. In our experiments we used AdaGrad (Duchi et al., 2011), an adaptive subgradient variant of standard stochastic gradient ascent for online learning.

4.1.4 Decoding

Finally, once the model parameters have been learned, we want to use the model to find taxonomy trees for particular sets of input terms. Note that if we limit our scores to be edge-factored, then, as in non-projective dependency parsing, finding the highest scoring taxonomy tree becomes an instance of the MST problem. Also known as the maximum arborescence problem (for the directed case), the MST problem can be solved efficiently in quadratic time (Tarjan, 1977) using the greedy, recursive Chu-Liu-Edmonds algorithm (Chu and Liu, 1965; Edmonds, 1967). See Georgiadis (2003) for a detailed algorithmic proof, and McDonald et al. (2005) for an illustrative example. Also, we constrain the Chu-Liu-Edmonds MST algorithm to output only *single-root* MSTs, where the (dummy) root has exactly one child (Koo et al., 2007), because multi-root spanning 'forests' are not applicable to our task.

Since the MST problem can be solved efficiently, the main challenge becomes finding a way to ensure that our scores are edge-factored. In the first version of our model, we could simply set the score of each edge to be $\mathbf{w} \cdot \mathbf{f}(x_i, x_j)$, and the MST recovered in this way would indeed be the highest scoring tree: $\arg\max_{\boldsymbol{y}} P(\boldsymbol{y}|\boldsymbol{x})$. However, this straightforward approach doesn't apply to the full model which also uses sibling features. Hence, at decoding time, we instead start out by once more using belief propagation to find marginal beliefs. Once we have computed marginal edge beliefs according to Equation 4.3, we set the score of each edge to be its belief odds ratio: $\frac{b_{Y_{ij}}(\text{on})}{b_{Y_{ij}}(\text{off})}$. The MST that is found using these edge scores is actually the minimum Bayes risk tree (Goodman, 1996) for an edge accuracy loss function (Smith and Eisner, 2008).

4.1.4.1 Relationship to Dependency Parsing

Spanning trees are familiar from non-projective dependency parsing (McDonald et al., 2005). However, there are some important differences between dependency parsing and taxonomy induction (see Figure 4.3). First, where the linear order (i.e. the indexing) of the words is critical for dependency parsing, it is purely notational for taxonomy induction, and so here no features will refer to the indices' order (such as the distance, linear direction, and in-between identity features used in dependency parsing). Moreover, features based on lexical identities and syntactic word classes, which are primary drivers for dependency parsing, are mostly uninformative here. Instead,

[5]By *active* we just mean that features are being extracted, so for E_{ij}, active means that $y_{ij} = \text{on}$, whereas for S_{ijk}, active means that $y_{ij} = y_{ik} = \text{on}$.

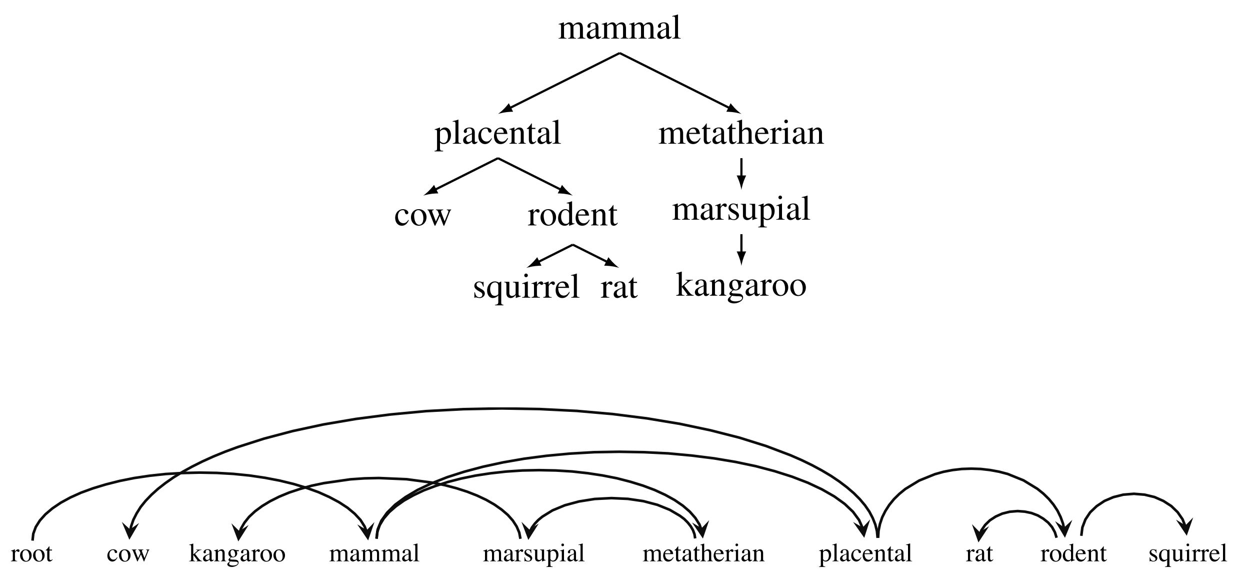


Figure 4.3: A *mammal* taxonomy subtree and its equivalent dependency tree (non-projective). In both cases, we have used the standard dependency notation of the arrow directed from parent to child.

in taxonomy induction, we need to learn semantic relational preferences, for which we will next present features that capture the relations based on co-occurrence patterns and statistics in large corpora, in addition to surface features that capture string similarities.

4.2 Features

Inducing taxonomies requires world knowledge to capture the semantic relations between various terms. Hence, configurational and word class features are mostly uninformative here. Instead, we use semantic cues to hypernymy and siblinghood via simple surface patterns and statistics in large text corpora.[6] We fire features on both the edge and the sibling factors. We first describe all the edge features in detail (Section 4.2.1 and Section 4.2.2), and then briefly describe the sibling features (Section 4.2.3), which are very similar to the edge ones.

For each edge factor E_{ij}, which represents the potential parent-child term pair (x_i, x_j), we fire the surface and semantic features discussed below. Since edges are directed, we have separate

[6]Note that one could also add various complementary types of useful features presented by previous work, e.g., bootstrapping using syntactic heuristics (Phillips and Riloff, 2002), dependency patterns (Snow et al., 2006), doubly anchored patterns (Kozareva et al., 2008; Hovy et al., 2009), and Web definition classifiers (Navigli et al., 2011). However, in our work, we focus mainly on the structured learning aspect.

features for the factors E_{ij} versus E_{ji}.

4.2.1 Surface Features

Capitalization: Checks which of x_i and x_j are capitalized, with one feature for each value of the tuple (isCap(x_j), isCap(x_i)). The intuition is that leaves of a hypernymy taxonomy are often proper names and hence capitalized. Therefore, the feature for (*false, true*) (i.e., parent capitalized but not the child) gets a substantially negative weight.

Ends with: Checks if x_j ends with x_i, or not. The intuition is that various children nodes in a hypernymy taxonomy are composed of their parent string as the suffix, e.g., pairs such as (*nut, chestnut*), (*bee, honeybee*), and (*salad, potato salad*) in our data.

Contains: Checks if x_j contains x_i, or not. The intuition is that various children nodes in a hypernymy taxonomy contain their parent string, e.g., pairs such as (*bird, bird of prey*), (*terror, bioterrorism*), and (*war, trench warfare*) in our data.

Suffix match: Checks whether the k-length suffixes of x_i and x_j match, or not, for $k = 1, 2, \ldots, 7$. This feature has an intuition of general string match, especially of the suffix portions of the strings.

LCS: We compute the longest common substring of x_i and x_j and create indicator features for rounded-off and binned values of $|LCS|/((|x_i| + |x_j|)/2)$. This feature also has a similar intuition of general surface match between the two term strings, and can capture pairs such as (*oratory, peroration*), (*octopod, octopus*), and (*undertide, undertow*) in our data.

Length difference: We compute the signed length difference between x_j and x_i, and create indicator features for rounded-off and binned values of $(|x_j| - |x_i|)/((|x_i| + |x_j|)/2)$.[7]

4.2.2 Semantic Features

4.2.2.1 Web n-gram Features

Patterns and counts: Hypernymy for a term pair (*P*=x_i, *C*=x_j) is often signaled by the presence of surface patterns like *C is a P*, *P such as C* in large text corpora, an observation going back to Hearst (1992). Therefore, for each potential parent-child edge (x_i, x_j), we mine the top k patterns (based on count) in which both x_i and x_j occur (we use k=200). We collect patterns in both directions, which allows us to judge the correct direction of an edge (e.g., *C is a P* is a positive signal for hypernymy whereas *P is a C* is a negative signal).[8] Next, for each pattern in this top-k list, we

[7] Yang and Callan (2009) use a variant for learning a similarity metric.

[8] We also allow patterns with surrounding words, e.g., ***the** C is a P* and *C , P **of***.

compute its normalized pattern count c, and fire an indicator feature on the tuple $(pattern, t)$, for all thresholds t (in a fixed set) s.t. $c \geq t$.

Pattern order: We fire features on the order (direction) in which the pair (x_i, x_j) found a pattern (in its top-k list) – indicator features for boolean values of the four cases: $P \dots C$, $C \dots P$, neither direction, and both directions.[9]

Individual counts: We also compute the individual Web-scale term counts c_{x_i} and c_{x_j}, and fire a comparison feature ($c_{x_i}{>}c_{x_j}$), plus features on values of the signed count difference $(|c_{x_i}| - |c_{x_j}|)/((|c_{x_i}| + |c_{x_j}|)/2)$, after rounding off, and binning at multiple granularities. The intuition is that this feature could learn whether the relative popularity of the terms signals their hypernymy direction.

4.2.2.2 Wikipedia Abstract Features

The Web n-grams corpus has broad coverage but is limited in length (up to 5-grams only), so it may not contain pattern-based evidence for various multi-word terms and pairs. Therefore, we supplement it with a full-sentence resource, namely *Wikipedia abstracts*, which are short, useful descriptions of a large variety of world entities.

Presence and distance: For each potential edge (x_i, x_j), we mine patterns from all abstracts in which the two terms co-occur in either order, allowing a maximum term distance of 20 (because beyond that, co-occurrence may not imply a relation). We fire a presence feature based on whether the process above found at least one pattern for that term pair, or not. We also fire features on the value of the minimum distance d_{min} at which the two terms were found in some abstract (plus thresholded versions).

Patterns: For each term pair, we take the top-k' patterns (based on count) of length up to l from its full list of patterns, and fire an indicator feature on each pattern string (without the counts). We use k'=5, l=10. Similar to the Web n-grams case, we also fire Wikipedia-based pattern order features.

4.2.3 Sibling Features

We also fire similar features on sibling factors. For each sibling factor S_{ijk} which represents the potential parent-children term triple (x_i, x_j, x_k) we consider the potential sibling term pair (x_j, x_k). Siblinghood for this pair is frequently indicated by the presence of surface patterns such as *either* C_1 *or* C_2, C_1 *is similar to* C_2 in large text corpora. Therefore, we fire Web n-gram pattern features and Wikipedia presence, distance, and pattern features, similar to those described above, on each potential sibling term pair.[10] The main difference here from the edge factors is that the

[9]Ritter et al. (2009) used the 'both' case for hypernym discovery.

[10]One can also fire features on the full triple (x_i, x_j, x_k) but most such features will be sparse.

sibling factors are symmetric (in the sense that S_{ijk} is redundant to S_{ikj}) and hence the patterns are undirected. Therefore, for each term pair, we first symmetrize the collected Web n-grams and Wikipedia patterns by accumulating the counts of symmetric patterns like *rats or squirrels* and *squirrels or rats*.

Note that the patterns and counts for all the Web and Wikipedia semantic features described above (both for hypernymy and siblinghood) are extracted after stemming the words in the terms, the n-grams, and the abstracts. Also, we threshold the features (to prune away the sparse ones) by considering only those that fire for at least t trees in the training data ($t = 4$ in our experiments).

4.3 Related Work

In our work, we assume a known term set and do not address the problem of extracting terms from text. However, a great deal of past work has considered automating this process, typically taking one of two major approaches. The clustering-based approach (Lin, 1998; Lin and Pantel, 2002; Davidov and Rappoport, 2006; Yamada et al., 2009) discovers relations based on the assumption that similar concepts appear in similar contexts (Harris, 1954). The pattern-based approach uses special lexico-syntactic patterns to extract pairwise relation lists (Phillips and Riloff, 2002; Girju et al., 2003; Pantel and Pennacchiotti, 2006; Suchanek et al., 2007; Nakov and Hearst, 2008; Ritter et al., 2009; Hovy et al., 2009; Baroni et al., 2010; Ponzetto and Strube, 2011) and semantic classes or class-instance pairs (Riloff and Shepherd, 1997; Katz and Lin, 2003; Pasca, 2004; Etzioni et al., 2005; Talukdar et al., 2008). More recent systems have combined the clustering and pattern based methods (Snow et al., 2004; Davidov and Rappoport, 2008a; Pennacchiotti and Pantel, 2009; Yang and Callan, 2009; Do and Roth, 2010) with stronger results.

Our work focuses on the second step of taxonomy induction, namely the structured organization of terms into a complete and coherent tree-like hierarchy.[11] Early work on this task assumes a starting partial taxonomy and inserts missing terms into it. Widdows (2003) use a class-labeling algorithm to place an unknown word into a region of the starting taxonomy with the most semantically-similar neighbors, where semantic neighbors of an unknown word are found using latent semantic analysis combined with part-of-speech information. Snow et al. (2006) incrementally add novel terms to a starting taxonomy by greedily maximizing the conditional probability of a set of relational evidence (over the heterogeneous relationships of hypernymy and coordination) given a taxonomy. Yang and Callan (2009) incrementally cluster terms based on an ontology metric, which is a pairwise semantic distance score computed using various heterogeneous features such as context, co-occurrence, and surface patterns. Lao et al. (2012) extend a large knowledge base using a path-constrained random walk model that learns syntactic-semantic inference rules for binary relations from a graph representation of a parsed Web-scale text corpus and the knowledge base.

However, the task of inducing full taxonomies without assuming a substantial initial partial taxonomy is relatively less well studied. There is some amount of prior work on the related task

[11]Determining the set of input terms is orthogonal to our work, and our method can be used in conjunction with various term extraction approaches.

of hierarchical clustering, or grouping together of semantically related words: some based on formal concept analysis (Cimiano et al., 2005), some on formal statements and parsing (Poon and Domingos, 2010), and some on random graphs (Fountain and Lapata, 2012). The task we focus on, though, is the discovery of direct taxonomic relationships (e.g., hypernymy) between words.

We know of two previous systems, Kozareva and Hovy (2010) and Navigli et al. (2011), that build full taxonomies from scratch, instead of relying on an initial seed taxonomy. Both of these systems use a process that starts by finding basic level terms (leaves of the final taxonomy tree, typically) and then using relational patterns (doubly-anchored, hand-selected ones in the case of Kozareva and Hovy (2010) and ones learned separately by a supervised pairwise classifier in the case of Navigli et al. (2011)) to find intermediate terms and all the attested hypernymy links between them.[12] To prune down the resulting taxonomy graph, Kozareva and Hovy (2010) use a procedure that iteratively retains the longest paths between root and leaf terms, removing conflicting graph edges as they go. The end result is acyclic, though not necessarily a tree; Navigli et al. (2011) instead use the longest path intuition to weight edges in the graph and then find the highest weight taxonomic tree using a standard maximum spanning tree (MST) algorithm.

4.4 Experiments

4.4.1 Data and Experimental Regime

We considered two distinct experimental setups, one that illustrates the general performance of our model by reproducing various medium-sized WordNet domains, and another that facilitates comparison to previous work by reproducing the much larger *animal* subtree provided by Kozareva and Hovy (2010).

General setup: In order to test the accuracy of structured prediction on medium-sized full-domain taxonomies, we extracted from WordNet 3.0 all bottomed-out full subtrees which had a tree-height of 3 (i.e., 4 nodes from root to leaf), and contained (10, 50] terms.[13] This process gives 761 trees which we partition into 70/15/15% (533/114/114 trees) train/dev/test splits.

Comparison setup: In order to facilitate comparison with previous work, we also tested on the much larger *animal* subtree made available by Kozareva and Hovy (2010), who created this dataset by selecting a set of 'harvested' terms and retrieving all the WordNet hypernyms between each input term and the root that is *animal*, resulting in ∼700 terms and ∼4,300 *is-a* ancestor-child

[12]Unlike our system, which assumes a complete set of terms and only attempts to induce the taxonomic structure, both these systems include term discovery in the taxonomy building process.

[13]Subtrees that had a smaller or larger tree height were discarded in order to avoid overlap between the training and test divisions. This is a stricter setting than other tasks such as parsing, which usually has repeated sentences, clauses and phrases. To project WordNet synsets to terms, we used the first (most frequent) word in each synset. A few synsets in WordNet have multiple parents so we only keep the first of each such pair of overlapping trees. We also discard a few trees with duplicate terms because this is mostly due to the projection of different synsets to the same term, and theoretically makes the tree a graph.

links.[14] The training set for this *animal* test case was generated from WordNet using the following process: First, we strictly remove the *full* animal subtree from WordNet in order to avoid any possible overlap with the test data. Next, we create random 25-sized trees by picking random nodes as singleton trees, and repeatedly adding child edges from WordNet to the tree. This process gives us a total of $\sim$1600 training trees. We choose this training regimen as different from that of the general setup (which contains only bottomed-out subtrees), so as to match the *animal* test tree, which has a depth of 12 and intermediate nodes from higher up in WordNet.

Feature sources: The n-gram semantic features are extracted from the Google n-grams corpus (Brants and Franz, 2006), a large collection of English n-grams (for $n = 1$ to 5) and their frequencies computed from almost 1 trillion tokens (95 billion sentences) of Web text. The Wikipedia abstracts are obtained via the publicly available dump, which contains almost $\sim$4.1 million articles.[15] Preprocessing includes standard XML parsing and tokenization. We use these offline corpora as opposed to search engine APIs or # of page hits, which face multiple issues such as daily query limits, speed, quality of post-processed search results, instability, and irreproducibility (Kilgarriff, 2007). That said, adding other large, complementary knowledge resources used by previous work should increase our coverage and evidence, and provide better results.

Efficient collection of feature statistics is important because these must be extracted for millions of query pairs (for each potential edge and sibling pair in each term set). For this, we use a hash-trie on term pairs, and scan once through the n-gram (or abstract) set, skipping many n-grams (or abstracts) based on fast checks of missing unigrams, exceeding length, suffix mismatches, etc.

4.4.2 Evaluation Metrics

Ancestor F1: Measures the precision, recall, and F1 of correctly predicted ancestors, i.e., pairwise *is-a* relations:

$$P = \frac{|\text{isa}_{\text{gold}} \cap \text{isa}_{\text{predicted}}|}{|\text{isa}_{\text{predicted}}|}$$

$$R = \frac{|\text{isa}_{\text{gold}} \cap \text{isa}_{\text{predicted}}|}{|\text{isa}_{\text{gold}}|}$$

$$F_1 = \frac{2\,P\,R}{(P+R)}$$

[14]This is somewhat different from our general setup where we work with any given set of terms; they start with a large set of leaves which have substantial Web-based relational information based on their selected, hand-picked patterns.
The gold standard data is provided only in the form of the *is-a* ancestor-child links, but in order to measure parent accuracy, we find the corresponding exact WordNet tree (by removing redundant ancestor links and some minor cleaning). The data is available at `http://www.isi.edu/~kozareva/downloads.html`. They also provide two other datasets on plants and vehicles but these are much smaller, and seemed to have some incorrect pairs and inconsistent terms.

[15]`http://dumps.wikimedia.org/enwiki/20130102/enwiki-20130102-abstract.xml`

	Ancestor evaluation			Parent evaluation
	Precision	Recall	F1	Accuracy
Edges-Only Model				
Baseline	5.9	8.3	6.9	3.6
Surface Features	17.5	41.3	24.6	28.8
Semantic Features	37.0	49.1	42.2	36.7
Surface + Semantic Features	41.1	54.4	46.8	45.3
Edges + Siblings Model				
Surface + Semantic Features	**53.1**	**56.6**	**54.8**	**46.4**
Surface + Semantic Features (Test)	48.0	55.2	51.4	46.0

Table 4.1: Main results on our general setup: on the development set, we present incremental results on the edges-only model where we start with the chance baseline, then use surface features only, semantic features only, and both. Finally, we add sibling factors and features to get results for the full, edges+siblings model with all features, and also report the final test result for this setting.

Parent Accuracy: We also evaluate on a different metric that more strictly evaluates the correctness of direct parent attachments.

$$Accuracy = \frac{\#\text{ nodes with correctly predicted parent}}{\#\text{ nodes}}$$

We compute the micro-averaged version of both the ancestor and parent metrics, where we first collect the no. of correctly predicted links across all instances (trees), and then calculate an overall P, R, and accuracy.

4.4.3 Results

Table 4.1 shows our main results for both ancestor-based F1 and parent-based accuracy on the general setup. We present a development set ablation study where we start with the edges-only model (Figure 4.2a) and its random tree baseline (which chooses any arbitrary spanning tree for the term set). Next, we show results on the edges-only model with surface features, semantic features, and both. We see that both surface and semantic features make substantial contributions. Finally, we add the sibling factors and features (Figure 4.2b), which further improves the results significantly (15% relative error reduction over the edges-only results on the ancestor F1 metric).

	Ancestor evaluation			Parent evaluation
System	Precision	Recall	F1	Accuracy
Fixed Prediction				
Kozareva and Hovy (2010)	**98.6**	36.2	52.9	20.8
This Work	84.6	**54.3**	**66.2**	**30.5**
Free Prediction				
Navigli et al. (2011)	**97.0***	43.7*	60.3*	–
This Work	79.3	**49.0**	**60.6**	28.1

Table 4.2: Comparison results on the *animal* dataset of Kozareva and Hovy (2010). For appropriate comparison to each previous work, we show results both for the 'Fixed Prediction' setup, which assumes the true root and leaves, and for the 'Free Prediction' setup, which doesn't assume any prior information. The * results represent a different data condition; see Section 4.4.3.

The last row shows the final test set results for the full model with all features.[16]

Table 4.2 shows our results for comparison to the larger *animal* dataset of Kozareva and Hovy (2010).[17] For appropriate comparison to each previous work, we show results for two different setups. The first setup 'Fixed Prediction' assumes that the model knows the true root and leaves of the taxonomy to provide for a fairer comparison to Kozareva and Hovy (2010). We get substantial improvements over their results on ancestor-based recall and F1 (a 28% relative error reduction), and also on parent-based accuracy (a 12% relative error reduction). The second setup 'Free Prediction' assumes no prior knowledge and predicts the full tree (similar to the general setup case). On this setup, we do compare as closely as possible to Navigli et al. (2011) and see a very small gain in F1, but regardless, we should note that their results are somewhat incomparable because they have a different data condition: their definition and hypernym extraction phase involves using the Google `define` keyword, which often returns WordNet glosses. However, we should also note that previous work achieves substantially higher ancestor precision, while our approach achieves a more even balance between precision and recall. Of course, precision and recall should both ideally be high, even if some applications weigh one over the other. This is why we chose to optimize F1, which represents a neutral combination for comparison, but other F_α metrics could also be optimized.

[16]Note that our results are based on tuning only on the ancestor metric, and one could obtain better parent-based results than reported, with independent optimization.

[17]These results are for the 1st order model due to the scale of the taxonomy. The Kozareva and Hovy (2010) ancestor results are obtained by using the output files provided on their webpage. The parent results are obtained by converting their graph output to a tree by removing redundant ancestor links and then randomly choosing one parent for any remaining terms with multiple parents.

Hypernymy features	
C and other P	*> P > C*
C , P of	*C is a P*
C , a P	*P , including C*
C or other P	*P (C*
C : a P	*C , american P*
C - like P	*C , the P*

Siblinghood features	
C_1 *and* C_2	C_1, C_2 *(*
C_1 *or* C_2 *of*	C_1 *and / or* C_2
, C_1 *,* C_2 *and*	*either* C_1 *or* C_2
the C_1 */* C_2	*<s>* C_1 *and* C_2 *</s>*

Table 4.3: Examples of high-weighted hypernymy and siblinghood features learned during development.

4.5 Analysis

Table 4.3 shows some of the semantic hypernymy and siblinghood features given highest weight by our model (in development experiments). The training process not only rediscovers most of the standard Hearst-style hypernymy patterns (e.g., *C and other P*, *C is a P*), but also finds various other intuitive patterns. For example, the pattern *C, american P* is popular because it captures pairs like *Lemmon, american actor* and *Bryon, american politician*, etc. Another pattern $>$ *P* $>$ *C* captures webpage navigation breadcrumb trails. Similarly, the algorithm also learns useful siblinghood features, e.g., *either* C_1 *or* C_2, C_1 *and / or* C_2, etc.

Finally, we look at some specific output errors to give as concrete a sense as possible of some system confusions, though of course any hand-chosen examples must be taken as illustrative. In Figure 4.4, we attach *white admiral* to *admiral*, whereas the gold standard makes these two terms siblings. In reality, however, white admirals are indeed a species of admirals, so WordNet's ground truth turns out to be incomplete. Another such example is that we place *logistic assessment* in the *evaluation* subtree of *judgment*, but WordNet makes it a direct child of *judgment*. However, other dictionaries do consider logistic assessments to be evaluations. Hence, this illustrates that there may be more than one right answer, and that the low results on this task should only be interpreted as such. In Figure 4.5, our algorithm did not recognize that *thermos* is a hyponym of *vacuum flask*, and that *jeroboam* is a kind of wine bottle. Here, our Web pattern features from the Google n-Grams (which only contain frequent n-grams) do not suffice. We would need to add richer Web data for such world knowledge to be reflected in the features.

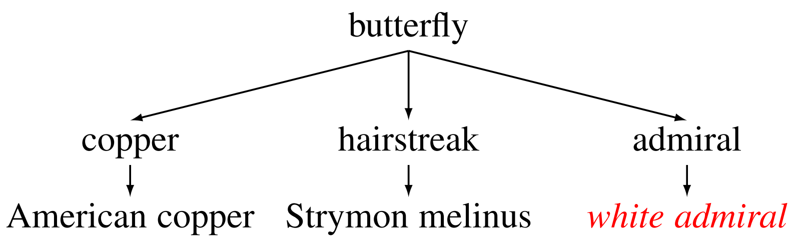


Figure 4.4: Excerpt from the predicted *butterfly* tree. The terms attached erroneously according to WordNet are marked in red and italicized.

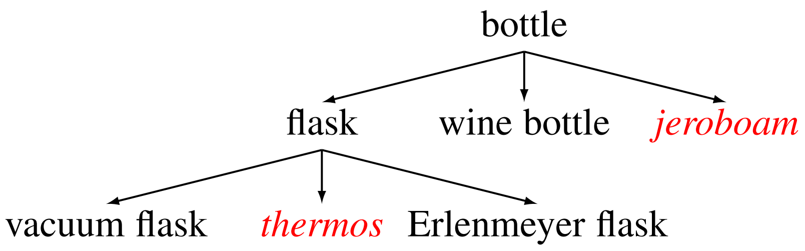


Figure 4.5: Excerpt from the predicted *bottle* tree. The terms attached erroneously according to WordNet are marked in red and italicized.

Chapter 5

Deeper Semantics via Intensity Ordering

Finally, in this chapter, we discuss a finer-grained semantic facet – intensity order.[1] Current lexical resources such as dictionaries and thesauri do not provide information about the intensity order of words. For example, both WordNet (Miller, 1995) and Roget's 21st Century Thesaurus present *acceptable*, *great*, and *superb* as synonyms or similar words of the adjective *good*. However, a native speaker knows that these words represent varying intensity and can in fact generally be ranked by intensity as *acceptable* $<$ *good* $<$ *great* $<$ *superb*. Similarly, *warm* $<$ *hot* $<$ *scorching* are identified as synonyms or similar words in these resources. Ranking information, however, is crucial because it allows us to differentiate e.g., between various intensities of an emotion, and is hence very useful for humans when learning a language, as well as for automatic text understanding and generation tasks such as sentiment analysis, question answering, summarization, and discourse analysis.

Our work attempts to automatically rank sets of related words by intensity, focusing in particular on near-synonym adjectives. This is made possible by the vast amounts of world knowledge that are now available. We use lexico-semantic information extracted from a Web-scale corpus in conjunction with an algorithm based on a Mixed Integer Linear Program (MILP). Linguistic analyses have identified phrases such as *good but not great* or *hot and almost scorching* in a text corpus as sources of evidence about the relative intensities of words . However, pure information extraction approaches often fail to provide enough coverage for real-world downstream applications (Tandon and de Melo, 2010), unless some form of advanced inference is used (Snow et al., 2006; Suchanek et al., 2009). In our work, we address this sparsity issue in two ways. First, we exploit large amounts of Web-scale data, using special intensity patterns and Web statistics to compute pairwise intensity order scores. Second, we use a linear programming model that extends the pairwise scores to a more complete joint ranking of words on a continuous scale, while maintaining global constraints such as transitivity and giving more weight to the order of word pairs with higher corpus evidence scores. Instead of considering intensity ranking as a pairwise decision process, we thus exploit the fact that *individual decisions may benefit from global information*, e.g., about

[1]The work described in this chapter was originally presented at ACL in 2013 (De Melo and Bansal, 2013). Despite the author order, this work represents an equal contribution of each author.

how two words relate to some third word.

Previous work (Sheinman and Tokunaga, 2009; Schulam and Fellbaum, 2010; Sheinman et al., 2012) has also used lexico-semantic patterns to order adjectives. They mainly evaluate their algorithm on a set of pairwise decisions, but also present a partitioning approach that attempts to form scales by placing each adjective to the left or right of pivot words. Unfortunately, this approach often fails because many pairs lack order-based evidence even on the Web, as explained in more detail in Section 5.2.

In contrast, our MILP jointly uses information from all relevant word pairs and captures complex interactions and inferences to produce intensity scales. We can thus obtain an order between two adjectives even when there is no explicit evidence in the corpus (using evidence for related pairs and transitive inference). Our global MILP is flexible and can incorporate additional exact-synonymy information if available (which helps the MILP find an even better ranking solution). Our approach extends easily to new languages. We describe two approaches for this multilingual extension: pattern projection and cross-lingual MILPs. Moreover, our approach can also straightforwardly be applied to other word classes (such as verbs).

We evaluate our predicted intensity rankings using both pairwise classification accuracy and ranking correlation coefficients, achieving strong results, significantly better than the previous approach by Sheinman & Tokunaga (32% relative error reduction) and quite close to human-level performance.

5.1 Method

In this section, we describe each step of our approach to ordering adjectives on a single, relative scale. Our method can also be applied to other word classes and to languages other than English.

5.1.1 Web-based Scoring Model

5.1.1.1 Intensity Scales

Near-synonyms may differ in intensity, e.g., *joy* vs. *euphoria*, or *drizzle* vs. *rain*. This is particularly true of adjectives, which can represent different degrees of a given quality or attribute such as size or age. Many adjectives are gradable and thus allow for grading adverbial modifiers to express such intensity degrees, e.g., a house can be *very big* or *extremely big*. Often, however, completely different adjectives refer to varying degrees on the same scale, e.g., *huge*, *gigantic*, *gargantuan*. Even adjectives like *enormous* (or *superb*, *impossible*) that are considered non-gradable from a syntactic perspective can be placed on a such a scale.

5.1.1.2 Intensity Patterns

Linguistic studies have found lexical patterns like '$\star$ but not $\star$' (e.g., *good but not great*) to reveal order information between a pair of adjectives (Sheinman and Tokunaga, 2009). We assume that we have two sets of lexical patterns that allow us to infer the most likely ordering between two

Weak-Strong Patterns	Strong-Weak Patterns
$\star$ (,) but not $\star$	not $\star$ (,) just $\star$
$\star$ (,) if not $\star$	not $\star$ (,) but just $\star$
$\star$ (,) although not $\star$	not $\star$ (,) still $\star$
$\star$ (,) though not $\star$	not $\star$ (,) but still $\star$
$\star$ (,) (and/or) even $\star$	not $\star$ (,) although still $\star$
$\star$ (,) (and/or) almost $\star$	not $\star$ (,) though still $\star$
not only $\star$ but $\star$	$\star$ (,) or very $\star$
not just $\star$ but $\star$	

Table 5.1: Ranking patterns used for computing intensity order Web scores. Among the patterns represented by the regular expressions above, we use only those that capture less than or equal to five words (to fit in the Google n-grams. Articles (*a, an, the*) are allowed to appear before the wildcards wherever possible.

(good, great)	(great, good)	(small, minute)
good , but not great $\rightarrow$ 24492.0	not great , just good $\rightarrow$ 248.0	small , almost minute $\rightarrow$ 97.0
good , if not great $\rightarrow$ 1912.0	great or very good $\rightarrow$ 89.0	small , even minute $\rightarrow$ 41.0
good , though not great $\rightarrow$ 504.0	not great but still good $\rightarrow$ 47.0	
good , or even great $\rightarrow$ 338.0		
not just good but great $\rightarrow$ 181.0		
good , almost great $\rightarrow$ 156.0		

Table 5.2: Some examples from the Web-scale corpus of useful intensity-based phrases on adjective pairs.

words when encountered in a corpus. A first pattern set, P_{ws}, contains patterns that reflect a weak-strong order between a pair of word (the first word is weaker than the second), and a second pattern set, P_{sw}, captures the strong-weak order. See Table 5.1 for the adjective patterns that we used in this work (and see Section 5.3.1 for implementation details regarding our pattern collection). Many of these patterns also apply to other parts of speech (e.g., ‘drizzle but not rain’, ‘running or even sprinting’), with significant discrimination on the Web in the right direction.

5.1.1.3 Pairwise Scores

Given an input set of words to be placed on a scale, we first collect evidence of their intensity order by using the above-mentioned intensity patterns and a large, Web-scale text corpus.

Previous work on information extraction from limited-sized raw text corpora revealed that coverage is often limited (Hearst, 1992; Hatzivassiloglou and McKeown, 1993). Some studies (Chklovski and Pantel, 2004; Sheinman and Tokunaga, 2009) used hit counts from an online search engine, but this is unstable and irreproducible (Kilgarriff, 2007). To avoid these issues, we use the largest available static corpus of counts, the Google n-grams corpus (Brants and Franz, 2006), which contains English n-grams ($n = 1$ to 5) and their observed frequency counts, generated from nearly 1 trillion word tokens and 95 billion sentences.

We consider each pair of words (a_1, a_2) in the input set in turn. For each pattern p in the two pattern sets (weak-strong P_{ws} and strong-weak P_{sw}), we insert the word pair into the pattern as $p(a_1, a_2)$ to get a phrasal *query* like "*big* but not *huge*". This is done by replacing the two wildcards in the pattern by the two words in order. Finally, we scan the Web n-grams corpus in a batch approach as described in Chapter 2 (Section 2.2), and collect frequencies of all our phrase queries. Table 5.2 depicts some examples of useful intensity-based phrase queries and their frequencies in the Web-scale corpus. We also collect frequencies for the input word unigrams and the patterns for normalization purposes. Given a word pair (a_1, a_2) and a corpus count function cnt, we define

$$\begin{aligned} W_1 &= \frac{1}{P_1} \sum_{p_1 \in P_{ws}} cnt(p_1(a_1, a_2)) \\ S_1 &= \frac{1}{P_2} \sum_{p_2 \in P_{sw}} cnt(p_2(a_1, a_2)) \\ W_2 &= \frac{1}{P_1} \sum_{p_1 \in P_{ws}} cnt(p_1(a_2, a_1)) \\ S_2 &= \frac{1}{P_2} \sum_{p_2 \in P_{sw}} cnt(p_2(a_2, a_1)) \end{aligned} \tag{5.1}$$

with

$$\begin{aligned} P_1 &= \sum_{p_1 \in P_{ws}} cnt(p_1) \\ P_2 &= \sum_{p_2 \in P_{sw}} cnt(p_2), \end{aligned} \tag{5.2}$$

such that the final overall *weak-strong score* is

$$score(a_1, a_2) = \frac{(W_1 - S_1) - (W_2 - S_2)}{cnt(a_1) \cdot cnt(a_2)}. \tag{5.3}$$

Here W_1 and S_1 represent Web evidence of a_1 and a_2 being in the weak-strong and strong-weak relation, respectively. W_2 and S_2 fit the reverse pair (a_2, a_1) in the patterns and hence represent the strong-weak and weak-strong relations, respectively, in the opposite direction. Hence, overall, $(W_1 - S_1) - (W_2 - S_2)$ represents the total weak-strong score of the pair (a_1, a_2), i.e. the score of a_1 being on the left of a_2 on a relative intensity scale, such that $score(a_1, a_2) = -score(a_2, a_1)$. The raw frequencies in the score are divided by counts of the patterns and by individual word unigram counts to obtain a pointwise mutual information (PMI) style normalization and hence avoid any bias in the score due to high-frequency patterns or word unigrams.[2]

5.1.2 Global Ordering with an MILP

5.1.2.1 Objective and Constraints

Given pairwise scores, we now aim at producing a global ranking of the input words that is much more informative than the original pairwise scores. Joint inference from multiple word pairs allows us to benefit from global information: Due to the sparsity of the pattern evidence, determining how two adjectives relate to each other can sometimes e.g., only be inferred by observing how each of them relate to some third adjective.

We assume that we are given N input words $A = a_1, \ldots, a_N$ that we wish to place on a linear scale, say $[0, 1]$. Thus each word a_i is to be assigned a position $x_i \in [0, 1]$ based on the pairwise weak-strong weights $score(a_i, a_j)$. A positive value for $score(a_i, a_j)$ means that a_i is supposedly weaker than a_j and hence we would like to obtain $x_i < x_j$. A negative value for $score(a_i, a_j)$ means that a_i is assumed to be stronger than a_j, so we would want to obtain $x_i > x_j$. Therefore, intuitively, our goal corresponds to maximizing the objective

$$\sum_{i,j} \text{sgn}(x_j - x_i) \cdot score(a_i, a_j) \tag{5.4}$$

Note that it is important to use the signum function sgn() here, because we only care about the relative order of x_i and x_j. Maximizing $\sum_{ij}(x_j - x_i) \cdot score(a_i, a_j)$ would lead to all words being placed at the edges of the scale, because the highest scores would dominate over all other ones. We do include the score magnitudes in the objective, because they help resolve contradictions in the pairwise scores (e.g., see Figure 5.1). This is discussed in more detail in Section 5.1.2.2.

In order to maximize this non-differentiable objective, we use Mixed Integer Linear Programming (MILP), a variant of linear programming in which some but not all of the variables are constrained to be integers. Using an MILP formalization, we can find a globally optimal solution in the joint decision space, and unlike previous work, we jointly exploit global information rather than just individual local (pairwise) scores. To encode the objective in a MILP, we need to introduce additional variables d_{ij}, w_{ij}, s_{ij} to capture the effect of the signum function, as explained below.

[2] In preliminary experiments on a development set, we also evaluated other intuitive forms of normalization.

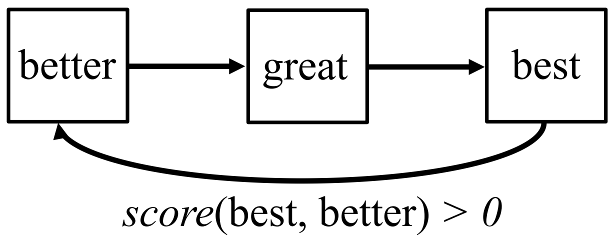


Figure 5.1: The input weak-strong data may contain one or more cycles, e.g., due to noisy patterns, so the final ranking will have to choose which input scores to honor and which to remove.

We additionally also enable our MILP to make use of any external equivalence (synonymy) information $E \subseteq \{1, \ldots, N\} \times \{1, \ldots, N\}$ that may be available. In this context, two words are considered synonymous if they are close enough in meaning to be placed on (almost) the same position in the intensity scale. If $(i, j) \in E$, we can safely assume that a_i, a_j have near-equivalent intensity, so we should encourage x_i, x_j to remain close to each other. The MILP is defined as follows:

maximize

$$\sum_{(i,j)\notin E} (w_{ij} - s_{ij}) \cdot score(a_i, a_j) - \sum_{(i,j)\in E} (w_{ij} + s_{ij})\, C$$

subject to

$$\begin{array}{ll}
d_{ij} = x_j - x_i & \forall i, j \in \{1, \ldots, N\} \\
d_{ij} - w_{ij}C \leq 0 & \forall i, j \in \{1, \ldots, N\} \\
d_{ij} + (1 - w_{ij})C > 0 & \forall i, j \in \{1, \ldots, N\} \\
d_{ij} + s_{ij}C \geq 0 & \forall i, j \in \{1, \ldots, N\} \\
d_{ij} - (1 - s_{ij})C < 0 & \forall i, j \in \{1, \ldots, N\} \\
x_i \in [0, 1] & \forall i \in \{1, \ldots, N\} \\
w_{ij} \in \{0, 1\} & \forall i, j \in \{1, \ldots, N\} \\
s_{ij} \in \{0, 1\} & \forall i, j \in \{1, \ldots, N\}
\end{array}$$

The difference variables d_{ij} simply capture differences between x_i, x_j. C is any very large constant greater than $\sum_{i,j} |score(a_i, a_j)|$; the exact value is irrelevant. The indicator variables w_{ij} and s_{ij} are jointly used to determine the value of the signum function $\text{sgn}(d_{ij}) = \text{sgn}(x_j - x_i)$. Variables w_{ij} become 1 if and only if $d_{ij} > 0$ and hence serve as indicator variables for weak-

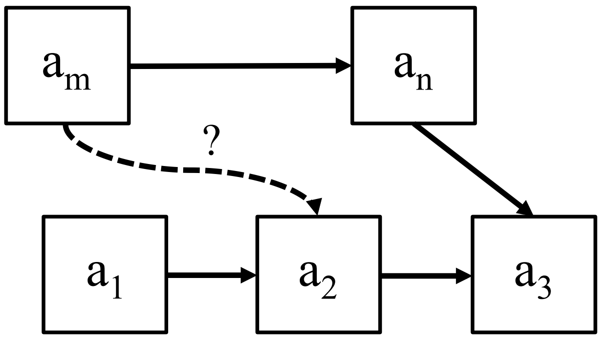


Figure 5.2: Equivalence Information: Knowing that a_m, a_2 are synonyms gives the MILP an indication of where to place a_n on the scale with respect to a_1, a_2, a_3.

strong relationships in the output. Variables s_{ij} become 1 if and only if $d_{ij} < 0$ and hence serve as indicator variables for a strong-weak relationship in the output. The objective encourages $w_{ij} = 1$ for $score(a_i, a_j) > 0$ and $s_{ij} = 1$ for $score(a_i, a_j) < 0$.[3] When equivalence (synonymy) information is available, then for $(i, j) \in E$ both $s_{ij} = 0$ and $w_{ij} = 0$ are encouraged.

5.1.2.2 Discussion

Our MILP uses intensity evidence of all input pairs together and assimilates all the scores via global transitivity constraints to determine the positions of the input words on a continuous real-valued scale. Hence, our approach addresses drawbacks of local or divide-and-conquer approaches, where adjectives are scored with respect to selected pivot words, and hence many adjectives that lack pairwise evidence with the pivots are not properly classified, although they may have order evidence with some third adjective that could help establish the ranking. Optional synonymy information can further help, as shown in Figure 5.2.

Moreover, our MILP also gives higher weight to pairs with higher scores, which is useful when breaking global constraint cycles as in the simple example in Figure 5.1. If we need to break a constraint violating triangle or cycle, we would have to make arbitrary choices if we were ranking based on sgn$(score(a, b))$ alone. Instead, we can choose a better ranking based on the magnitude of the pairwise scores. A stronger score between an adjective pair doesn't necessarily mean that they should be further apart in the ranking. It means that these two words are attested together on the Web with respect to the intensity patterns more than with other candidate words. Therefore, we try to respect the order of such word pairs more in the final ranking when we are breaking constraint-violating cycles.

[3]In order to avoid numeric instability issues due to very small $score(a_i, a_j)$ values after frequency normalization, in practice we have found it necessary to rescale them by a factor of 1 over the smallest $|score(a_i, a_j)| > 0$.

5.2 Related Work

Hatzivassiloglou and McKeown (1993) presented the first step towards automatic identification of adjective scales, thoroughly discussing the background of adjective semantics and a means of discovering clusters of adjectives that belong on the same scale, thus providing one way of creating the input for our ranking algorithm.

Inkpen and Hirst (2006) study near-synonyms and nuances of meaning differentiation (such as stylistic, attitudinal, etc.). They attempt to automatically acquire a knowledge base of near-synonym differences via an unsupervised decision-list algorithm. However, their method depends on a special dictionary of synonym differences to learn the extraction patterns, while we use only a raw Web-scale corpus.

Mohammad et al. (2013) proposed a method of identifying whether two adjectives are antonymous. This problem is related but distinct, because the degree of antonymy does not necessarily determine their position on an intensity scale. Antonyms (e.g., *little*, *big*) are not necessarily on the extreme ends of scales.

Sheinman and Tokunaga (2009) and Sheinman et al. (2012) present the most closely related previous work on adjective intensities. They collect lexico-semantic patterns via bootstrapping from seed adjective pairs to obtain pairwise intensities, albeit using search engine 'hits', which are unstable and problematic (Kilgarriff, 2007). While their approach is primarily evaluated in terms of a local pairwise classification task, they also suggest the possibility of ordering adjectives on a scale using a pivot-based partitioning approach. Although intuitive in theory, the extracted pairwise scores are frequently too sparse for this to work. Thus, many adjectives have no score with a particular headword. In our experiments, we reimplemented this approach and show that our MILP method improves over it by allowing individual pairwise decisions to benefit more from global information. Schulam and Fellbaum (2010) apply the approach of Sheinman and Tokunaga (2009) to German adjectives. Our method extends easily to various foreign languages as described in Section 5.4.

Another related task is the extraction of lexico-syntactic and lexico-semantic intensity-order patterns from large text corpora. The seminal work by Hearst (1992) extracted patterns for the hypernymy and meronymy relations. Later work (Chklovski and Pantel, 2004; Yang and Su, 2007; Davidov and Rappoport, 2008b; Turney, 2008; Tandon and de Melo, 2010) explored similar patterns for various other relations such as verb strengths, synonymy, antonymy, and coreference. Sheinman and Tokunaga (2009) follows Davidov and Rappoport (2008b) to automatically bootstrap adjective scaling patterns using seed adjectives and Web hits. These methods thus can be used to provide the input patterns for our algorithm.

VerbOcean by Chklovski and Pantel (2004) extracts various fine-grained semantic relations (including the stronger-than relation) between pairs of verbs, using lexico-syntactic patterns over the Web. Our approach of jointly ranking a set of words using pairwise evidence is also applicable to the VerbOcean pairs, and should help address similar sparsity issues of local pairwise decisions. Such scales will again be quite useful for language learners and language understanding tools.

Marneffe et al. (2010) infer yes-or-no answers to questions with responses involving scalar adjectives in a dialogue corpus. They correlate adjectives with ratings in a movie review corpus to find that *good* appears in lower-rated reviews than *excellent*.

Finally, there has been a lot of work on measuring the general sentiment polarity of words (Hatzivassiloglou and McKeown, 1997; Hatzivassiloglou and Wiebe, 2000; Turney and Littman, 2003; Liu and Seneff, 2009; Taboada et al., 2011; Yessenalina and Cardie, 2011; Pang and Lee, 2008). Our work instead aims at producing a large, unrestricted number of individual intensity scales for different qualities and hence can help in fine-grained sentiment analysis with respect to very particular content aspects.

5.3 Experiments

5.3.1 Data

Input Clusters In order to obtain input clusters for evaluation, we started out with the satellite cluster or 'dumbbell' structure of adjectives in WordNet 3.0, which consists of two direct antonyms as the poles and a number of other satellite adjectives that are semantically similar to each of the poles (Gross and Miller, 1990). For each antonymy pair, we determined an extended dumbbell set by looking up synonyms and words in related (satellite adjective and see-also) synonym sets. We cut such an extended dumbbell into two antonymous halves and treated each of these halves as a potential input adjective cluster.

Most of these WordNet clusters are noisy for the purpose of our task, i.e. they contain adjectives that appear unrelatable on a single scale due to polysemy and semantic drift, e.g., *violent* with respect to *supernatural* and *affected*. Motivated by Sheinman and Tokunaga (2009), we split such hard-to-relate adjectives into smaller scale-specific subgroups using the corpus evidence[4]. For this, we consider an undirected edge between each pair of adjectives that has a non-zero intensity score (based on the Web-scale scoring procedure described in Section 5.1.1.3). The resulting graph is then partitioned into connected components such that any adjectives in a subgraph are at least indirectly connected via some path and thus much more likely to belong to the same intensity scale. While this does break up partitions whenever there is no corpus evidence connecting them, ordering the adjectives within each such partition remains a challenging task. This is because the Web evidence will still not necessarily directly relate all adjectives (in a partition) to each other. Additionally, the Web evidence may still indicate the wrong direction. Figure 5.3 shows the size distribution of the resulting partitions.

Patterns To construct our intensity pattern set, we started with a couple of common rankable adjective seed pairs such as (*good, great*) and (*hot, boiling*) and used the Web-scale n-grams corpus (Brants and Franz, 2006) to collect the few most frequent patterns between and around these seed-pairs (in both directions). Among these, we manually chose a small set of intuitive patterns that are linguistically useful for ordering adjectives, several of which had not been discovered in previous

[4]Note that we do not use the WordNet dataset of Sheinman and Tokunaga (2009) for evaluation, as it does not provide full scales. Instead, their annotators only made pairwise comparisons with select words, using a 5-way classification scheme (*neutral*, *mild*, *very mild*, *intense*, *very intense*).

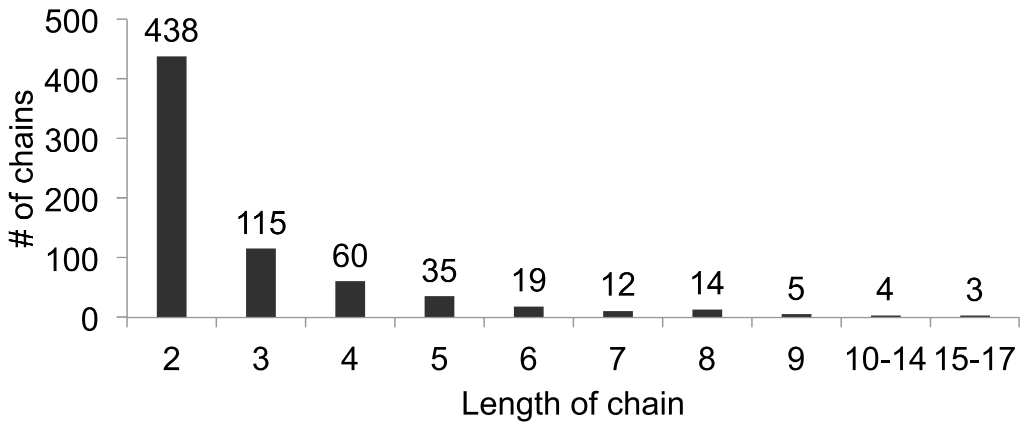


Figure 5.3: The histogram of the adjective cluster sizes after partitioning.

work. These are shown in Table 5.1. Note that we only collected patterns that were not ambiguous in the two orders, for example the pattern '$\star$ *, not* $\star$' is ambiguous because it can be used as both '*good, not great*' and '*great, not good*'. Alternatively, one can easily also use fully-automatic bootstrapping techniques based on seed word pairs (Hearst, 1992; Chklovski and Pantel, 2004; Yang and Su, 2007; Turney, 2008; Davidov and Rappoport, 2008b). However, our semi-automatic approach is a simple and fast process that extracts a small set of high-quality and very general adjective-scaling patterns. This process can quickly be repeated from scratch in any other language. Moreover, as described in Section 5.4.1, the English patterns can also be projected automatically to patterns in other languages.

Development and Test Sets Section 5.1.1 describes the method for collecting the intensity scores for adjective pairs, using Web-scale n-grams (Brants and Franz, 2006). We relied on a small development set to test the MILP structure and the pairwise score setup. For this, we manually chose 5 representative adjective clusters from the full set of clusters.

The final test set, distinct from this development set, consists of 569 word pairs in 88 clusters, each annotated by two native speakers of English. To arrive at this data, we randomly drew 30 clusters each for cluster sizes 3, 4, and 5+ from the histogram of partitioned adjective clusters in Figure 5.3. While labeling a cluster, annotators could exclude words that they deemed unsuitable to fit on a single shared intensity scale with the rest of the cluster. Fortunately, the partitioning described earlier had already separated most such cases into distinct clusters. The annotators ordered the remaining words on a scale. Words that seemed indistinguishable in strength could share positions in their annotation.

As our goal is to compare scale formation algorithms, we did not include trivial clusters of size

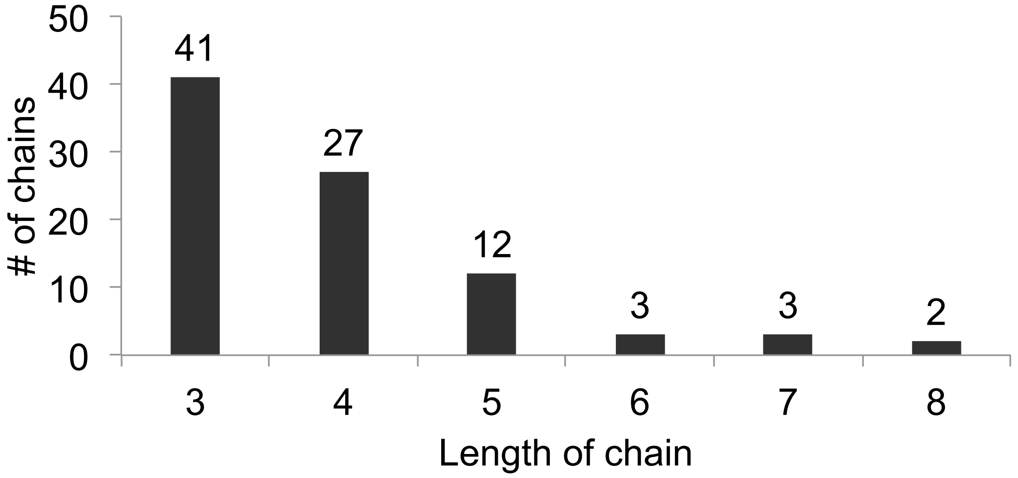


Figure 5.4: The histogram of adjective cluster sizes in the test set.

2. On such trivial clusters, the Web evidence alone determines the output and hence all algorithms, including the baseline, obtain the same pairwise accuracy (defined below) of 93.3% on a separate set of 30 random clusters of size 2.

Figure 5.4 shows the distribution of cluster sizes in our main gold set. The inter-annotator agreement in terms of Cohen's κ (Cohen, 1960) on the pairwise classification task with 3 labels (weaker, stronger, or equal/unknown) was 0.64. In terms of pairwise accuracy, the agreement was 78.0%.

5.3.2 Metrics

In order to thoroughly evaluate the performance of our adjective ordering procedure, we rely on both pairwise and ranking-correlation evaluation metrics. Consider a set of input words $A = \{a_1, a_2, \ldots, a_n\}$ and two rankings for this set – a gold-standard ranking $r_G(A)$ and a predicted ranking $r_P(A)$.

5.3.2.1 Pairwise Accuracy

For a pair of words a_i, a_j, we may consider the classification task of choosing one of three labels ($<$, $>$, $=?$) for the case of a_i being weaker, stronger, and equal (or unknown) in intensity, respectively, compared to a_2:

$$L(a_1, a_2) = \begin{cases} < & \text{if } \; r(a_i) < r(a_j) \\ > & \text{if } \; r(a_i) > r(a_j) \\ =? & \text{if } \; r(a_i) = r(a_j) \end{cases}$$

For each pair (a_1, a_2), we compute gold-standard labels $L_G(a_1, a_2)$ and predicted labels $L_P(a_1, a_2)$ as above, and then the pairwise accuracy $PW(A)$ for a particular ordering on A is simply the fraction of pairs that are correctly classified, i.e. for which the predicted label is same as the gold-standard label:

$$PW(A) = \frac{\sum\limits_{i<j} \mathbf{1}\{L_G(a_i, a_j) = L_P(a_i, a_j)\}}{\sum\limits_{i<j} 1}$$

5.3.2.2 Ranking Correlation Coefficients

Our second type of evaluation assesses the rank correlation between two ranking permutations (gold-standard and predicted). Many studies use *Kendall's tau* (Kendall, 1938), which measures the total number of pairwise inversions, while others prefer *Spearman's rho* (Spearman, 1904), which measures the L1 distance between ranks.

Kendall's tau correlation coefficient We use the τ_b version of Kendall's correlation metric, as it incorporates a correction for ties (Kruskal, 1958; Dou et al., 2008):

$$\tau_b = \frac{P - Q}{\sqrt{(P + Q + X_0) \cdot (P + Q + Y_0)}}$$

where P is the number of concordant pairs, Q is the number of discordant pairs, X_0 is the number of pairs tied in the first ranking, Y_0 is the number of pairs tied in the second ranking. Given the two rankings of an adjective set A, the gold-standard ranking $r_G(A)$ and the predicted ranking $r_P(A)$, two words a_i, a_j are:

- *concordant* iff both rankings have the same strict order of the two elements, i.e., $r_G(a_i) > r_G(a_j)$ and $r_P(a_i) > r_P(a_j)$, or $r_G(a_i) < r_G(a_j)$ and $r_P(a_i) < r_P(a_j)$.
- *discordant* iff the two rankings have an inverted strict order of the two elements, i.e., $r_G(a_i) > r_G(a_j)$ and $r_P(a_i) < r_P(a_j)$, or $r_G(a_i) < r_G(a_j)$ and $r_P(a_i) > r_P(a_j)$.
- *tied* iff $r_G(a_i) = r_G(a_j)$ or $r_P(a_i) = r_P(a_j)$.

Spearman's rho correlation coefficient For two n-sized ranked lists $\{x_i\}$ and $\{y_i\}$, the Spearman correlation coefficient is defined as the Pearson correlation coefficient between the ranks of variables:

$$\rho = \frac{\sum\limits_i (x_i - \bar{x}) \cdot (y_i - \bar{y})}{\sqrt{\sum\limits_i (x_i - \bar{x})^2 \cdot \sum\limits_i (y_i - \bar{y})^2}}$$

Here, $\bar{x}$ and $\bar{y}$ denote the means of the values in the respective lists. We use the standard procedure for handling ties correctly. Tied values are assigned the average of all ranks of items sharing the same value in the ranked list sorted in ascending order of the values.

Handling Inversions While annotating, we sometimes observed that the ordering itself was very clear but the annotators disagreed about which end of a particular scale was to count as the strong one, e.g., when transitioning from *soft* to *hard* or from *alpha* to *beta*. We thus also report average *absolute* values of both correlation coefficients, as these properly account for anticorrelations. Our test set only contains clusters of size 3 or larger, so there is no need to account for inversions in clusters of size 2.

5.3.2.3 Inter-annotator agreement

To measure the agreement between human annotators, we additionally report the standard Cohen's kappa coefficient (Cohen, 1960) on the pairwise classification task with 3 labels (as described above):

$$\kappa = \frac{\text{Pr}(a) - \text{Pr}(e)}{1 - \text{Pr}(e)}$$

where Pr(a) is the relative observed agreement among raters, and Pr(e) is the hypothetical probability of chance agreement, using the observed data to calculate the probabilities of each observer randomly selecting each category.

5.3.3 Results

In Table 5.3, we use the evaluation metrics mentioned above to compare several different approaches.

Web Baseline The first baseline simply reflects the original pairwise Web-based intensity scores. We classify (with one of 3 labels) a given pair of adjectives using the Web-based intensity scores (as described in Section 5.1.1.3) as follows:

Method	Pairwise Accuracy	Avg. τ	Avg. $\|\tau\|$	Avg. ρ	Avg. $\|\rho\|$
Web Baseline	48.2%	N/A	N/A	N/A	N/A
Divide-and-Conquer	50.6%	0.45	0.53	0.52	0.62
Sheinman and Tokunaga (2009)	55.5%	N/A	N/A	N/A	N/A
MILP	69.6%	0.57	0.65	0.64	0.73
MILP with synonymy	78.2%	0.57	0.66	0.67	0.80
Inter-Annotator Agreement	78.0%	0.67	0.76	0.75	0.86

Table 5.3: The main test results on global intensity ordering for adjectives.

$$L_{baseline}(a_1, a_2) = \begin{cases} < & \text{if} \quad score(a_i, a_j) > 0 \\ > & \text{if} \quad score(a_i, a_j) < 0 \\ =? & \text{if} \quad score(a_i, a_j) = 0 \end{cases}$$

Since $score(a_i, a_j)$ represents the weak-strong score of the two adjectives, a more positive value means a higher likelihood of a_i being weaker (<, on the left) in intensity than a_j.

In Table 5.3, we observe that the (micro-averaged) pairwise accuracy, as defined earlier, for the original Web baseline is 48.2%, while the ranking measures are undefined because the individual pairs do not lead to a coherent scale.

Divide-and-Conquer The divide-and-conquer baseline recursively splits a set of words into three subgroups, placed to the left (weaker), on the same position (no evidence), or to the right (stronger) of a given randomly chosen pivot word.

While this approach shows only a minor improvement in terms of the pairwise accuracy (50.6%), its main benefit is that one obtains well-defined intensity scales rather than just a collection of pairwise scores.

Sheinman and Tokunaga The approach by Sheinman and Tokunaga (2009) involves a similar divide-and-conquer based partitioning in the first phase, except that their method makes use of synonymy information from WordNet and uses all synonyms in WordNet's synset for the headword as neutral pivot elements (if the headword is not in WordNet, then the word with the maximal unigram frequency is chosen). In the second phase, their method performs pairwise comparisons within the more intense and less intense subgroups. We reimplement their approach here, using the Google N-Grams dataset instead of online Web search engine hits. We observe a small improvement over the Web baseline in terms of pairwise accuracy. Note that the rank correlation measure scores are undefined for their approach. This is because in some cases their method placed all words on the same position in the scale, which these measures cannot handle even in their tie-corrected versions.

		Predicted Class		
		Weaker	Tie	Stronger
True Class	Weaker	117	127	15
	Tie	5	42	15
	Stronger	11	122	115

Table 5.4: The confusion matrix for the Web baseline output.

		Predicted Class		
		Weaker	Tie	Stronger
True Class	Weaker	177	29	53
	Tie	9	24	29
	Stronger	15	38	195

Table 5.5: The confusion matrix for the MILP output.

Overall, the Sheinman and Tokunaga approach does not aggregate information sufficiently well at the global level and often fails to make use of transitive inference.

MILP Our MILP exploits the same pairwise scores to induce significantly more accurate pairwise labels with 69.6% accuracy, a 41% relative error reduction over the Web baseline, 38% over Divide-and-Conquer, and 32% over Sheinman and Tokunaga (2009). We further see that our MILP method is able to exploit external synonymy (equivalence) information (using synonyms marked by the annotators). The accuracy of the pairwise scores as well as the quality of the overall ranking increase even further to 78.2%, approaching the human inter-annotator agreement. In terms of average correlation coefficients, we observe similar improvement trends from the MILP, but of different magnitudes, because these averages give small clusters the same weight as larger ones.

5.3.4 Analysis

Confusion Matrices For a given approach, we can study the confusion matrix obtained by cross-tabulating the gold classification with the predicted classification of every unique pair of adjectives in the ground truth data. Table 5.4 shows the confusion matrix for the Web baseline. We observe that due to the sparsity of pairwise intensity order evidence, the baseline method predicts too many ties.

Table 5.5 provides the confusion matrix for the MILP (without external equivalence information) for comparison. Although the middle column still shows that the MILP predicts more ties

Accuracy	Prediction	Gold Standard
Good	hard < painful < hopeless	hard < painful < hopeless
	full < stuffed < (overflowing, overloaded)	full < stuffed < overflowing < overloaded
	unusual < uncommon < rare < exceptional < extraordinary	uncommon < unusual < rare < extraordinary < exceptional
Average	creepy < scary < sinister < frightening < terrifying	creepy < (scary, frightening) < terrifying < sinister
Bad	(awake, conscious) < alive < aware	alive < awake < (aware, conscious)
	strange < (unusual, weird) < (funny, eerie)	(strange, funny) < unusual < weird < eerie

Table 5.6: Some examples (of bad, average and good accuracy) of our MILP predictions (without synonymy information) and the corresponding gold-standard annotation.

than humans annotators, we find that a clear majority of all unique pairs are now correctly placed along the diagonal. This confirms that our MILP successfully infers new ordering decisions, although it uses the same input (corpus evidence) as the baseline. The remaining ties are mostly just the result of pairs for which there simply is no evidence at all in the input Web counts. Note that this problem could for instance be circumvented by relying on a crowdsourcing approach: A few dispersed tie-breakers are enough to allow our MILP to correct many other predictions.

Weak-Strong Patterns		Strong-Weak Patterns	
English	German	English	German
⋆ but not ⋆	⋆ aber nicht ⋆	not ⋆ just ⋆	nicht ⋆ gerade ⋆
⋆ if not ⋆	⋆ wenn nicht ⋆	not ⋆ but just ⋆	nicht ⋆ aber nur ⋆
⋆ and almost ⋆	⋆ und fast ⋆	not ⋆ though still ⋆	nicht ⋆ aber immer noch ⋆
not just ⋆ but ⋆	nicht nur ⋆ sondern ⋆	⋆ or very ⋆	⋆ oder sehr ⋆

Table 5.7: Examples of German intensity patterns projected (translated) directly from the English patterns.

Predicted Examples Finally, in Table 5.6, we provide a selection of real results obtained by our algorithm. For instance, it correctly inferred that *terrifying* is more intense than *creepy* or *scary*, although the Web pattern counts did not provide any explicit information about these words pairs. In some cases, however, the Web evidence did not suffice to draw the right conclusions, or it was misleading due to issues like polysemy (as for the word *funny*).

While we show results on gold-standard chains here for evaluation purposes, in practice one can also recombine two $[0, 1]$ chains for a pair of antonymic clusters to form a single scale from $[-1, 1]$ that visualizes the full spectrum of available adjectives along a dimension, from *adjacent* all the way to *removed*, or from *black* to *glaring*.

5.4 Extension to Multilingual Ordering

Our method for globally ordering words on a scale can easily be applied to languages other than English. The entire process is language-independent as long as the required resources are available and a small number of patterns are chosen. For morphologically rich languages, the information extraction step of course may require additional morphological analysis tools for stemming and aggregating frequencies across different forms.

Alternatively, a cross-lingual projection approach is possible at multiple levels, utilizing information from the English data and ranking. As the first step, the set of words in the target language that we wish to rank can be projected from the English word set if necessary – e.g., as shown in Melo and Weikum (2009). Next, we outline two projection methods for the ordering step. The first method is based on projection of the English intensity-ordering patterns to the new language, and then using the same MILP as described in Section 5.1.2. In the second method, we also change the MILP and add cross-lingual constraints to better inform the target language's adjective ranking. A detailed empirical evaluation of these approaches remains future work.

5.4.1 Cross-lingual Pattern Projection

Instead of creating new patterns, in many cases we obtain quite adequate intensity patterns by using cross-lingual projection. We simply take several adjective pairs, instantiate the English patterns with them, and obtain new patterns using a machine translation system. Filling the wildcards in a pattern, say '$\star$ but not $\star$', with *good/excellent* results in 'good but not excellent'. This phrase is then translated into the target language using the translation system, say into German 'gut aber nicht ausgezeichnet'. Finally, put back the wildcards in the place of the translations of the adjective words, here *gut* and *ausgezeichnet*, to get the corresponding German pattern '$\star$ aber nicht $\star$'. Table 5.7 shows various German intensity patterns that we obtain by projecting from the English patterns as described. The process is repeated with multiple adjective pairs in case different variants are returned, e.g., due to morphology. Most of these translations deliver useful results.

Now that we have the target language adjectives and the ranking patterns, we can compute the pairwise intensity scores using large-scale data in that language. We can use the Google n-grams corpora for 10 European languages (Brants and Franz, 2009), and also for Chinese (LDC2010T02) and Japanese (LDC2009T08). For other languages, one can use available large raw-text corpora or Web crawling tools.

5.4.2 Cross-lingual MILP

To improve the rankings for lesser-resourced languages, we can further use a joint MILP approach for the new language we want to transfer this process to. Additional constraints between the English words and their corresponding target language translations, in combination with the English ranking information, allow the algorithm to obtain better rankings for the target words whenever the non-English target language corpus does not provide sufficient intensity order evidence.

In this case, the input set A contains words in multiple languages. The Web intensity scores $score(a_i, a_j)$ should be set to zero when comparing words across languages. We instead link them using a translation table $T \subseteq \{1, \ldots, N\} \times \{1, \ldots, N\}$ from a translation dictionary or phrase table. Here, $(i, j) \in T$ signifies that a_i is a translation of a_j. We do not require a bijective relationship between them (i.e., translations needn't be unique). The objective function is augmented by adding the new term

$$\sum_{(i,j)\in T} (w'_{ij} + s'_{ij})C_T \tag{5.5}$$

for a constant $C_T > 0$ that determines how much weight we assign to translations as opposed to the corpus count scores. The MILP is extended by adding the following extra constraints.

$$\begin{array}{ll}
d_{ij} - w'_{ij} C_T < -d_{\max} & \forall i,j \in \{1,\ldots,N\} \\
d_{ij} + (1 - w'_{ij}) C_T \geq -d_{\max} & \forall i,j \in \{1,\ldots,N\} \\
d_{ij} + s'_{ij} C_T > d_{\max} & \forall i,j \in \{1,\ldots,N\} \\
d_{ij} - (1 - s'_{ij}) C_T \leq d_{\max} & \forall i,j \in \{1,\ldots,N\} \\
w'_{ij} \in \{0,1\} & \forall i,j \in T \\
s'_{ij} \in \{0,1\} & \forall i,j \in T
\end{array}$$

The variables $d_{i,j}$, as before, encode distances between positions of words on the scale, but now also include cross-lingual pairs of words in different languages. The new constraints encourage translational equivalents to remain close to each other, preferably within a desired (but not strictly enforced) maximum distance $d_{\max}$. The new variables w'_{ij}, s'_{ij} are similar to w_{ij}, s_{ij} in the standard MILP. However, the w'_{ij} become 1 if and only if $d_{ij} \geq -d_{\max}$ and the s'_{ij} become 1 if and only if $d_{ij} \leq d_{\max}$. If both w'_{ij} and s'_{ij} are 1, then the two words have a small distance $-d_{\max} \leq d_{ij} \leq d_{\max}$. The augmented objective function explicitly encourages this for translational equivalents. Overall, this approach thus allows evidence from a language with more Web evidence to improve the process of adjective ordering in lesser-resourced languages.

Chapter 6

Conclusions

This thesis has discussed the idea of incorporating world knowledge in structured NLP tasks via surface features from Web-scale corpora, especially a large n-gram corpus. We have found that such shallow features are effective in resolving various complex semantic ambiguities in syntactic parsing, coreference resolution, taxonomy induction, and intensity ordering.

When investigating syntactic ambiguities, our Web features capture useful cases where local surface counts and context words are good indicators of attachments, e.g., an adverb next to a preposition, a preposition next to a noun, a pronoun such as *it* between a verb and a preposition, a capitalized determiner such as *The* before a noun and preposition, and so on. Our automatically learned, highest-weight features rediscover most of the hand-designed cues discussed in previous work and also discover various new, intuitive cues. Similarly, Web features for coreference resolution are successful in correctly determining the referent of rare proper names, e.g., *Mulcahy* and *athletic director*, *Barry Bonds* and *the best baseball player*, etc. For the task of taxonomy induction, our probabilistic, heterogeneous model learns a suite of useful hypernymy and siblinghood features from the Web, e.g., *C and other P*, $> P > C$, *either* C_1 *or* C_2, and C_1 *and / or* C_2. Finally, patterns like *good but not great*, *not excellent but still great*, and *excellent or very good*, and their Web statistics capture pairwise ordering of near-synonyms like *good*, *great*, and *excellent*, which when incorporated into a global inference method, achieves accurate semantic intensity orderings.

However, there are still various challenges that need to be addressed. For instance, we use an n-gram version of the Web which has a major drawback of limiting us to short-distance queries (which are of size 5 in our case). This especially becomes as issue when we are working with long, multi-word terms and phrases, because these will usually not fit in 5-grams. For syntactic parsing and coreference resolution, we chose to work with shorter representations of the full phrase, e.g., head words in this case. However, for the task of taxonomy induction, we need to work with the exact full term, and hence, we included an additional, full-sentence dataset such as Wikipedia. Therefore, adding broader, deeper, and larger Web-scale resources so as to overcome short-distance limits and sparsity, is an important future direction.

Even more crucial is the handling of out-of-domain and specialized-domain scenarios in the various semantic facets that we addressed. These scenarios bring in new challenges because rare languages and domains seldom have large datasets to extract surface cues from, and hence, one might benefit from domain adaptation techniques. Moreover, deeper semantic ambiguities such as those in detection of entailment (*buy* $\rightarrow$ *own*, *murdered* $\rightarrow$ *dead*), finer-grained -nymy relations (e.g., meronymy and troponymy), metaphors and idioms, pragmatics (meaning in context), and grounded implicature (*run* $\rightarrow$ *increased heart-rate, move from A to B*) will need new world knowledge cues and models, and these are exciting next directions.

Bibliography

Michaela Atterer and Hinrich Schutze (2007). Prepositional phrase attachment without oracles. In *Computational Linguistics* 33(4), pp. 469–476.

Amit Bagga and Breck Baldwin (1998). Algorithms for scoring coreference chains. In *Proceedings of the International Conference on Language Resources and Evaluation Workshop on Linguistics Coreference (LREC MUC)*.

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. (1998). The Berkeley Framenet project. In *Proceedings of the 17th International Conference on Computational Linguistics and the 36th annual meeting of the Association for Computational Linguistics (COLING-ACL)*.

Mohit Bansal and Dan Klein (2011). Web-scale features for full-scale parsing. In *Proceedings of the Annual Meeting on Association for Computational Linguistics (ACL)*.

Mohit Bansal and Dan Klein (2012). Coreference semantics from Web features. In *Proceedings of the Annual Meeting on Association for Computational Linguistics (ACL)*.

Mohit Bansal, David Burkett, Gerard De Melo, and Dan Klein (2013). Structured learning for taxonomy induction with belief propagation. In *submission*.

Marco Baroni, Brian Murphy, Eduard Barbu, and Massimo Poesio (2010). Strudel: A corpus-based semantic model based on properties and types. In *Cognitive Science* 34.2, pp. 222–254.

Eric Bengtson and Dan Roth (2008). Understanding the value of features for coreference resolution. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Shane Bergsma and Dekang Lin (2006). Bootstrapping path-based pronoun resolution. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics (COLING-ACL)*.

Shane Bergsma, Emily Pitler, and Dekang Lin (2010). Creating robust supervised classifiers via Web-scale n-gram data. In *Proceedings of the Annual Meeting on Association for Computational Linguistics (ACL).*

Thorsten Brants and Alex Franz (2006). The Google Web 1T 5-gram corpus version 1.1. In *LDC2006T13*.

Thorsten Brants and Alex Franz (2009). Web 1T 5-gram, 10 European languages, version 1. In *LDC2009T25*.

Eugene Charniak and Mark Johnson (2005). Coarse-to-fine n-best parsing and MaxEnt discriminative reranking. In *Proceedings of the Annual Meeting on Association for Computational Linguistics (ACL).*

Timothy Chklovski and Patrick Pantel (2004). VerbOcean: mining the Web for fine-grained semantic verb relations. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP).*

Yoeng-Jin Chu and Tseng-Hong Liu (1965). On the shortest arborescence of a directed graph. In *Science Sinica* 14.1396-1400, p. 270.

Kenneth Ward Church and Patrick Hanks (1989). Word association norms, mutual information, and lexicography. In *Proceedings of the Annual Meeting on Association for Computational Linguistics (ACL).*

Philipp Cimiano, Andreas Hotho, and Steffen Staab (2005). Learning concept hierarchies from text corpora using formal concept analysis. In *Journal of Artificial Intelligence Research* 24.1, pp. 305–339.

Jacob Cohen (1960). A coefficient of agreement for nominal scales. In *Educational and Psychological Measurement* 20(1), pp. 37–46.

Michael Collins (1999). Head-driven statistical models for natural language parsing. In *Ph.D. thesis, University of Pennsylvania, Philadelphia.*

Michael Collins (2000). Discriminative reranking for natural language parsing. In *Proceedings of the International Conference on Machine Learning (ICML).*

Michael Collins (2002). Discriminative training methods for Hidden Markov Models: Theory and experiments with perceptron algorithms. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP).*

Michael Collins (2003). Head-driven statistical models for natural language parsing. In *Computational linguistics* 29.4, pp. 589–637.

Michael Collins and Terry Koo (2005). Discriminative reranking for natural language parsing. In *Computational Linguistics* 31(1), pp. 25–70.

Hal Daumé III and Daniel Marcu (2005). A large-scale exploration of effective global features for a joint entity detection and tracking model. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP).*

Dmitry Davidov and Ari Rappoport (2006). Efficient unsupervised discovery of word categories using symmetric patterns and high frequency words. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics (COLING-ACL).*

Dmitry Davidov and Ari Rappoport (2008a). Classification of semantic relationships between nominals using pattern clusters. In *Proceedings of the Annual Meeting on Association for Computational Linguistics (ACL).*

Dmitry Davidov and Ari Rappoport (2008b). Unsupervised discovery of generic relationships using pattern clusters and its evaluation by automatically generated sat analogy questions. In *Proceedings of the Annual Meeting on Association for Computational Linguistics (ACL).*

Gerard De Melo and Mohit Bansal (2013). Good, great, excellent: Global inference of semantic intensities. In *Proceedings of the Transactions of the Association for Computational Linguistics (TACL).*

Q.X. Do and D. Roth (2010). Constraints based taxonomic relation classification. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP).*

Zhicheng Dou, Ruihua Song, Xiaojie Yuan, and Ji-Rong Wen (2008). Are click-through data adequate for learning Web search rankings? In *Proceedings of the ACM International Conference on Information and Knowledge Management (CIKM).*

John Duchi, Elad Hazan, and Yoram Singer (2011). Adaptive subgradient methods for online learning and stochastic optimization. In *The Journal of Machine Learning Research* 12, pp. 2121–2159.

Jack Edmonds (1967). Optimum branchings. In *Journal of Research of the National Bureau of Standards B* 71, pp. 233–240.

Bradley Efron and Robert Tibshirani (1993). An introduction to the bootstrap. In *Chapman & Hall CRC*.

Oren Etzioni, Michael Cafarella, Doug Downey, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S. Weld, and Alexander Yates (2005). Unsupervised named-entity extraction from the Web: An experimental study. In *Artificial Intelligence* 165.1, pp. 91–134.

David Ferrucci, Eric Brown, Jennifer Chu-Carroll, James Fan, David Gondek, Aditya A Kalyanpur, Adam Lally, J William Murdock, Eric Nyberg, John Prager, Nico Schlaefer, and Chris Welty (2010). Building Watson: An overview of the DeepQA project. In *AI magazine* 31.3, pp. 59–79.

Jenny Rose Finkel, Alex Kleeman, and Christopher D. Manning (2008). Efficient, feature-based, conditional random field parsing. In *Proceedings of the Annual Meeting on Association for Computational Linguistics (ACL)*.

Trevor Fountain and Mirella Lapata (2012). Taxonomy induction using hierarchical random graphs. In *Proceedings of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Leonidas Georgiadis (2003). Arborescence optimization problems solvable by Edmond's algorithm. In *Theoretical Computer Science* 301.1, pp. 427–437.

Roxana Girju, Adriana Badulescu, and Dan Moldovan (2003). Learning semantic constraints for the automatic discovery of part-whole relations. In *Proceedings of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Joshua Goodman (1996). Parsing algorithms and metrics. In *Proceedings of the Annual Meeting on Association for Computational Linguistics (ACL)*.

Derek Gross and Katherine J. Miller (1990). Adjectives in WordNet. In *International Journal of Lexicography* 3(4), pp. 265–277.

Aria Haghighi and Dan Klein (2009). Simple coreference resolution with rich syntactic and semantic features. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP).*

Aria Haghighi and Dan Klein (2010). Coreference resolution in a modular, entity-centered model. In *Proceedings of the Human Language Technologies and the North American Chapter of the Association for Computational Linguistics (HLT-NAACL).*

Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten (2009). The WEKA data mining software: An update. In *SIGKDD Explorations* 11(1).

Zellig Harris (1954). Distributional structure. In *Word* 10(23), 146162.

Vasileios Hatzivassiloglou and Kathleen R. McKeown (1993). Towards the automatic identification of adjectival scales: clustering adjectives according to meaning. In *Proceedings of the Annual Meeting on Association for Computational Linguistics (ACL).*

Vasileios Hatzivassiloglou and Kathleen R. McKeown (1997). Predicting the semantic orientation of adjectives. In *Proceedings of the Annual Meeting on Association for Computational Linguistics (ACL).*

Vasileios Hatzivassiloglou and Janyce M. Wiebe (2000). Effects of adjective orientation and gradability on sentence subjectivity. In *Proceedings of the International Conference on Computational Linguistics (COLING).*

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

Donald Hindle and Mats Rooth (1993). Structural ambiguity and lexical relations. In *Computational Linguistics* 19(1), 103120.

Eduard Hovy, Zornitsa Kozareva, and Ellen Riloff (2009). Toward completeness in concept extraction and classification. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP).*

Liang Huang (2008). Forest reranking: Discriminative parsing with non-local features. In *Proceedings of the Annual Meeting on Association for Computational Linguistics (ACL).*

Diana Inkpen and Graeme Hirst (2006). Building and using a lexical knowledge base of near-synonym differences. In *Computational Linguistics* 32(2), 223262.

Boris Katz and Jimmy Lin (2003). Selectively using relations to improve precision in question answering. In *Proceedings of the Workshop on NLP for Question Answering in the European Chapter of the Association for Computational Linguistics (EACL).*

Maurice G. Kendall (1938). A new measure of rank correlation. In *Biometrika* 30(1/2), pp. 81–93.

Adam Kilgarriff (2007). Googleology is bad science. In *Computational Linguistics* 33(1).

Hamidreza Kobdani, Hinrich Schutze, Michael Schiehlen, and Hans Kamp (2011). Bootstrapping coreference resolution using word associations. In *Proceedings of the Annual Meeting on Association for Computational Linguistics (ACL).*

Terry Koo, Xavier Carreras, and Michael Collins (2008). Simple semi-supervised dependency parsing. In *Proceedings of the Annual Meeting on Association for Computational Linguistics (ACL).*

Terry Koo and Michael Collins (2010). Efficient third-order dependency parsers. In *Proceedings of the Annual Meeting on Association for Computational Linguistics (ACL).*

Terry Koo, Amir Globerson, Xavier Carreras, and Michael Collins (2007). Structured prediction models via the Matrix-Tree theorem. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing and the Conference on Computational Natural Language Learning (EMNLP-CoNLL).*

Zornitsa Kozareva and Eduard Hovy (2010). A semi-supervised method to learn and construct taxonomies using the Web. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP).*

Zornitsa Kozareva, Ellen Riloff, and Eduard Hovy (2008). Semantic class learning from the Web with hyponym pattern linkage graphs. In *Proceedings of the Annual Meeting on Association for Computational Linguistics (ACL).*

William H. Kruskal (1958). Ordinal measures of association. In *Journal of the American Statistical Association* 53(284), pp. 814–861.

Frank R Kschischang, Brendan J Frey, and H-A Loeliger (2001). Factor graphs and the sum-product algorithm. In *Information Theory, IEEE Transactions on* 47.2, pp. 498–519.

Ni Lao, Amarnag Subramanya, Fernando Pereira, and William W. Cohen (2012). Reading the Web with learned syntactic-semantic inference rules. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Mirella Lapata and Frank Keller (2004). The Web as a baseline: Evaluating the performance of unsupervised Web-based models for a range of NLP tasks. In *Proceedings of the Human Language Technologies and the North American Chapter of the Association for Computational Linguistics (HLT-NAACL)*.

Mark Lauer (1995). Corpus statistics meet the noun compound: Some empirical results. In *Proceedings of the Annual Meeting on Association for Computational Linguistics (ACL)*.

Dekang Lin (1998). Automatic retrieval and clustering of similar words. In *Proceedings of the International Conference on Computational Linguistics (COLING)*.

Dekang Lin and Patrick Pantel (2002). Concept discovery from text. In *Proceedings of the International Conference on Computational Linguistics (COLING)*.

Dekang Lin and Xiaoyun Wu (2009). Phrase clustering for discriminative learning. In *Proceedings of the Annual Meeting on Association for Computational Linguistics (ACL)*.

Dekang Lin, Kenneth Church, Heng Ji, Satoshi Sekine, David Yarowsky, Shane Bergsma, Kailash Patil, Emily Pitler, Rachel Lathbury, Vikram Rao, Kapil Dalwani, and Sushant Narsale (2010). New tools for Web-scale n-grams. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC)*.

Jingjing Liu and Stephanie Seneff (2009). Review sentiment scoring via a parse-and-paraphrase paradigm. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

David J.C. MacKay (2003). Information theory, inference, and learning algorithms. In *Cambridge University Press*.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini (1993). Building a large annotated corpus of English: The Penn Treebank. In *Computational Linguistics*.

Katja Markert and Malvina Nissim (2005). Comparing knowledge sources for nominal anaphora resolution. In *Computational Linguistics* 31(3), pp. 367–402.

Katja Markert, Malvina Nissim, and Natalia N. Modjeska (2003). Using the Web for nominal anaphora resolution. In *Proceedings of the Workshop on the Computational Treatment of Anaphora in the European Chapter of the Association for Computational Linguistics (EACL).*

Marie-Catherine de Marneffe, Christopher D. Manning, and Christopher Potts (2010). Was it good? it was provocative. learning the meaning of scalar adjectives. In *Proceedings of the Annual Meeting on Association for Computational Linguistics (ACL).*

André F. T. Martins, Noah A. Smith, and Eric P. Xing (2009). Concise integer linear programming formulations for dependency parsing. In *Proceedings of the Annual Meeting on Association for Computational Linguistics and the International Joint Conference on Natural Language Processing (ACL-IJCNLP).*

Ryan McDonald, Koby Crammer, and Fernando Pereira (2005). Online large-margin training of dependency parsers. In *Proceedings of the Annual Meeting on Association for Computational Linguistics (ACL).*

Ryan McDonald and Fernando Pereira (2006). Online learning of approximate dependency parsing algorithms. In *Proceedings of the European Chapter of the Association for Computational Linguistics (EACL).*

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič (2005). Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of the Human Language Technologies and the North American Chapter of the Association for Computational Linguistics (HLT-NAACL).*

Gerard de Melo and Gerhard Weikum (2009). Towards a universal WordNet by learning from combined evidence. In *Proceedings of the ACM International Conference on Information and Knowledge Management (CIKM).* ISBN: 978-1-60558-512-3. DOI: `http://doi.acm.org/10.1145/1645953.1646020`.

George A. Miller (1995). WordNet: A lexical database for English. In *Communications of the ACM* 38(11), pp. 39–41.

Natalia N. Modjeska, Katja Markert, and Malvina Nissim (2003). Using the Web in machine learning for other-anaphora resolution. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP).*

Said M. Mohammad, Bonnie J. Dorr, Graeme Hirst, and Peter D. Turney (2013). Computing lexical contrast. In *Computational Linguistics*.

Kevin P. Murphy, Yair Weiss, and Michael I. Jordan (1999). Loopy belief propagation for approximate inference: An empirical study. In *Proceedings of the Conference on Uncertainty in Articial Intelligence (UAI)*.

Preslav Nakov and Marti Hearst (2005a). Search engine statistics beyond the n-gram: Application to noun compound bracketing. In *Proceedings of the Conference on Computational Natural Language Learning (CoNLL)*.

Preslav Nakov and Marti Hearst (2005b). Using the Web as an implicit training set: Application to structural ambiguity resolution. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Preslav Nakov and Marti Hearst (2008). Solving relational similarity problems using the Web as a corpus. In *Proceedings of the Annual Meeting on Association for Computational Linguistics (ACL)*.

Roberto Navigli, Paola Velardi, and Stefano Faralli (2011). A graph-based algorithm for inducing lexical taxonomies from scratch. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*.

Vincent Ng (2010). Supervised noun phrase coreference research: The first fifteen years. In *Proceedings of the Annual Meeting on Association for Computational Linguistics (ACL)*.

Vincent Ng and Claire Cardie (2002). Improving machine learning approaches to coreference resolution. In *Proceedings of the Annual Meeting on Association for Computational Linguistics (ACL)*.

NIST (2004). The ACE evaluation plan. In *US National Institute for Standards and Technology (NIST)*.

Eric W. Noreen (1989). Computer intensive methods for hypothesis testing: An introduction. In *Wiley, New York*.

Bo Pang and Lillian Lee (Jan. 2008). Opinion mining and sentiment analysis. In *Foundations and Trends in Information Retrieval* 2.1-2, pp. 1–135. ISSN: 1554-0669. DOI: 10.1561/1500000011. URL: http://dx.doi.org/10.1561/1500000011.

Patrick Pantel and Marco Pennacchiotti (2006). Espresso: Leveraging generic patterns for automatically harvesting semantic relations. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics (COLING-ACL).*

Marius Pasca (2004). Acquisition of categorized named entities for Web search. In *Proceedings of the ACM International Conference on Information and Knowledge Management (CIKM).*

M. Pennacchiotti and P. Pantel (2009). Entity extraction via ensemble semantics. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP).*

Marco Pennacchiotti and Patrick Pantel (2006). Ontologizing semantic relations. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics (COLING-ACL).*

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein (2006). Learning accurate, compact, and interpretable tree annotation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics (COLING-ACL).*

William Phillips and Ellen Riloff (2002). Exploiting strong syntactic heuristics and co-training to learn semantic lexicons. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP).*

Emily Pitler, Shane Bergsma, Dekang Lin, and Kenneth Church (2010). Using Web-scale n-grams to improve base NP parsing performance. In *Proceedings of the International Conference on Computational Linguistics (COLING).*

Massimo Poesio, Rahul Mehta, Axel Maroudas, and Janet Hitzeman (2004). Learning to resolve bridging references. In *Proceedings of the Annual Meeting on Association for Computational Linguistics (ACL).*

Simone Paolo Ponzetto and Michael Strube (2011). Taxonomy induction based on a collaboratively built knowledge repository. In *Artificial Intelligence* 175.9, pp. 1737–1756.

Hoifung Poon and Pedro Domingos (2010). Unsupervised ontology induction from text. In *Proceedings of the Annual Meeting on Association for Computational Linguistics (ACL).*

John R. Quinlan (1993). C4.5: Programs for machine learning. In *Morgan Kaufmann Publishers Inc., San Francisco, CA, USA*.

Altaf Rahman and Vincent Ng (2009). Supervised models for coreference resolution. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Altaf Rahman and Vincent Ng (2011). Coreference resolution with world knowledge. In *Proceedings of the Annual Meeting on Association for Computational Linguistics (ACL)*.

Adwait Ratnaparkhi (1996). A maximum entropy model for part-of-speech tagging. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Deepak Ravichandran, Patrick Pantel, and Eduard Hovy (2005). Randomized algorithms and NLP: Using locality sensitive hash functions for high speed noun clustering. In *Proceedings of the Annual Meeting on Association for Computational Linguistics (ACL)*.

Marta Recasens and Eduard Hovy (2010). Coreference resolution across corpora: Languages, coding schemes, and preprocessing information. In *Proceedings of the Annual Meeting on Association for Computational Linguistics (ACL)*.

Ellen Riloff and Jessica Shepherd (1997). A corpus-based approach for building semantic lexicons. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Alan Ritter, Stephen Soderland, and Oren Etzioni (2009). What is this, anyway: Automatic hypernym discovery. In *Proceedings of AAAI Spring Symposium on Learning by Reading and Learning to Read*.

Peter F. Schulam and Christiane Fellbaum (2010). Automatically determining the semantic gradation of German adjectives. In *Proceedings of KONVENS*.

Vera Sheinman and Takenobu Tokunaga (2009). AdjScales: Visualizing differences between adjectives for language learners. In *IEICE Transactions on Information and Systems* 92(8), 15421550.

Vera Sheinman, Takenobu Tokunaga, Isaac Julien, Peter Schulam, and Christiane Fellbaum (2012). Refining WordNet adjective dumbbells using intensity relations. In *Proceedings of Global WordNet Conference*.

David A. Smith and Jason Eisner (2008). Dependency parsing by belief propagation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

R. Snow, D. Jurafsky, and A.Y. Ng (2004). Learning syntactic patterns for automatic hypernym discovery. In *Proceedings of Neural Information Processing Systems (NIPS)*.

Rion Snow, Daniel Jurafsky, and Andrew Y. Ng (2006). Semantic taxonomy induction from heterogenous evidence. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics (COLING-ACL)*.

Wee Meng Soon, Hwee Tou Ng, and Daniel Chung Yong Lim (2001). A machine learning approach to coreference resolution of noun phrases. In *Computational Linguistics* 27(4), pp. 521–544.

Charles Spearman (1904). The proof and measurement of association between two things. In *The American Journal of Psychology* 15(1), pp. 72–101.

Veselin Stoyanov, Nathan Gilbert, Claire Cardie, and Ellen Riloff (2009). Conundrums in noun phrase coreference resolution: Making sense of the state-of-the-art. In *Proceedings of the Annual Meeting on Association for Computational Linguistics and the International Joint Conference on Natural Language Processing (ACL-IJCNLP)*.

Veselin Stoyanov, Claire Cardie, Nathan Gilbert, Ellen Riloff, David Buttler, and David Hysom (2010). Reconcile: A coreference resolution research platform. In *Technical report, Cornell University*.

Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum (2007). Yago: A core of semantic knowledge. In *Proceedings of the International Conference on World Wide Web (WWW)*.

Fabian M. Suchanek, Mauro Sozio, and Gerhard Weikum (2009). SOFIE: A self-organizing framework for information extraction. In *Proceedings of the International Conference on World Wide Web (WWW)*.

Maite Taboada, Julian Brooke, Milan Tofiloskiy, and Kimberly Vollz (2011). Lexicon-based methods for sentiment analysis. In *Computational Linguistics*.

Partha Pratim Talukdar, Joseph Reisinger, Marius Paşca, Deepak Ravichandran, Rahul Bhagat, and Fernando Pereira (2008). Weakly-supervised acquisition of labeled class instances using graph random walks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Niket Tandon and Gerard de Melo (2010). Information extraction from Web-scale n-gram data. In *Proceedings of the Web N-gram Workshop at the International ACM Conference on Research and Development in Information Retrieval (SIGIR)*.

Robert E. Tarjan (1977). Finding optimum branchings. In *Networks* 7, pp. 25–35.

Peter D. Turney (2008). A uniform approach to analogies, synonyms, antonyms, and associations. In *Proceedings of the International Conference on Computational Linguistics (COLING)*.

Peter D. Turney and Michael L. Littman (Oct. 2003). Measuring praise and criticism: inference of semantic orientation from association. In *ACM Trans. Inf. Syst.* 21.4, pp. 315–346. ISSN: 1046-8188. DOI: 10.1145/944012.944013. URL: http://doi.acm.org/10.1145/944012.944013.

William T. Tutte (1984). Graph theory. In *Addison-Wesley*.

David Vadas and James R. Curran (2007). Adding noun phrase structure to the Penn Treebank. In *Proceedings of the Annual Meeting on Association for Computational Linguistics (ACL)*.

Marc Vilain, John Burger, John Aberdeen, Dennis Connolly, and Lynette Hirschman (1995). A model-theoretic coreference scoring scheme. In *Proceedings of the Conference on Message Understanding (MUC)*.

Martin Volk (2001). Exploiting the WWW as a corpus to resolve PP attachment ambiguities. In *Proceedings of Corpus Linguistics*.

Dominic Widdows (2003). Unsupervised methods for developing taxonomies by combining syntactic and statistical information. In *Proceedings of the Human Language Technologies and the North American Chapter of the Association for Computational Linguistics (HLT-NAACL)*.

Ichiro Yamada, Kentaro Torisawa, Jun'ichi Kazama, Kow Kuroda, Masaki Murata, Stijn De Saeger, Francis Bond, and Asuka Sumida (2009). Hypernym discovery based on distributional similarity and hierarchical structures. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Hui Yang and Jamie Callan (2009). A metric-based framework for automatic taxonomy induction. In *Proceedings of the Annual Meeting on Association for Computational Linguistics and the International Joint Conference on Natural Language Processing (ACL-IJCNLP)*.

Xiaofeng Yang and Jian Su (2007). Coreference resolution using semantic relatedness information from automatically discovered patterns. In *Proceedings of the Annual Meeting on Association for Computational Linguistics (ACL)*.

Xiaofeng Yang, Jian Su, and Chew Lim Tan (2005). Improving pronoun resolution using statistics-based semantic compatibility information. In *Proceedings of the Annual Meeting on Association for Computational Linguistics (ACL)*.

Alexander Yates, Stefan Schoenmackers, and Oren Etzioni (2006). Detecting parser errors using Web-based semantic filters. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Ainur Yessenalina and Claire Cardie (2011). Compositional matrix-space models for sentiment analysis. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.