Using Web-scale N-grams to Improve Base NP Parsing Performance

Emily Pitler
Computer and Information Science
University of Pennsylvania
epitler@seas.upenn.edu

Shane Bergsma
Department of Computing Science
University of Alberta
sbergsma@ualberta.ca

Dekang Lin
Google, Inc.
lindek@google.com

Kenneth Church
Human Language Technology Center of Excellence
Johns Hopkins University
kenneth.church@jhu.edu

Abstract

We use web-scale N-grams in a base NP parser that correctly analyzes 95.4% of the base NPs in natural text. Web-scale data improves performance. That is, there is no data like more data. Performance scales log-linearly with the number of parameters in the model (the number of unique N-grams). The web-scale N-grams are particularly helpful in harder cases, such as NPs that contain conjunctions.

1 Introduction

Noun phrases (NPs) provide an index to the world's information. About 70% of web queries are NPs (Barr et al., 2008). A robust NP parser could help search engines improve retrieval performance on multi-word NP queries (Zhai, 1997). For example, by knowing the correct parse of "washed (baby carrots)," a search engine could ensure that returned pages (and advertisements) concern clean carrots rather than clean babies. NP structure is also helpful for query expansion and substitution (Jones et al., 2006).

This paper is concerned with base NP parsing. We are given a base NP string as input, and the task is to produce a parse tree as output. Base NPs are NPs that do not contain embedded noun phrases. These are sometimes called NP chunks, or core/non-recursive NPs (Church, 1988; Ramshaw and Marcus, 1995). Correctly parsing (or, equivalently, bracketing) base NPs is challenging because the same part-of-speech (POS) sequence can be parsed differently depending on the specific words involved. For example, "*retired (science teacher)*" and "*(social science) teacher*" have different structures even though they have identical POS sequences.

Lexical statistics are therefore needed in order to parse the above examples, and they must be computed over a lot of text to avoid sparsity. All of our lexical statistics are derived from a new and improved web-scale N-gram corpus (Lin et al., 2010), which we call Google V2.

Despite the importance of base NPs, most sentence parsers do not parse base NPs, since the main training corpus for parsers, the Penn Treebank (PTB) (Marcus et al., 1994), leaves a flat structure for base NPs. Recent annotations by Vadas and Curran (2007a) added NP structure to the PTB. We use these annotations (described in Section 3) for our experiments.

NP parsers usually focus on bracketing three-word noun compounds. Parsing three-word noun compounds is a fairly artificial task; we show that sequences of three nouns make up less than 1% of the three-word-or-longer base NPs in natural text. As the NP length increases, the number of possible binary trees (parses) increases with the Catalan numbers (Church and Patil, 1982). NPs of length three have just two possible parses (chance is 50%), while NPs of length six already have forty-two possible parses (chance is 2%). Long NPs therefore provide much more opportunity to improve performance over the baseline. In Table 1 (Section 7), we show the distribution of base NP length in the PTB. While most NPs are of length three, NP length has a long tail.

The three-word noun compound assumption also restricts research to the case in which all words are nouns, while base NPs also contain determiners, possessives, adjectives, and conjunctions. Conjunctions and their scopes are particularly challenging. For example, in the NP, "*French television and movie producers*," a parser should conjoin "(*television*) *and* (*movie*)," as opposed to "(*French television*) *and* (*movie*)," "(*French television*) *and* (*movie producers*)" or "(*television*) *and* (*movie producers*)."

To resolve these issues, we train a classifier which uses contextual information from the entire NP and lexical statistics derived from the web-scale N-gram corpus to predict if a given span is a constituent. Our parser then uses this classifier to produce a score for every possible NP-internal bracketing and creates a chart of bracketing scores. This chart can be used as features in a full sentence parser or parsed directly with a chart parser. Our parses are highly accurate, creating a strong new standard for this task.

Finally, we present experiments that investigate the effects of N-gram frequency cutoffs and various sources of N-gram data. We show an interesting relationship between accuracy and the number of unique N-gram types in the data.

2 Related Work

2.1 Three-Word Noun Compounds

The most commonly used data for NP parsing is from Lauer (1995), who extracted 244 three-word noun compounds from the Grolier encyclopedia. When there are only three words, this task reduces to a binary decision:

- Left Branching: * [*retired science*] *teacher*
- Right Branching: *retired* [*science teacher*]

In Lauer (1995)'s set of noun compounds, two-thirds are left branching.

The main approach to these three-word noun compounds has been to compute association statistics between pairs of words and then choose the bracketing that corresponds to the more highly associated pair. The two main models are the *adjacency model* (Marcus, 1980; Liberman and Sproat, 1992; Pustejovsky et al., 1993; Resnik, 1993) and the *dependency model* (Lauer, 1995). Under the adjacency model, the bracketing decision is made by comparing the associations between words one and two versus words two and three (i.e. comparing *retired science* versus *science teacher*). In contrast, the dependency model compares the associations between one and two versus one and three (*retired science* versus *retired teacher*). Lauer (1995) compares the two models and finds the dependency model to be more accurate.

Nakov and Hearst (2005) compute the association scores using frequencies, conditional probabilities, χ^2, and mutual information, for both pairs of words and for linguistically-motivated paraphrases. Lapata and Keller (2005) found that using web-scale data for associations is better than using the (smaller) 100M-word British National Corpus.

2.2 Longer NPs

Focusing on only the three word case misses a large opportunity for base NP parsing. NPs longer than three words commonly occur, making up 29% of our test set. In addition, a chance baseline does exponentially worse as the length of the NP increases. These longer NPs are therefore a major opportunity to improve overall base NP parsing.

Since in the general case, NP parsing can no longer be thought of as a single binary classification problem, different strategies are required.

Barker (1998) reduces the task of parsing longer NPs to making sequential three-word decisions, moving a sliding window along the NP. The window is first moved from right-to-left, inserting right bracketings, and then again from left-to-right, finalizing left bracketings. While Barker (1998) assumes that these three-word decisions can be made in isolation, this is not always valid.[1] Vadas and Curran (2007b) employ Barker's algorithm, but use a supervised classifier to make the sequential bracketing decisions. Because these approaches rely on a sequence of binary decisions,

[1]E.g., although the right-most three words are identical in 1) "*soap opera stars and television producers*," and 2) "*movie and television producers*," the initial right-bracketing decision for "*and television producers*" should be different in each.

early mistakes can cascade and lead to a chain of incorrect bracketings.

Our approach differs from previous work in NP parsing; rather than greedily inserting brackets as in Barker's algorithm, we use dynamic programming to find the global maximum-scoring parse. In addition, unlike previous approaches that have used local features to make local decisions, we use the full NP to score each potential bracketing.

A related line of research aims to *segment* longer phrases that are queried on Internet search engines (Bergsma and Wang, 2007; Guo et al., 2008; Tan and Peng, 2008). Bergsma and Wang (2007) focus on NP queries of length four or greater. They use supervised learning to make segmentation decisions, with features derived from the noun compound bracketing literature. Evaluating the benefits of *parsing* NP queries, rather than simply segmenting them, is a natural application of our system.

3 Annotated Data

Our training and testing data are derived from recent annotations by Vadas and Curran (2007a). The original PTB left a flat structure for base noun phrases. For example, "*retired science teacher*," would be represented as:

(NP (JJ retired) (NN science) (NN teacher))

Vadas and Curran (2007a) annotated NP-internal structure by adding annotations whenever there is a left-bracketing. If no annotations were added, right-branching is assumed. The inter-annotator agreement for exactly matching the brackets on an NP was 98.5%.

This data provides a valuable new resource for parsing research, but little work has so far made use of it. Vadas and Curran (2007b) perform some preliminary experiments on NP bracketing, but use gold standard part-of-speech and named-entity annotations as features in their classifier. Our work establishes a strong and realistic standard on this data; our results will serve as a basis for further research on this topic.

4 Unlabeled N-gram Data

All of our N-gram features described in Section 6.1 rely on probabilities derived from unlabeled data. To use the largest amount of data possible, we exploit web-scale N-gram corpora. N-gram counts are an efficient way to compress large amounts of data (such as all the text on the web) into a manageable size. An N-gram corpus records how often each unique sequence of words occurs. Co-occurrence probabilities can be calculated directly from the N-gram counts. To keep the size manageable, N-grams that occur with a frequency below a particular threshold can be filtered.

The corpus we use is **Google V2** (Lin et al., 2010): a new N-gram corpus with N-grams of length 1-5 that we created from the same 1 trillion word snapshot of the web as Google N-grams Version 1 (Brants and Franz, 2006), but with several enhancements. Duplicate sentences are removed, as well as "sentences" which are probably noise (indicated by having a large proportion of non-alphanumeric characters, being very long, or being very short). Removing duplicate sentences is especially important because automatically-generated websites, boilerplate text, and legal disclaimers skew the source web data, with sentences that may have only been authored once occurring millions of times. We use the suffix array tools described in Lin et al. (2010) to quickly extract N-gram counts.

5 Base NP Parsing Approach

Our goal is to take a base NP string as input and produce a parse tree as output. In practice, it would be most useful if the NP parse could be integrated into a sentence parser. Previous NP parsers are difficult to apply in practice.[2] Work in prepositional phrase attachment that assumes gold-standard knowledge of the competing attachment sites has been criticized as unrealistic (Atterer and Schütze, 2007).

Our system can easily be integrated into full parsers. Its input can be identified quickly and reliably and its output is compatible with downstream parsers.

[2]For example, Vadas and Curran (2007b) report results on NP parsing, but these results include NPs containing prepositional or adverbial phrases (confirmed by personal communication). Practical application of their system would therefore require resolving prepositional phrase attachment as a preprocessing step.

Our parser's input is base NPs, which can be identified with very high accuracy. Kudo and Matsumoto (2001) report 95.8% NP chunking accuracy on PTB data.

Once provided with an NP, our system uses a supervised classifier to predict the probability of a particular contiguous subsequence (span) of the NP being a constituent, given the entire NP as context. This probability can be inserted into the chart that a standard chart parser would use.

For example, the base NP "*French television and movie producers*" would be decomposed into nine different classification problems, scoring the following potential bracketings:

(French television) and movie producers
French (television and) movie producers
(French television and) movie producers ...
French television and (movie producers)

In Section 6, we detail the set of statistical and structural features used by the classifier.

The output of our classifier can be easily used as a feature in a full-sentence structured prediction parser, as in Taskar et al. (2004). Alternatively, our work could be integrated into a full-sentence parser by using our feature representations directly in a discriminative CFG parser (Finkel et al., 2008), or in a parse re-ranker (Ratnaparkhi et al., 1994; Collins and Koo, 2005; Charniak and Johnson, 2005).

While our main objective is to use web-scale lexical statistics to create an accurate classifier for base NP-internal constituents, we do produce a parse tree for evaluation purposes. The probability of a parse tree is defined as the product of the probabilities of all the spans (constituents) in the tree. The most probable tree is computed with the CYK algorithm.

6 Features

Over the course of development experiments, we discovered that the more position-specific our features were, the more effectively we could parse NPs. We define a word's position as its distance from the right of the full NP, as the semantic head of NPs is most often the right-most word. Ultimately, we decided to conjoin each feature with the position of the proposed bracketing. Since the features for differing proposed bracketings are now disjoint, this is equivalent to scoring bracketings with different classifiers, with each classifier chosen according to the bracketing position. We now outline the feature types that are common, but weighted differently, in each proposed bracketing's feature set.

6.1 N-gram Features

All of the features described in this section require estimates of the probability of specific words or sequences of words. All probabilities are computed using **Google V2** (Section 4).

6.1.1 PMI

Recall that the adjacency model for the three-word task uses the associations of the two pairs of adjacent words, while the dependency model uses the associations of the two pairs of attachment sites for the initial noun. We generalize the adjacency and dependency models by including the pointwise mutual information (Church and Hanks, 1990) between *all* pairs of words in the NP:

$$\mathrm{PMI}(x, y) = \log \frac{p(\text{“x y”})}{p(\text{“x”})p(\text{“y”})} \tag{1}$$

For NPs of length n, for each proposed bracketing, we include separate features for the PMI between all $\binom{n}{2}$ pairs of words in the NP. For NPs including conjunctions, we include additional PMI features (Section 6.1.2).

Since these features are also tied to the proposed bracketing positions (as explained above), this allows us to learn relationships between various associations within the NP and each potential bracketing. For example, consider a proposed bracketing from word 4 to word 5. We learn that a high association of words inside a bracketing (here, a high association between word 4 and word 5) indicates a bracketing is likely, while a high association between words that cross a proposed bracketing (e.g., a high association between word 3 and word 4) indicates the bracketing is unlikely.

The value of these features is the PMI, if it is defined. If the PMI is undefined, we include one of two binary features:
$p(\text{“x y”}) = 0$ or $p(\text{“x”}) \vee p(\text{“y”}) = 0$.

We illustrate the PMI features with an example. In deciding whether (*movie producers*) is a reasonable bracketing within "*French television and movie producers*," the classifier weighs features for all of:

PMI(*French, television*)
PMI(*French, and*)
...
PMI(*television, producers*)
PMI(*and, producers*)
PMI(*movie, producers*)

6.1.2 Conjunctions

Properly handling NPs containing conjunctions (NP+conj) requires special statistical features. For example, *television* and *movie* are commonly conjoined, but the relevant statistics that suggest placing brackets around the phrase "*television and movie*" are not provided by the above PMI features (i.e., this is not clear from PMI(*television, and*), PMI(*television, movie*), nor PMI(*and, movie*)). Rather, we want to know if the full phrase "television and movie" is common.

We thus have additional NP+conj features that consider the PMI association across the word *and*:

$$\text{PMI}_{and}(x, y) = \log \frac{p(\text{"}x \text{ and } y\text{"})}{p(\text{"}x \text{ and"})p(\text{"and } y\text{"})} \quad (2)$$

When PMI_{and} between a pair of words is high, they are likely to be the constituents of a conjunction.

Let $NP{=}(w_1 \ldots w_{i-1}, \text{'}and\text{'}, w_{i+1} \ldots w_n)$ be an NP+conj. We include the PMI_{and} features between w_{i-1} and all $w \in w_{i+1} \ldots w_n$. In the example "*French television and movie producers*," we would include features PMI_{and}(*television, movie*) and PMI_{and}(*television, producers*).

In essence, we are assuming w_{i-1} is the head of one of the items being conjoined, and we score the likelihood of each of the words to the right of the *and* being the head for the other item. In our running example, the conjunction has narrow scope, and PMI_{and}(*television, movie*) is greater than PMI_{and}(*television, producers*), indicating to our classifier that (*television and movie*) is a good bracketing. In other examples the conjunction will join heads that are further apart, as in *((French TV) and (British radio)) stars,* where both of the following hold:

$\text{PMI}_{and}(\text{TV, radio}) > \text{PMI}_{and}(\text{TV, British})$
$\text{PMI}_{and}(\text{TV, radio}) > \text{PMI}_{and}(\text{TV, stars})$

6.2 Lexical

We include a binary feature to indicate the presence of a particular word at each position in the NP. We learn that, for instance, the word *Inc.* in names tends to occur outside of brackets.

6.3 Shape

Previous work on NP bracketing has used gold-standard named entity tags (Vadas and Curran, 2007b) as features. We did not want to use any gold-standard features in our experiments, however NER information is helpful in separating premodifiers from names, i.e. *(news reporter) (Walter Cronkite).*

As an expedient way to get both NER information and useful information from hyphenated adjectives, abbreviations, and other punctuation, we normalize each string using the following regular expressions:

[A-Z]+ $\rightarrow$ A [a-z]+ $\rightarrow$ a

We use this normalized string as an indicator feature. E.g. the word "Saudi-born" will fire the binary feature "Aa-a."

6.4 Position

We also include the position of the proposed bracketing as a feature. This represents the prior of a particular bracketing, regardless of the actual words.

7 Experiments

7.1 Experimental Details

We use Vadas and Curran (2007a)'s annotations (Section 3) to create training, development and testing data for base NPs, using standard splits of the Penn Treebank (Table 1). We consider all non-trivial base NPs, i.e., those longer than two words.

For training, we expand each NP in our training set into independent examples corresponding to all the possible internal NP-bracketings, and represent these examples as feature vectors (Section 5). Each example is positively labeled if it is

Data Set	Train	Dev	Test	Chance
PTB Section	2-22	24	23	
Length=3	41353	1428	2498	50%
Length=4	12067	445	673	20%
Length=5	3930	148	236	7%
Length=6	1272	34	81	2%
Length>6	616	29	34	< 1%
Total NPs	59238	2084	3522	

Table 1: Breakdown of the PTB base NPs used in our experiments. Chance = 1/Catalan(length).

Features	All NPs	NP+conj	NP-conj
All features	**95.4**	**89.7**	**95.7**
-N-grams	94.0	84.0	94.5
-lexical	92.2	87.4	92.5
-shape	94.9	**89.7**	95.2
-position	95.3	**89.7**	95.6
Right bracketing	72.6	58.3	73.5

Table 2: Accuracy (%) of base NPs parsing; ablation of different feature classes.

consistent with the gold-standard bracketing, otherwise it is a negative example.

We train using LIBLINEAR, an efficient linear Support Vector Machine (SVM).[3] We use an L2-loss function, and optimize the regularization parameter on the development set (reaching an optimum at C=1). We converted the SVM output to probabilities.[4] Perhaps surprisingly, since SVMs are not probabilistic, performance on the development set with these SVM-derived probabilities was higher than using probabilities from the LIBLINEAR logistic regression solver.

At test time, we again expand the NPs and calculate the probability of each constituent, inserting the score into a chart. We run the CYK algorithm to find the most probable parse of the entire NP according to the chart. Our evaluation metric is **Accuracy**: the proportion of times our proposed parse of the NP exactly matches the gold standard.

8 Results

8.1 Base NPs

Our method improves substantially over the baseline of assuming a completely right-branching structure, 95.4% versus 72.6% (Table 2). The accuracy of the constituency classifier itself (before the CYK parser is used) is 96.1%.

The lexical features are most important, but all feature classes are somewhat helpful. In particular, including N-gram PMI features significantly improves the accuracy, from 94.0% to 95.4%.[5] Correctly parsing more than 19 base NPs out of 20 is an exceptional level of accuracy, and provides a strong new standard on this task. The most comparable result is by Vadas and Curran (2007b), who achieved 93.0% accuracy on a different set of PTB noun phrases (see footnote 2), but their classifier used features based on gold-standard part-of-speech and named-entity information.

Exact match is a tough metric for parsing, and the difficulty increases as the length of the NP increases (because there are more decisions to make correctly). At three word NPs, our accuracy is 98.5%; by six word NPs, our accuracy drops to 79.0% (Figure 1). Our method's accuracy decreases as the length of the NP increases, but much less rapidly than a right-bracketing or chance baseline.

8.2 Base NPs with Conjunctions

N-gram PMI features help more on NP+conj than on those that do not contain conjunctions (NP-conj) (Table 2). N-gram PMI features are the most important features for NP+conj, increasing accuracy from 84.0% to 89.7%, a 36% relative reduction in error.

8.3 Effect of Thresholding N-gram data

We now address two important related questions: 1) how does our parser perform as the amount of unlabeled auxiliary data varies, and 2) what is the effect of thresholding an N-gram corpus? The second question is of widespread relevance as

[3]`www.csie.ntu.edu.tw/~cjlin/liblinear/`

[4]Following instructions in `http://www.csie.ntu.edu.tw/~cjlin/liblinear/FAQ.html`

[5]McNemar's test, $p < 0.05$

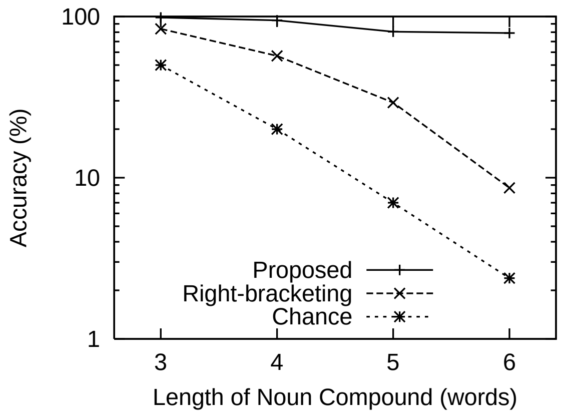

Figure 1: Accuracy (log scale) over different NP lengths, of our method, the right-bracketing baseline, and chance (1/Catalan(length)).

thresholded N-gram corpora are now widely used in NLP. Without thresholds, web-scale N-gram data can be unmanageable.

While we cannot lower the threshold after creating the N-gram corpus, we can raise it, filtering more N-grams, and then measure the relationship between threshold and performance.

Threshold	Unique N-grams	Accuracy
10	4,145,972,000	95.4%
100	391,344,991	95.3%
1,000	39,368,488	95.2%
10,000	3,924,478	94.8%
100,000	386,639	94.8%
1,000,000	37,567	94.4%
10,000,000	3,317	94.0%

Table 3: There is no data like more data. Accuracy improves with the number of parameters (unique N-grams).

We repeat the parsing experiments while including in our PMI features only N-grams with a count $\geq$10 (the whole data set), $\geq$100, $\geq$1000, ..., $\geq 10^7$. All other features (lexical, shape, position) remain unchanged. The N-gram data almost perfectly exhibits Zipf's power law: raising the threshold by a factor of ten decreases the number of unique N-grams by a factor of ten (Table 3). The improvement in accuracy scales log-linearly with the number of unique N-grams. From a practical standpoint, we see a trade-off between storage and accuracy. There are consistent improvements in accuracy from lowering the threshold and increasing the amount of auxiliary data. If for some application it is necessary to reduce storage by several orders of magnitude, then one can easily estimate the resulting impact on performance.

Corpus	# of tokens	τ	# of types
NEWS	3.2 B	1	3.7 B
Google V1	1,024.9 B	40	3.4 B
Google V2	207.4 B	10	4.1 B

Table 4: N-gram data, with total number of words (*tokens*) in the original corpus (in billions, B), frequency threshold used to filter the data, τ, and total number of unique N-grams (*types*) remaining in the data after thresholding.

We repeat the thresholding experiments using two other N-gram sources:

NEWS: N-gram data created from a large set of news articles including the Reuters and Gigaword (Graff, 2003) corpora, not thresholded.

Google V1: The original web-scale N-gram corpus (Section 4).

Details of these sources are given in Table 4.

For a given number of unique N-grams, using any of the three sources does about the same (Figure 2). It does not matter that the source corpus for Google V1 is about five times larger than the source corpus for Google V2, which in turn is sixty-five times larger than NEWS (Table 4). Accuracies increase linearly with the log of the number of *types* in the auxiliary data set.

Google V1 is the one data source for which the relationship between accuracy and number of N-grams is not monotonic. After about 100 million unique N-grams, performance starts decreasing. This drop shows the need for Google V2. Since Google V1 contains duplicated web pages and sentences, mistakes that should be rare can appear to be quite frequent. Google V2, which comes from the same snapshot of the web as Google V1, but has only unique sentences, does not show this drop.

We regard the results in Figure 2 as a companion to Banko and Brill (2001)'s work on exponentially increasing the amount of labeled training data. Here we see that varying the amount of

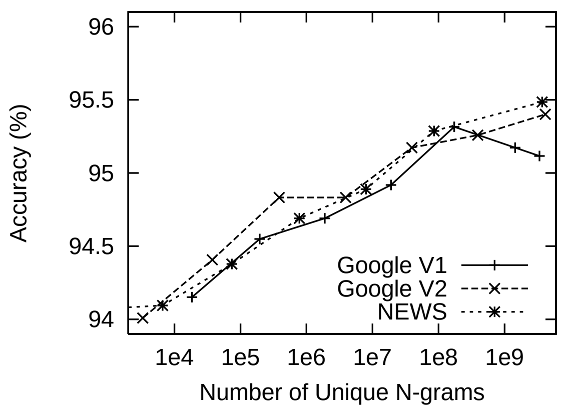

Figure 2: There is no data like more data. Accuracy improves with the number of parameters (unique N-grams). This trend holds across three different sources of N-grams.

unlabeled data can cause an equally predictable improvement in classification performance, without the cost of labeling data.

Suzuki and Isozaki (2008) also found a log-linear relationship between unlabeled data (up to a billion words) and performance on three NLP tasks. We have shown that this trend continues well beyond Gigaword-sized corpora. Brants et al. (2007) also found that more unlabeled data (in the form of input to a language model) leads to improvements in BLEU scores for machine translation.

Adding noun phrase parsing to the list of problems for which there is a “bigger is better” relationship between performance and unlabeled data shows the wide applicability of this principle. As both the amount of text on the web and the power of computer architecture continue to grow exponentially, collecting and exploiting web-scale auxiliary data in the form of N-gram corpora should allow us to achieve gains in performance linear in time, without any human annotation, research, or engineering effort.

9 Conclusion

We used web-scale N-grams to produce a new standard in performance of base NP parsing: 95.4%. The web-scale N-grams substantially improve performance, particularly in long NPs that include conjunctions. There is no data like more data. Performance improves log-linearly with the number of parameters (unique N-grams). One can increase performance with larger models, e.g., increasing the size of the unlabeled corpora, or by decreasing the frequency threshold. Alternatively, one can decrease storage costs with smaller models, e.g., decreasing the size of the unlabeled corpora, or by increasing the frequency threshold. Either way, the log-linear relationship between accuracy and model size makes it easy to estimate the trade-off between performance and storage costs.

Acknowledgments

We gratefully acknowledge the Center for Language and Speech Processing at Johns Hopkins University for hosting the workshop at which this research was conducted.

References

Atterer, M. and H. Schütze. 2007. Prepositional phrase attachment without oracles. *Computational Linguistics*, 33(4):469–476.

Banko, M. and E. Brill. 2001. Scaling to very very large corpora for natural language disambiguation. In *ACL*.

Barker, K. 1998. A trainable bracketer for noun modifiers. In *Twelfth Canadian Conference on Artificial Intelligence (LNAI 1418)*.

Barr, C., R. Jones, and M. Regelson. 2008. The linguistic structure of English web-search queries. In *EMNLP*.

Bergsma, S. and Q.I. Wang. 2007. Learning noun phrase query segmentation. In *EMNLP-CoNLL*.

Brants, T. and A. Franz. 2006. The Google Web 1T 5-gram Corpus Version 1.1. LDC2006T13.

Brants, T., A.C. Popat, P. Xu, F.J. Och, and J. Dean. 2007. Large language models in machine translation. In *EMNLP*.

Charniak, E. and M. Johnson. 2005. Coarse-to-fine n-best parsing and MaxEnt discriminative reranking. In *ACL*.

Church, K.W. and P. Hanks. 1990. Word association norms, mutual information, and lexicography. *Computational Linguistics*, 16(1):22–29.

Church, K. and R. Patil. 1982. Coping with syntactic ambiguity or how to put the block in the box on the table. *Computational Linguistics*, 8(3-4):139–149.

Church, K.W. 1988. A stochastic parts program and noun phrase parser for unrestricted text. In *ANLP*.

Collins, M. and T. Koo. 2005. Discriminative reranking for natural language parsing. *Computational Linguistics*, 31(1):25–70.

Finkel, J.R., A. Kleeman, and C.D. Manning. 2008. Efficient, feature-based, conditional random field parsing. In *ACL*.

Graff, D. 2003. English Gigaword. LDC2003T05.

Guo, J., G. Xu, H. Li, and X. Cheng. 2008. A unified and discriminative model for query refinement. In *SIGIR*.

Jones, R., B. Rey, O. Madani, and W. Greiner. 2006. Generating query substitutions. In *WWW*.

Kudo, T. and Y. Matsumoto. 2001. Chunking with support vector machines. In *NAACL*.

Lapata, M. and F. Keller. 2005. Web-based models for natural language processing. *ACM Transactions on Speech and Language Processing*, 2(1):1–31.

Lauer, M. 1995. Corpus statistics meet the noun compound: some empirical results. In *ACL*.

Liberman, M. and R. Sproat. 1992. The stress and structure of modified noun phrases in English. *Lexical matters*, pages 131–181.

Lin, D., K. Church, H. Ji, S. Sekine, D. Yarowsky, S. Bergsma, K. Patil, E. Pitler, R. Lathbury, V. Rao, K. Dalwani, and S. Narsale. 2010. New tools for web-scale n-grams. In *LREC*.

Marcus, M.P., B. Santorini, and M.A. Marcinkiewicz. 1994. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Marcus, M.P. 1980. *Theory of Syntactic Recognition for Natural Languages*. MIT Press, Cambridge, MA, USA.

Nakov, P. and M. Hearst. 2005. Search engine statistics beyond the n-gram: Application to noun compound bracketing. In *CoNLL*.

Pustejovsky, J., P. Anick, and S. Bergler. 1993. Lexical semantic techniques for corpus analysis. *Computational Linguistics*, 19(2):331–358.

Ramshaw, L.A. and M.P. Marcus. 1995. Text chunking using transformation-based learning. In *3rd ACL Workshop on Very Large Corpora*.

Ratnaparkhi, A., S. Roukos, and R.T. Ward. 1994. A maximum entropy model for parsing. In *Third International Conference on Spoken Language Processing*.

Resnik, P. 1993. *Selection and information: a class-based approach to lexical relationships*. Ph.D. thesis, University of Pennsylvania.

Suzuki, J. and H. Isozaki. 2008. Semi-supervised sequential labeling and segmentation using giga-word scale unlabeled data. In *ACL*.

Tan, B. and F. Peng. 2008. Unsupervised query segmentation using generative language models and Wikipedia. In *WWW*.

Taskar, B., D. Klein, M. Collins, D. Koller, and C. Manning. 2004. Max-margin parsing. In *EMNLP*.

Vadas, D. and J.R. Curran. 2007a. Adding noun phrase structure to the Penn Treebank. In *ACL*.

Vadas, D. and J.R. Curran. 2007b. Large-scale supervised models for noun phrase bracketing. In *PACLING*.

Zhai, C. 1997. Fast statistical parsing of noun phrases for document indexing. In *ANLP*.