Learning Noun Phrase Query Segmentation

Shane Bergsma and Qin Iris Wang
Department of Computing Science
University of Alberta
Edmonton, Alberta, Canada, T6G 2E8
{bergsma,wqin}@cs.ualberta.ca

Abstract

Query segmentation is the process of taking a user's search-engine query and dividing the tokens into individual phrases or semantic units. Identification of these query segments can potentially improve both document-retrieval precision, by first returning pages which contain the exact query segments, and document-retrieval recall, by allowing query expansion or substitution via the segmented units. We train and evaluate a machine-learned query segmentation system that achieves 86% segmentation-decision accuracy on a gold standard set of segmented noun phrase queries, well above recently published approaches. Key enablers of this high performance are features derived from previous natural language processing work in noun compound bracketing. For example, token association features beyond simple N-gram counts provide powerful indicators of segmentation.

1 Introduction

Billions of times every day, people around the world communicate with Internet search engines via a small text box on a web page. The user provides a sequence of words to the search engine, and the search engine interprets the query and tries to return web pages that not only *contain* the query tokens, but that are also somehow *about* the topic or idea that the query terms describe.

Recent years have seen a widespread recognition that the user is indeed providing natural language text to the search engine; query tokens are not independent, unordered symbols to be matched on a web document but rather ordered words and phrases with syntactic relationships. For example, Zhai (1997) pointed out that indexing on single-word symbols is not able to distinguish a search for "bank terminology" from one for "terminology bank." The reader can submit these queries to a current search engine to confirm that modern indexing does recognize the effect of token order on query meaning in some way.

Accurately interpreting query semantics also depends on establishing relationships between the query tokens. For example, consider the query "two man power saw." There are a number of possible interpretations of this query, and these can be expressed through a number of different segmentations or bracketings of the query terms:

1. [two man power saw]
2. [two man] [power saw]
3. [two] [man] [power saw]
4. [two] [man power] [saw], etc.

One simple way to make use of these interpretations in search would be to put quotation marks around the phrasal segments to require the search engine to only find pages with exact phrase matches. If, as seems likely, the searcher is seeking pages about the large, mechanically-powered two-man saws used by lumberjacks and sawyers to cut big trees, then the first segmentation is correct. Indeed, a phrasal search for "two man power saw" on Google does find the device of interest. So does the second interpretation, but along with other, less-relevant pages discussing competitions involving "*two-man* handsaw,

Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pp. 819–826, Prague, June 2007.

two-woman handsaw, *power saw* log bucking, etc." The top document returned for the third interpretation, meanwhile, describes a *man* on a rampage at a subway station with *two* cordless *power saws*, while the fourth interpretation finds pages about topics ranging from hockey's thrilling *two-man power* play advantage to the *man power* situation during the Second World War. Clearly, choosing the right segmentation means finding the right documents faster.

Query segmentation can also help if insufficient pages are returned for the original query. A technique such as query substitution or expansion (Jones et al., 2006) can be employed using the segmented units. For example, we could replace the sexist "two man" modifier with the politically-correct "two person" phrase in order to find additional relevant documents. Without segmentation, expanding via the individual words "two," "man," "power," or "saw" could produce less sensible results.

In this paper, we propose a data-driven, machine-learned approach to query segmentation. Similar to previous segmentation approaches described in Section 2, we make a decision to segment or not to segment between each pair of tokens in the query. Unlike previous work, we view this as a classification task where the decision parameters are learned discriminatively from gold standard data. In Section 3, we describe our approach and the features we use. Section 4 describes our labelled data, as well as the specific tools used for our experiments. Section 5 provides the results of our evaluation, and shows the strong gains in performance possible using a wide set of features within a discriminative framework.

2 Related Work

Query segmentation has previously been approached in an unsupervised manner. Risvik et al. (2003) combine the frequency count of a segment and the mutual information (MI) between pairs of words in the segment in a heuristic scoring function. The system chooses the segmentation with the highest score as the output segmentation. Jones et al. (2006) use MI between pairs of tokens as the sole factor in deciding on segmentation breaks. If the MI is above a threshold (optimized on a small training set), the pair of tokens is joined in a segment. Otherwise, a segmentation break is made.

Query segmentation is related to the task of noun compound (NC) bracketing. NC bracketing determines the syntactic structure of an NC as expressed by a binary tree, or, equivalently, a binary bracketing (Nakov and Hearst, 2005a). Zhai (1997) first identified the importance of syntactic query/corpus parsing for information retrieval, but did not consider query segmentation itself. In principle, as N increases, the number of binary trees for an N-token compound is much greater than the 2^{N-1} possible segmentations. In practice, empirical NC research has focused on three-word compounds. The computational problem is thus deciding whether the three-word NC has a left or right-bracketing structure (Lauer, 1995). For the segmentation task, analysing a three-word NC requires deciding between four different segmentations. For example, there are two bracketings for "used car parts," the left-bracketing "[[used car] parts]" and the right-bracketing "[used [car parts]]," while there are four segmentations, including the case where there is only one segment, "[used car parts]" and the base case where each token forms its own segment, "[used] [car] [parts]." Query segmentation thus naturally handles the case where the query consists of multiple, separate noun phrases that should not be analysed with a single binary tree.

Despite the differences between the tasks, it is worth investigating whether the information that helps disambiguate left and right-bracketings can also be useful for segmentation. In particular, we explored many of the sources of information used by Nakov and Hearst (2005a), as well as several novel features that aid segmentation performance and should also prove useful for NC analysis researchers. Unlike all previous approaches that we are aware of, we apply our features in a flexible discriminative framework rather than a classification based on a vote or average of features.

NC analysis has benefited from the recent trend of using web-derived features rather than corpus-based counts (Keller and Lapata, 2003). Lapata and Keller (2004) first used web-based co-occurrence counts for the bracketing of NCs. Recent innovations have been to use statistics "beyond the N-gram," such as counting the number of web pages where a pair of words w, x participate in a genitive relationship ("w's x"), occur collapsed as a single phrase ("wx") (Nakov and Hearst, 2005a) or have a definite article as a left-boundary marker ("the w x") (Nicholson and Baldwin, 2006). We show strong performance gains when such features are employed for query segmentation.

NC bracketing is part of a larger field of research on multiword expressions including general NC interpretation. NC interpretation explores not just the syntactic dependencies among compound constituents, but the semantics of the nominal relationships (Girju et al., 2005). Web-based statistics have also had an impact on these wider analysis tasks, including work on interpretation of verb nominalisations (Nicholson and Baldwin, 2006) and NC coordination (Nakov and Hearst, 2005b).

3 Methodology

3.1 Segmentation Classification

Consider a query $\boldsymbol{x} = \{x_1, x_2, ..., x_N\}$ consisting of N query tokens. Segmentation is a mapping $S : \boldsymbol{x} \to \boldsymbol{y} \in Y_N$, where $\boldsymbol{y}$ is a segmentation from the set Y_N. Since we can either have or not have a segmentation break at each of the $N-1$ spaces between the N tokens, $|Y_N| = 2^{N-1}$. Supervised machine learning can be applied to derive the mapping S automatically, given a set of training examples consisting of pairs of queries and their segmentations $T = \{(\boldsymbol{x_i}, \boldsymbol{y_i})\}$. Typically this would be done via a set of features $\Psi(\boldsymbol{x}, \boldsymbol{y})$ for the structured examples. A set of weights $\boldsymbol{w}$ can be learned discriminatively such that each training example $(\boldsymbol{x_i}, \boldsymbol{y_i})$ has a higher score, $Score_{\boldsymbol{w}}(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{w} \cdot \Psi(\boldsymbol{x}, \boldsymbol{y})$, than alternative query-segmentation pairs, $(\boldsymbol{x_i}, \boldsymbol{z_i}), z_i \neq y_i$.[1] At test time, the classifier chooses the segmentation for $\boldsymbol{x}$ that has the highest score according to the learned parameterization: $\hat{\boldsymbol{y}} = \mathrm{argmax}_{\boldsymbol{y}} Score_{\boldsymbol{w}}(\boldsymbol{x}, \boldsymbol{y})$. Unlike many problems in NLP such as parsing or part-of-speech tagging, the small cardinality of Y_N makes enumerating all the alternative query segmentations computationally feasible.

In our preliminary experiments, we used a Support Vector Machine (SVM) ranker (Joachims, 2002) to learn the structured classifier.[2] We also investigated a Hidden Markov Model SVM (Altun et al., 2003) to label the segmentation breaks using information from past segmentation decisions. Ultimately, the mappings produced by these approaches were not as accurate as a simple formulation that creates a full query segmentation $\boldsymbol{y}$ as the combination of independent classification decisions made between each pair of tokens in the query.[3]

In the classification framework, the input is a query, $\boldsymbol{x}$, a position in the query, i, where $0 < i < N$, and the output is a segmentation decision yes/no. The training set of segmented queries is converted into examples of decisions between tokens and learning is performed on this set. At test time, $N-1$ segmentation decisions are made for the N-length query and an output segmentation $\boldsymbol{y}$ is produced. Here, features depend only on the input query $\boldsymbol{x}$ and the position in the query i. For a decision at position i, we use features from tokens up to three positions to the left and to the right of the decision location. That is, for a decision between x_{L0} and x_{R0}, we extract features from a window of six tokens in the query: $\{..., x_{L2}, x_{L1}, x_{L0}, x_{R0}, x_{R1}, x_{R2}, ...\}$. We now detail the features derived from this window.

3.2 Features

There are a number of possible indicators of whether a segmentation break occurs between a pair of tokens. Some of these features fire separately for each token x in our feature window, while others are defined over pairs or sets of tokens in the window. We first describe the features that are defined for the tokens around the decision boundary, x_{L0} and x_{R0}, before describing how these same features are extended to longer phrases and other token pairs.

3.2.1 Decision-boundary features

Table 1 lists the binary features that fire if particular aspects of a token or pair of tokens are present. For example, one of the *POS-tags* features will fire if the pair's part-of-speech tags are DT JJ, another feature will fire if the position of the pair in the to-

[1] See e.g. Collins (2002) for a popular training algorithm.

[2] A ranking approach was also used previously by Daumé III and Marcu (2004) for the CoNLL-99 nested noun phrase identification task.

[3] The structured learners did show large gains over the classification framework on the dev-set when using only the basic features for the decision-boundary tokens (see Section 3.2.1), but not when the full feature set was deployed. Also, features only available to structured learners, e.g. number of segments in query, etc., did improve the performance of the structured approaches, but not above that of the simpler classifier.

Table 1: Indicator features.

Name	Description
is-the	token x = "the"
is-free	token x = "free"
POS-tags	Part-of-speech tags of pair $x_{L0}\ x_{R0}$
fwd-pos	position from beginning, i
rev-pos	position from end $N - i$

ken is 2, etc. The two lexical features (for when the token is "the" and when the token is "free") fire separately for the left and right tokens around the decision boundary. They are designed to add discrimination for these common query words, motivated by examples in our training set. For example, in the training set, "free" often occurs in its own segment when it's on the left-hand-side of a decision boundary (e.g. "free" "online" ...), but may join into a larger segment when it's on the right-hand-side of a collocation (e.g. "sulfite free" or "sugar free"). The classifier can use the feature weights to encourage or discourage segmentation in these specific situations.

For statistical features, previous work (Section 2) suggests that the mutual information between the decision tokens x_{L0} and x_{R0} may be appropriate. The log of the pointwise mutual information (Church and Hanks, 1989) between the decision-boundary tokens x_{L0}, x_{R0} is:

$$\text{MI}(x_{L0}, x_{R0}) = \log \frac{\Pr(x_{L0}x_{R0})}{\Pr(x_{L0})\Pr(x_{R0})}$$

This is equivalent to the sum: $\log C(x_{L0}x_{R0}) + \log K - \log C(x_{L0}) - \log C(x_{R0})$. For web-based features, the counts $C(.)$ can be taken as a search engine's count of the number of pages containing the term. The normalizer K is thus the total number of pages on the Internet.

Represented as a summation, we can see that providing MI as the feature effectively ties the weights on the logarithmic counts $C(x_{L0}x_{R0})$, $C(x_{L0})$, and $C(x_{R0})$. Another approach would be to provide these logarithmic counts as separate features to our learning algorithm, which can then set the weights optimally for segmentation. We call this set of counts the "Basic" features. In Section 5, we confirm results on our development set that showed using the basic features untied increased segmentation performance by up to 4% over using MI – an important observation for all researchers using association models as features in their discriminative classifiers.

Table 2: Statistical features.

Name	Description
web-count	count of "x" on the web
pair-count	web count "$w\ x$"
definite	web count "the $w\ x$"
collapsed	web count "wx" (one word)
and-count	web count "w and x"
genitive	web count "w's x"
Qcount-1	Counts of "x" in query database
Qcounts-2	Counts of "$w\ x$" in database

Furthermore, with this technique, we do not need to normalize the counts for the other pairwise statistical features given in Table 2. We can simply rely on our learning algorithm to increase or decrease the weights on the logarithm of the counts as needed.

To illustrate how the statistical features work, consider a query from our development set: "star wars weapons guns." The phrase "star wars" can easily be interpreted as a phrase; there is a high co-occurrence count (pair-count), and many pages where they occur as a single phrase (collapsed), e.g. "starwars.com." "Weapons" and "guns," on the other hand, should not be joined together. Although they may have a high co-occurrence count, the coordination feature (and-count) is high ("weapons and guns") showing these to be related concepts but not phrasal constituents. Including this novel feature resulted in noticeable gains on the development set.

Since this is a query-based segmentation, features that consider whether sets of tokens occurred elsewhere in the query database may provide domain-specific discrimination. For each of the Qcount features, we look for two quantities: the number of times the phrase occurs as a query on its own and the number of times the phrase occurs within another query.[4] Including both of these counts also resulted in performance gains on the development set.

We also extensively investigated other corpus-based features, such as the number of times the phrase occurred hyphenated or capitalized, and the

[4]We exclude counts from the training, development, and testing queries discussed in Section 4.1.

corpus-based distributional similarity (Lin, 1998) between a pair of tokens. These features are not available from search-engine statistics because search engines disregard punctuation and capitalization, and collecting page-count-based distributional similarity statistics is computationally infeasible.

Unfortunately, none of the corpus-based features improved performance on the development set and are thus excluded from further consideration. This is perhaps not surprising. For such a task that involves real user queries, with arbitrary spellings and sometimes exotic vocabulary, gathering counts from web search engines is the only way to procure reliable and broad-coverage statistics.

3.2.2 Context Features

Although the tokens at the decision boundary are of paramount importance, information from the neighbouring tokens is also critical for segmentation decision discrimination. We thus include features that take into consideration the preceding and following tokens, x_{L1} and x_{R1}, as context information. We gather all the token indicator features for each of these tokens, as well as all pairwise features between x_{L1} and x_{L0}, and then x_{R0} and x_{R1}. If context tokens are not available at this position in the query, a feature fires to indicate this. Also, if the context features are available, we include trigram web and query-database counts of "x_{L1} x_{L0} x_{R0}" and "x_{L0} x_{R0} x_{R1}", and a fourgram spanning both contexts. Furthermore, if tokens x_{L2} and x_{R2} are available, we collect relevant token-level, pairwise, trigram, and fourgram counts including these tokens as well.

In Section 5, we show that context features are very important. They allow our system to implicitly leverage surrounding segmentation decisions, which cannot be accessed directly in an independent segmentation-decision classifier. For example, consider the query "bank loan amoritization schedule." Although "loan amoritization" has a strong connection, we may nevertheless insert a break between them because "bank loan" and "amoritization schedule" each have even stronger association.

3.2.3 Dependency Features

Motivated by work in noun phrase parsing, it might be beneficial to check if, for example, token x_{L0} is more likely to modify a later token, such as x_{R1}. For example, in "female bus driver", we might not wish to segment "female bus" because "female" has a much stronger association with "driver" than with "bus". Thus, as features, we include the pairwise counts between x_{L0} and x_{R1}, and then x_{L1} and x_{R0}. Features from longer range dependencies did not improve performance on the development set.

4 Experimental Setup

4.1 Data

Our dataset was taken from the AOL search query database (Pass et al., 2006), a collection of 35 million queries submitted to the AOL search engine. Most punctuation has been removed from the queries.[5] Along with the query, each entry in the database contains an anonymous user ID and the domain of the URL the user clicked on, if they selected one of the returned pages. For our data, we used only those queries with a click-URL. This subset has a higher proportion of correctly-spelled queries, and facilitates annotation (described below).

We then tagged the search queries using a maximum entropy part-of-speech tagger (Ratnaparkhi, 1996). As our approach was designed particularly for noun phrase queries, we selected for our final experiments those AOL queries containing only determiners, adjectives, and nouns. We also only considered phrases of length four or greater, since queries of these lengths are most likely to benefit from a segmentation, but our approach works for queries of any length. Future experiments will investigate applying the current approach to phrasal verbs, prepositional idioms and segments with other parts of speech.

We randomly selected 500 queries for training, 500 for development, and 500 for final testing. These were all manually segmented by our annotators. Manual segmentation was done with improving search precision in mind. Annotators were asked to analyze each query and form an idea of what the user was searching for, taking into consideration the click-URL or performing their own online searches, if needed. The annotators were then asked to segment the query to improve search retrieval, by forcing a search engine to find pages with the segments

[5] Including, unfortunately, all quotation marks, precluding our use of users' own segmentations as additional labelled examples or feature data for our system

occurring as unbroken units.

One annotator segmented all three data sets, and these were used for all the experiments. Two additional annotators also segmented the final test set to allow inter-annotator agreement calculation. The pairwise agreement on segmentation decisions (between each pair of tokens) was between 84.0% and 84.6%. The agreement on entire queries was between 57.6% and 60.8%. All three agreed completely on 219 of the 500 queries, and we use this "intersected" set for a separate evaluation in our experiments.[6] If we take the proportion of segmentation decisions the annotators would be expected to agree on by chance to be 50%, the *Kappa* statistic (Jurafsky and Martin, 2000, page 315) is around .69, below the .8 considered to be good reliability.

This observed agreement was lower than we anticipated, and reflects both differences in query interpretation and in the perceived value of different segmentations for retrieval performance. Annotators agreed that terms like "real estate," "work force," "west palm beach," and "private investigator" should be separate segments. These are collocations in the linguistics sense (Manning and Schütze, 1999, pages 183-187); we cannot substitute related words for terms in these expressions nor apply syntactic transformations or paraphrases (e.g. we don't say "investigator of privates"). However, for a query such as "bank manager," should we exclude web pages that discuss "manager of the bank" or "branch manager for XYZ bank"? If a user is searching for a particular webpage, excluding such results could be harmful. However, for query substitution or expansion, identifying that "bank manager" is a single unit may be useful. We can resolve the conflicting objectives of our two motivating applications by moving to a multi-layer query bracketing scheme, first segmenting unbreakable collocations and then building them into semantic units with a query segmentation grammar. This will be the subject of future research.

4.2 Experiments

All of our statistical feature information was collected using the Google SOAP Search API.[7] For training and classifying our data, we use the popular Support Vector Machine (SVM) learning package SVM^{light} (Joachims, 1999). SVMs are maximum-margin classifiers that achieve good performance on a range of tasks. In each case, we learn a linear kernel on the training set segmentation decisions and tune the parameter that trades-off training error and margin on the development set.

We use the following two evaluation criteria:

1. Seg-Acc: *Segmentation decision accuracy*: the proportion of times our classifier's decision to insert a segment break or not between a pair of tokens agrees with the gold standard decision.

2. Qry-Acc: *Query segmentation accuracy*: the proportion of queries for which the complete segmentation derived from our classifications agrees with the gold standard segmentation.

5 Results

Table 3 provides our results for various configurations of features and token-combinations as described in Section 3.[8] For comparison, a baseline that always chooses a segmentation break achieves 44.8% Seg-Acc and 4.2% Qry-Acc, while a system that inserts no breaks achieves 55.2% Seg-Acc and 4.0% Qry-Acc. Our comparison system is the MI approach used by Jones et al. (2006), which achieves 68% Seg-Acc and 26.6% Qry-Acc (Table 3). We let the SVM set the threshold for MI on the training set.

Note that the *Basic*, *Decision-Boundary* system (Section 3.2.1), which uses exactly the same co-occurrence information as the MI system (in the form of the *Basic* features) but allows the SVM to discriminatively weight the logarithmic counts, immediately increases Seg-Acc performance by 3.7%. Even more strikingly, adding the *Basic* count information for the *Context* tokens (Section 3.2.2) boosts performance by another 8.5%, increasing Qry-Acc by over 22%. Smaller, further gains arise by adding *Dependency* token information (Section 3.2.3).

Also, notice that moving from *Basic* features for the *Decision-Boundary* tokens to all of our indicator (Table 1) and statistical (Table 2) features (referred to as *All* features) increases performance from 71.7% to 84.3%. These gains convincingly justify

[6] All queries and statistical feature information is available at http://www.cs.ualberta.ca/~bergsma/QuerySegmentation/

[7] http://code.google.com/apis/soapsearch/

[8] Statistically significant intra-row differences in Qry-Acc are marked with an asterix (McNemar's test, p<0.05)

Table 3: Segmentation Performance (%)

Feature Type	Feature Span	Test Set		Intersection Set	
		Seg-Acc	Qry-Acc	Seg-Acc	Qry-Acc
MI	Decision-Boundary	68.0	26.6	73.8	34.7
Basic	Decision-Boundary	71.7	29.2	77.6	39.7
Basic	Decision-Boundary, Context	80.2	52.0*	85.6	62.1*
Basic	Decision-Boundary, Context, Dependency	81.1	53.2	86.2	64.8
All	Decision-Boundary	84.3	57.8*	86.6	63.5
All	Decision-Boundary, Context	86.3	63.8*	89.2	71.7*
All	Decision-Boundary, Context, Dependency	85.8	61.0	88.7	69.4

our use of an expanded feature set for this task. Including *Context* with the expanded features adds another 2%, while adding *Dependency* information actually seems to hinder performance slightly, although gains were seen when adding *Dependency* information on the development set.

Note, however, that these results must also be considered in light of the low inter-annotator agreement (Section 4.1). Indeed, results are lower if we evaluate using the test-set labels from another annotator (necessarily training on the original annotator's labels). On the intersected set of the three annotators, however, results are better still: 88.7% Seg-Acc and 69.4% Qry-Acc on the intersected queries for the full-featured system (Table 3). Since high performance is dependent on consistent training and test labellings, it seems likely that developing more-explicit annotation instructions may allow further improvements in performance as within-set and between-set annotation agreement increases.

It would also be theoretically interesting, and of significant practical importance, to develop a learning approach that embraces the agreement of the annotations as part of the learning algorithm. Our initial ranking formulation (Section 3.1), for example, could learn a model that prefers segmentations with higher agreement, but still prefers any annotated segmentation to alternative, unobserved structures. As there is growing interest in making maximal use of annotation resources within discriminative learning techniques (Zaidan et al., 2007), developing a general empirical approach to learning from ambiguously-labelled examples would be both an important contribution to this trend and a potentially helpful technique in a number of NLP domains.

6 Conclusion

We have developed a novel approach to search query segmentation and evaluated this approach on actual user queries, reducing error by 56% over a recent comparison approach. Gains in performance were made possible by both leveraging recent progress in feature engineering for noun compound bracketing, as well as using a flexible, discriminative incorporation of association information, beyond the decision-boundary tokens. We have created and made available a set of manually-segmented user queries, and thus provided a new testing platform for other researchers in this area. Our initial formulation of query segmentation as a structured learning problem, and our leveraging of association statistics beyond the decision boundary, also provides powerful tools for noun compound bracketing researchers to both move beyond three-word compounds and to adopt discriminative feature weighting techniques.

The positive results achieved on this important application should encourage further inter-disciplinary collaboration between noun compound interpretation and information retrieval researchers. For example, analysing the semantics of multiword expressions may allow for more-focused query expansion; knowing to expand "bank manager" to include pages describing a "manager of the bank," but not doing the same for non-compositional phrases like "real estate" or "private investigator," requires exactly the kind of techniques being developed in the noun compound interpretation community. Thus for query expansion, as for query segmentation, work in natural language processing has the potential to make a real and immediate impact on search-engine technology.

The next step in this research is to directly investigate how query segmentation affects search performance. For such an evaluation, we would need to know, for each possible segmentation (including no segmentation), the document retrieval performance. This could be the proportion of returned documents that are deemed to be relevant to the original query. Exactly such an evaluation was recently used by Kumaran and Allan (2007) for the related task of query contraction. Of course, a dataset with queries and retrieval scores may serve for more than evaluation; it may provide the examples used by the learning module. That is, the parameters of the contraction or segmentation scoring function could be discriminatively set to optimize the retrieval of the training set queries. A unified framework for query contraction, segmentation, and expansion, all based on discriminatively optimizing retrieval performance, is a very appealing future research direction. In this framework, the size of the training sets would not be limited by human annotation resources, but by the number of queries for which retrieved-document relevance judgments are available. Generating more training examples would allow the use of more powerful, finer-grained lexical features for classification.

Acknowledgments

We gratefully acknowledge support from the Natural Sciences and Engineering Research Council of Canada, the Alberta Ingenuity Fund, the Alberta Ingenuity Center for Machine Learning, and the Alberta Informatics Circle of Research Excellence.

References

Yasemin Altun, Ioannis Tsochantaridis, and Thomas Hofmann. 2003. Hidden markov support vector machines. In *ICML*.

Kenneth Ward Church and Patrick Hanks. 1989. Word association norms, mutual information, and lexicography. In *ACL*, pages 76–83.

Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *EMNLP*, pages 1–8.

Roxana Girju, Dan Moldovan, Marta Tatu, and Daniel Antohe. 2005. On the semantics of noun compounds. *Computer Speech and Language*, 19(4):479–496.

Hal Daumé III and Daniel Marcu. 2004. NP bracketing by maximum entropy tagging and SVM reranking. In *EMNLP*, pages 254–261.

Thorsten Joachims. 1999. Making large-scale Support Vector Machine learning practical. In B. Schölkopf and C. Burges, editors, *Advances in Kernel Methods: Support Vector Machines*, pages 169–184. MIT-Press.

Thorsten Joachims. 2002. Optimizing search engines using clickthrough data. In *ACM Conference on Knowledge Discovery and Data Mining*, pages 133–142.

Rosie Jones, Benjamin Rey, Omid Madani, and Wiley Greiner. 2006. Generating query substitutions. In *WWW*, pages 387–396.

Daniel Jurafsky and James H. Martin. 2000. *Speech and language processing*. Prentice Hall.

Frank Keller and Mirella Lapata. 2003. Using the web to obtain frequencies for unseen bigrams. *Computational Linguistics*, 29(3):459–484.

Giridhar Kumaran and James Allan. 2007. A case for shorter queries, and helping users create them. In *NAACL-HLT*, pages 220–227.

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 *HLT-NAACL*, pages 121–128.

Mark Lauer. 1995. Corpus statistics meet the noun compound: Some empirical results. In *ACL*, pages 47–54.

Dekang Lin. 1998. Automatic retrieval and clustering of similar words. In *COLING/ACL*, pages 768–773.

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

Preslav Nakov and Marti Hearst. 2005a. Search engine statistics beyond the n-gram: Application to noun compound bracketing. In *CoNLL*, pages 17–24.

Preslav Nakov and Marti Hearst. 2005b. Using the web as an implicit training set: application to structural ambiguity resolution. In *HLT/EMNLP*, pages 835–842.

Jeremy Nicholson and Timothy Baldwin. 2006. Interpretation of compound nominalisations using corpus and web statistics. In *ACL Workshop on Multiword Expressions*, pages 54–61.

Greg Pass, Abdur Chowdhury, and Cayley Torgeson. 2006. A picture of search. In *The First International Conference on Scalable Information Systems*.

Adwait Ratnaparkhi. 1996. A maximum entropy model for part-of-speech tagging. In *EMNLP*, pages 133–142.

Knut Magne Risvik, Tomasz Mikolajewski, and Peter Boros. 2003. Query segmentation for web search. In *WWW (Poster Session)*.

Omar Zaidan, Jason Eisner, and Christine Piatko. 2007. Using "annotator rationales" to improve machine learning for text categorization. In *NAACL-HLT*, pages 260–267.

Chengxiang Zhai. 1997. Fast statistical parsing of noun phrases for document indexing. In *ANLP*, pages 312–319.