On the Role of Lexical Features in Sequence Labeling

Yoav Goldberg* and **Michael Elhadad**
Ben Gurion University of the Negev
Department of Computer Science
POB 653 Be'er Sheva, 84105, Israel
{yoavg|elhadad}@cs.bgu.ac.il

Abstract

We use the technique of SVM anchoring to demonstrate that lexical features extracted from a training corpus are not necessary to obtain state of the art results on tasks such as Named Entity Recognition and Chunking. While standard models require as many as 100K distinct features, we derive models with as little as 1K features that perform as well or better on different domains. These robust reduced models indicate that the way rare lexical features contribute to classification in NLP is not fully understood. Contrastive error analysis (with and without lexical features) indicates that lexical features do contribute to resolving some semantic and complex syntactic ambiguities – but we find this contribution does not generalize outside the training corpus. As a general strategy, we believe lexical features should not be directly derived from a training corpus but instead, carefully inferred and selected from other sources.

1 Introduction

Common NLP tasks, such as Named Entity Recognition and Chunking, involve the identification of spans of words belonging to the same phrase. These tasks are traditionally reduced to a tagging task, in which each word is to be classified as either **B**eginning a span, **I**nside a span, or **O**utside of a span. The decision is based on the word to be classified and its neighbors. Features supporting the classification usually include the word forms themselves and properties derived from the word forms, such as prefixes, suffixes, capitalization information, and parts-of-speech. While early approaches to the NP-chunking task (Cardie and Pierce, 1998) relied on part-of-speech information alone, it is widely accepted that lexical information (word forms) is crucial for building accurate systems for these tasks. Indeed, all the better-performing systems in the CoNLL shared tasks competitions for Chunking (Sang and Buchholz, 2000) and Named Entity Recognition (Tjong Kim Sang, 2002; Tjong Kim Sang and De Meulder, 2003) make extensive use of such lexical information.

Is this belief justified? In this paper, we show that the influence of lexical features on such sequence labeling tasks is more complex than is generally assumed. We find that exact word forms aren't necessary for accurate classification. This observation is important because relying on the exact word forms that appear in a training corpus leads to over-fitting, as well as to larger models.

In this work, we focus on learning with Support Vector Machines (SVMs) (Vapnik, 1995). SVM classifiers can handle very large feature spaces, and produce state-of-the-art results for NLP applications (see *e.g.* (Kudo and Matsumoto, 2000; Nivre et al., 2006)). Alas, when trained on pruned feature sets, in which rare lexical items are removed, SVM models suffer a loss in classification accuracy. It would seem that rare lexical items are indeed crucial for SVM classification performance. However, in Goldberg and Elhadad (2007), we suggested that the SVM learner is using the rare lexical features for singling out hard cases rather than for learning meaningful generalizations. We provide further evidence to support this claim in this paper.

*Supported by the Lynn and William Frankel Center for Computer Sciences, Ben Gurion University

Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pages 1142–1151,
Singapore, 6-7 August 2009.

We show that by using a variant of SVM – *Anchored SVM Learning* (Goldberg and Elhadad, 2007) with a polynomial kernel, one can learn accurate models for English NP-chunking (Marcus and Ramshaw, 1995), base-phrase chunking (CoNLL 2000), and Dutch Named Entity Recognition (CoNLL 2002), on a heavily pruned feature space. Our models make use of only a fraction of the lexical features available in the training set (less than 1%), and yet provide highly-competitive accuracies.

For the Chunking and NP-Chunking tasks, the most heavily pruned experiments, in which we consider only features appearing at least 100 times in the training corpus, do show a small but significant drop in accuracy on the testing corpus compared to the non-pruned models exposed to all available features in the training data. We provide detailed error analysis of a development set in Section 6, revealing the causes for these differences. We suggest one additional binary feature in order to account for some of the performance gap. Moreover, we show that the differences in accuracy vanish when the lexicalized and unlexicalized models are tested on text from slightly different sources than the training corpus (Section 7).

This goes to show that with an appropriate learning method, orthographic and structural (in the form of POS tag sequences) information is sufficient for achieving state-of-the-art performance on these kind of sequence labeling tasks. This does not mean semantic information is not needed for these tasks. It does mean that current models capture only a tiny amount of such semantic information through rare lexical features, and in a manner that does not generalize well.

We believe this data motivates a different strategy to incorporate lexical features into classification models: instead of collecting the raw lexical forms appearing in a training corpus, we should attempt to actively construct a feature space including lexical features derived from external sources. The feature representation of (Collobert and Weston, 2008) could be a step in that direction. We also believe that hard cases for sequence labeling (POS ambiguity, coordination, long syntactic constructs) could be directly approached with specialized classifiers.

1.1 Related Work

This work complements a similar line of results from the parsing literature. While it was initially believed that lexicalization of PCFG parsers (Collins, 1997; Charniak, 2000) is crucial for obtaining good parsing results, Gildea (2001) demonstrated that the lexicalized Model-1 parser of Collins (1997) does not benefit from bilexical information when tested on a new text domain, and only marginally benefits from such information when tested on the same text domain as the training corpora. This was followed by (Bikel, 2004) who showed that bilexical-information is used in only 1.49% of the decisions in Collins' Model-2 parser, and that removing this information results in "an exceedingly small drop in performance". However, uni-lexical information was still considered crucial. Klein and Manning (2003) bridged the gap between lexicalized and unlexicalized parsing performance, providing a competitive unlexicalized parsing model, relying on lexical information for only a few closed-class lexical items. This was recently followed by (Matsuzaki et al., 2005; Petrov et al., 2006) who introduce state-of-the-art nearly unlexicalized PCFG parsers.

Similarly for discriminative dependency parsing, state-of-the-art parsers (McDonald, 2006; Nivre et al., 2006) are highly lexicalized. However, the model analysis in (McDonald, 2006) reveals that bilexical features hardly contribute to the performance of a discriminative MST-based dependency parser, while Kawahara and Uchimoto (2007) demonstrate that minimally-lexicalized shift-reduce based dependency parsers can produce near state-of-the-art accuracy.

In this work, we address the same question of determining the impact of lexical features on a different family of tasks: sequence labeling, as illustrated by named entity recognition and chunking. As discussed above, all state-of-the-art published methods rely on lexical features for such tasks (Zhang et al., 2001; Sha and Pereira, 2003; Finkel et al., 2005; Ratinov and Roth, 2009). Sequence labeling includes both a structural aspect (bracketing the chunks) and a tagging aspect (classifying the chunks). While we expect the structural aspect can benefit from techniques similar to those used in the parsing literature, it is unclear whether the tagging component could perform well without detailed lexical information. We demonstrate in this work that, indeed, lexical features are not necessary to obtain competitive performance. Our approach consists in performing a detailed analysis

of the role played by rare lexical features in SVM models. We distinguish the information brought to the model by such features from the role they play in a specific learning method.

2 Learning with Less Features

We adopt the common feature representation in which each data-point is represented as a sparse D dimensional binary-valued vector f. Each of the D possible features f_i is an indicator function. The indicator functions look at properties of the current or neighbouring words. An example of such function f_i is `1 iff the previous word-form is DOG, 0 otherwise`. The lexical (word-form) features result in extremely high-dimensional (yet very sparse) feature vectors – each word-form in the vocabulary of the training set correspond to (at-least) one indicator function.

Due to the Zipfian distribution of language data, many of the lexical features are very rare, and appear only a couple times in the training set. Ideally, we would like our classifiers to learn only from robust features: consider only features that appear at least k times in the training data (rare-feature pruning). These features are more likely to appear in unseen test data, and thus such features can support more robust generalization.

However, we find empirically that performing such feature pruning *prior* to learning SVM models hurts the performance of the learned models. Our intuition is that this sensitivity to rare lexical features is not explained by the richness of information such rare features bring to the model. Instead, we believe that rare lexical features help the classifier because they make the data *artificially* more separable. To demonstrate this claim, we experiment with anchored SVM, which introduces artificial mechanical anchors into the model to achieve separability, and make rare lexical features unnecessary.

3 Learning Method

SVM are discriminative, max-margin, linear classifiers (Vapnik, 1995), which can be kernelized. For the formulation of SVMs in the context of NLP applications, see (Kudo and Matsumoto, 2001). SVMs with a polynomial kernel of degree 2 were shown to provide state-of-the-art performance in many NLP application, see for example (Kudo and Matsumoto, 2000; Nivre et al., 2006; Isozaki and Kazawa, 2002; Goldberg et al., 2006).

SVMs cope with inseparable data by introducing a *soft-margin* – allowing some of the training instances to be classified incorrectly subject to a penalty, controlled by a parameter C.

Anchored SVM As we show in Section 5, the soft-margin heuristic performs sub-optimally for NLP tasks when the data is inseparable. We use instead the *Anchored Learning* heuristic, introduced in (Goldberg and Elhadad, 2007). The idea behind anchored learning is that some training instances are inherently ambiguous. This ambiguity stems from ambiguity in language structure, which cannot be resolved with a given feature representation. When a data-point cannot be classified, it might be due to missing information, which is not available in the data representation. Instead of allowing ambiguous items to be misclassified during training, we make the training data artificially separable. This is achieved by adding a unique feature to each training example (an *anchor*). These anchor features cause each data-point to be slightly more similar to itself than to any other data point. At test time, we remove anchor features.
In terms of kernel-based learning, anchored learning can be achieved by redefining the dot product between two vectors to take into account the identity of the vectors: $x_i \cdot_{anc} x_j = x_i \cdot x_j + \delta_{ij}$.

The classifier learned over the anchored data takes into account the fine interactions between the various inseparable data points. In our experiments, SVM models over anchored data have many more support vectors than soft-margin SVM models. However, the anchored models generalize much better when less features are available.

Relation to L2 SVM The classic soft-margin SVM formulation uses L1-penalty for misclassified instances. Specifically, the objective of the learner is to minimize $\frac{1}{2}||w||^2 + C\sum_i \xi_i$ subject to some margin constraints, where w is a weight vector to be learned and ξ_i is the misclassification error for instance i. This is equivalent to maximizing the dual problem:
$\sum_{i=1}^{M} \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j K(x_i, x_j)$

Another variant is L2-penalty SVM (Koshiba and Abe, 2003), in which there is a quadratic penalty for misclassified instances.
Here, the learning objective is to minimize: $\frac{1}{2}||w||^2 + \frac{1}{2}C\sum_i \xi_i^2$ or alternatively maximize the dual: $\sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j (K(x_i, x_j) + \frac{\delta_{ij}}{C})$.

Interestingly, for the linear kernel, SVM-

anchoring reduces to L2-SVM with C=1. However, for the case of non-linear kernels, anchored and L2-SVM produce different results, as the anchoring is applied prior to the kernel expansion. Specifically for the case of the second-degree polynomial kernel, L2-SVM aims to maximize:

$\sum_i \alpha_i - \frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y_i y_j((x_i \cdot x_j + 1)^2 + \frac{\delta_{ij}}{C})$,

while the anchored-SVM variant would maximizes: $\sum_i \alpha_i - \frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y_i y_j(x_i \cdot x_j + \delta_{ij} + 1)^2$.

In our experiments, as discussed in Section 5.4, we find that anchored-SVM and soft-margin SVM with tuned C value both reach good results when we reduce the amount of lexical features. Anchored-SVM, however, does not require fine-tuning of the error-parameter C since it insures separability. As a result, we learn anchored-SVM models quickly (few hours) as opposed to several days per model for C-tuned soft-margin SVM. Anchored-SVMs also provide an easy explanation of the role of features in terms of separability. Therefore, we use anchored-SVMs in our experiments as the learning method, but we expect that other learning methods are capable of learning with the same reduced feature sets.

4 Experiment Setup

How important are the rare lexical features for learning accurate NLP models? To investigate this question, we experiment with 3 different NLP sequence-labeling tasks. For each task, we train a sequence of polynomial kernel (d=2) SVM classifiers, using both soft-margin (C=1) and anchored SVM. Each classifier is trained on a pruned feature set, in which only features appearing at least k times in the training data are kept. We vary the pruning parameter k. Pruning is performed over all the features in the model, but lexical features are most affected by it.

For all the models, we use the B-I-O representation, and perform multiclass classification using pairwise-voting. For our features, we consider properties of tokens in a 5-token window centered around the token to be classified, as well as the two previous classifier predictions. Results are reported as F-measure over labeled identified spans.

Polynomial vs. Linear models The polynomial kernel of degree 2 allows us to efficiently and implicitly include in our models all feature pairs. Syntactic structure information as captured by pairs of POS-tags and Word-POS pairs is certainly important for such syntactic tasks as Chunking and NER, as demonstrated by the many systems described in (Sang and Buchholz, 2000; Tjong Kim Sang, 2002). By using the polynomial kernel, we can easily make use of this information without intensive feature-tuning for the most successful feature pairs.

L1-SVM, L2-SVM and the choice of the C parameter Throughout our experiments, we use the "standard" variant of SVM, L1-penalty soft margin SVM, as implemented by the TinySVM[1] software package, with the default C value of 1. This setting is shown to produce good results for sequence labeling tasks in previous work (Kudo and Matsumoto, 2000), and is what most end-users of SVM classifiers are likely to use. As we show in Sect.5.4, fine-tuning the C parameter reaches better accuracy than L1-SVM with C=1. However, as this fine-tuning is computationally expensive, we first report the comparison L1-SVM/C=1 vs. anchored-SVM, which consistently reached the best results, and was the quickest to train.

Feature Pruning vs. Feature Selection Our aim in this set of experiments is not to find the optimal set of lexical features, but rather to demonstrate that most lexical items are not needed for accurate classification in sequence labeling tasks. To this end, we perform very crude frequency based feature pruning. We believe better motivated feature selection technique taking into account linguistic (*e.g. prune only open-class words*) or statistic information could result in slightly more accurate models with even fewer lexical items.

5 Experiments and Results

5.1 Named Entity Recognition (NER)

We use the Dutch data set from the CoNLL 2002 shared task (Tjong Kim Sang, 2002). The aim is to identify named entities (persons, locations, organizations and miscellaneous) in text. The task has two stages: identification of the entities, and classification of the identified entities into their corresponding types. We focus here on the identification task.

Features: We use the following properties for each of the relevant tokens: word-form, POS, ORT, prefix1, prefix2, prefix3, suffix1, suffix2, suffix3. The ORT feature can take one of the following values: {number, contains-digit, contains-hyphen, capitalized, all-capitalized, URL, punctuation, regular}.

[1] http://chasen.org/~taku/software/TinySVM/

PRUNING	#FEATURES	SOFT-MARGIN	ANCHORED
0	186,421	90.92	90.78
100	5,804	90.73	90.75
1000	1,207	88.56	90.10
1500	821	85.92	89.29

Table 1: Named Entity Identification results (F-score) on dev set, with various pruning thresholds.

Results are presented in Table 1. Without feature pruning, we achieve an F-score of 90.9. This dataset proved to be quite resilient to feature pruning. Pruning features appearing less than 100 times results in just a slight decrease in F-score. Extremely aggressive pruning, keeping only features appearing more than 1,000 or 1,500 times in the training data, results in a big drop in F-score for the soft-margin SVM (from about 91 to 86). Much less so for the Anchored-SVM. Using Anchored SVM we achieve an F-score of 90.1 after pruning with $k = 1,000$. This model has 1207 active features, and 27 unique active lexical forms.

5.2 NP Chunking

The goal of this task (Marcus and Ramshaw, 1995) is the identification of non-recursive NPs. We use the data from the CoNLL 2000 shared task: NP chunks are extracted from Sections 15-18 (train) and 20 (test) of the Penn WSJ corpus. POS tagged are automatically assigned by the Brill Tagger.

Features: We consider the POS and word-form of each token.

PRUNING	#FEATURES	SOFT-MARGIN	ANCHORED
0	92,805	94.12	94.08
1	46,527	93.78	94.09
2	32,583	93.58	94.00
5	18,092	93.42	94.01
10	10,812	93.00	93.98
20	5,952	92.48	93.92
50	2,436	92.33	93.96
100	1,168	91.94	93.83

Table 2: NP-Chunking results (F-score), with various pruning thresholds.

Results are presented in Table 2. Without feature pruning ($k = 0$), the soft-margin SVM performs slightly better than the Anchored-SVM. Either of the results are state-of-the-art for this task. However, even modest pruning ($k = 2$) hurts the soft-margin model significantly. Not so for the anchored-SVM. Even with relatively aggressive pruning ($k = 100$), the anchored model still achieves an impressive F-score of 93.83. Remarkably, in that last model, there are only 1,168 active features, and only 209 unique active lexical forms.

5.3 Chunking

The goal of the Chunking task (Sang and Buchholz, 2000) is the identification of an assortment of linguistic base-phrases. We use the data from the CoNLL 2000 shared task.

Features: We perform two experiments. In the first experiment, we consider the POS and word-form of each token. In this setting, feature pruning resulted in a bigger loss in performance than in the two previous tasks. Preliminary error analysis revealed that many errors are due to tagger errors, especially of the present participle forms. This led us to the second experiment, in which we added as features the 2- and 3- letter suffixes for the word to be classified (but not for the surrounding words).

Results are presented in Tables 3 and 4. In the first experiment (POS + Word), the non-pruned soft-margin model is the same system as the top-performing system in the original shared task, and yields state-of-the-art results. Unlike the NP-chunking case, here feature pruning has a relatively large impact on the results even for the anchored models. However, the anchored models are still far more robust than the soft-margin ones. With $k = 100$ pruning, the soft-margin model suffers a drop of 2.5 F points, while the anchored model suffers a drop of only 0.84 F points. Even after this drop, the anchored $k = 100$ model still performs above the top-third system in the CoNLL 2000 shared task. This anchored $k = 100$ model has 1,180 active features, and only 209 unique active lexical features.

The second experiment (POS + word-form + suffixes for main word) adds crude morphological information to the learner, helping it to avoid common tagger mistakes. This additional information is helpful: pruning with $k = 100$ leads to an accurate anchored model (93.12 F) with only 209 unique lexical items. Note that with the addition of the suffix features, the pruned model $k = 20$ beats the purely lexical model (no suffix features) with no pruning (93.51 vs. 93.44) with 10 times less features. When we combine suffixes and all lexical forms, we still see a slight advantage to the lexical model (93.73 vs. 93.12 with pruning at $k = 100$).

Even less lexicalization How robust are the suffixes? We performed a third experiment, in which

PRUNING	#FEATURES	SOFT-MARGIN	ANCHORED
0	92,837	93.44	93.40
1	46,557	93.20	93.32
2	32,614	93.10	93.31
5	18,126	92.89	93.29
10	10,834	92.73	93.23
20	5,975	92.18	93.16
50	2,463	91.80	92.89
100	1,180	90.94	92.56

Table 3: Chunking results (F), with various pruning thresholds. Experiment 1. Features: POS, Word.

PRUNING	#FEATURES	SOFT-MARGIN	ANCHORED
0	104,304	93.73	93.69
1	72,228	93.56	93.68
2	57,578	93.50	93.64
5	37,210	93.35	93.62
10	23,968	93.26	93.56
20	14,060	92.84	93.51
50	6,326	92.28	93.37
100	3,340	91.83	93.12

Table 4: Chunking results (F), with various pruning thresholds. Experiment 2. Features: POS, Word, {Suff2, Suff3} of main Word.

we replaced any explicit word-forms by 2- and 3-letter suffixes. This gives us the complete word form of many function words, and a reasonable amount of morphological marking. Results are presented in Table 5. Surprisingly, this information proves to be quite robust. Without feature pruning, both the anchored and soft-margin model achieve near state-of-the-art performance of 93.25F. Pruning with $k = 100$ hurts the result of the soft-margin model, but the anchored model remains robust with an F-score of 93.18. This last model has 2,563 active features. With further pruning ($k = 250$), the result of the anchored model drops to 92.87F (still 3rd place in the CoNLL shared task), with only 1,508 active features in the model.

5.4 Fine-tuned soft-margin SVMs

For the sake of completeness, and to serve as a better comparison to the soft-margin SVM, we report results of some experiments with both L1 and L2 SVMs, with tuned C values. NP-chunking performance with tuned C values and various pruning thresholds is presented in Table 6.

For these results, the C parameter was tuned on a development set using Brent's 1-dimension minimization method (Brent, 1973). While taking about 40 hours of computation to fit, the final results catch up with those of the anchored-SVM but still remain slightly lower. This further highlights our main point: accurate models can be achieved also with mostly unlexicalized models, and the lexical features do not contribute substantial semantic information, but rather affect the separability of the data. This is nicely demonstrated by SVM-anchoring, in which lexical information is practically replaced by artificial semantically void indexes, but similar performance can also be achieved by fine-tuning other learning parameters.

PRUNING	#FEATURES	SOFT-MARGIN	ANCHORED
0	19,910	93.25	93.23
100	2,563	92.87	93.18
250	1,508	92.40	92.87

Table 5: Chunking results (F), with various pruning thresholds. Experiment 3. Features: POS , Suff2, Suff3 .

K	L1 (C)	L2 (C)	ANCHORED
0	94.12 (1.0001)	94.09 (2.6128)	94.08
50	93.79 (0.0524)	93.71 (0.0082)	93.96
100	93.72 (0.0567)	93.59 (0.0072)	93.83

Table 6: NP-Chunking results (F), with various pruning thresholds K, for L1 and L2 SVMs with tuned C values

6 Error Analysis

Our experiments so far indicate that very aggressive feature pruning hurts performance slightly (by about 0.5F point). The feature-pruned models are still accurate, indicating that lexical features contribute little to the classification accuracy. We now investigate the differences between the lexicalized and pruned models, in order to characterize the kind of information that is available to the lexicalized models but missing from the pruned ones. In the next section, we also verify that pruned-models are more stable than the fully lexicalized ones when tested over different text genres and domains.

We focus our analysis on the chunking task, which is a superset of the NP-chunking task. We compare the fully lexicalized soft-margin SVM model with the POS+suffix2+suffix3 anchored-SVM model with $k = 100$ pruning. We analyze the models' respective performance on section 05 of the WSJ corpus. This dataset is different than the official test set. It is, however, part of the same annotated corpus as both the training and test sets. On this dataset, the fully lexicalized SVM model

achieves an F-score of 93.24, vs. 92.59 for the suffix-based pruned anchored-SVM model. (The pruned anchored-SVM model ($k = 100$) from experiment 2, achieve a slightly higher F-score of 92.84)

We investigate only those chunks which are identified correctly by one model but not by the other. Overall, there are 440 chunks (363 unique) which are identified correctly only by the lexicalized model, and 258 chunks (232 unique) only by the pruned model.

Where the pruned model is always wrong

Some errors are unique to the pruned model.

Over 45 of the cases that are identified correctly only in the lexicalized model (more than 10%) are due to the words "including" (18 cases) and "If" (9 cases), as well as other -ing forms such as "following", "according", "rising" and "suspecting."

The word "including" appears 80 times in the training data, always tagged as VBG and functioning as a PP chunk, which is an odd chunk for VBGs. The lexicalized model easily picked up on this behaviour, while the pruned model couldn't. Similarly, the word "following/VBG" appears 32 times, 20 of which as PP, and the word "according/VBG" 53 times, all of them as PP. The pruned model could not distinguish those from the rest of the VBGs and tagged them as VPs. What seems to happen in these cases, is that certain verbal forms participate in idiomatic constructions and behave syntactically as prepositions. The POS tagger does not pick this ambiguity in function and contributes only the most likely tag for the words (VBG). Lexical models learn that certain VBGs are "becoming" prepositions in the observed dataset. These words do not appear as specific features in the pruned models, and hence these usage shifts are often misclassified. Interestingly, the pruned model did learn that verbal forms can sometimes be PPs: it made use of that information by mis-identifying 11 verbal VBGs and 6 verbal VBNs as PPs.

The word "If/IN", unlike most prepositions, it always starts an SBAR rather than a PP chunk in the corpus. The pruned model learned this behaviour correctly for the lower-cased "if/IN", but missed the upper-cased version appearing in 79 sentence initial locations in the corpus.

These cases are caused by a mismatch between the POS tag and the syntactic function observed in the chunked dataset.

Additional cases include the adverbs (Already, Nearby, Soon, Maybe, Perhaps, once, Then): they are sometimes not chunked as ADVP but are left outside of any chunk. Some one-word ADJP chunks being chunked as NPs (short, general, sure, worse, ...) (6 cases) and some are chunked as ADVPs (hard, British-born, ...) (4 cases).

There are 10 cases where the pruned model splits an NP into ADVP and NP, such as: [later] [this week], [roughly][18 more U.S. stores]. In addition, the pruned model failed to learn the construction "typical of", resulting in 2 NP chunks such as: [The more intuitive approach typical].

Some mistakes of the pruned model seem like mistakes/pecularities of the annotated corpus, which the lexicalized model found a way to work around. Consider the following gold-standard cases from the annotated corpus:

- [VP seems] [ADVP rarely] [VP to cut]
- [ADVP just] [PP after]
- [VP is] [NP anything] [O but] [VP fixing]
- [ADJP as high] [PP as] [NP 8.3 %]
- [ADJP less] [PP than] [ADJP rosy]
- [NP 40 %] [PP to] [NP 45 %]

Which were each identified as a single chunk by the pruned model. It can be argued these are mistakes in the tagged dataset.

Where the lexical model is sometimes better

Both models fail on conjunctions, but the lexicalized model do slightly better. Conjunction error types come in two main varieties, either chunking [x][and][y] instead of [x and y] (pruned: 21 cases, lex: 14 cases) or chunking [x and y] instead of [x][and][y] (pruned: 26 cases, lex: 24 cases).

Joining VP and NP into an NP, due to a verb/adj ambiguity. For example chunking [NP fired six executives] instead of [VP fired] [NP six executives], or [NP keeping viewers] instead of [VP keeping] [NP viewers]. 12 such cases are resolved correctly only by the lexicalized model, and 5 only by the pruned one.

SBAR/PP confusion for words such as: "as","after","with","since" (both ways). 13 cases for the pruned model, 6 for lexicalized one.

Where both model are similar

Merging back-to-back NPs: Both models tend to erroneously join back-to-back NPs to a single NP, *e.g.* : [NP Westinghouse this year], or [NP themselves fashion enterprises]. No model is better than the other on these cases, each model failed

on 16 cases the other model succeeded on.

Joining NP and VP into an NP due to Verb/Noun ambiguity and tagger mistakes:

- [NP the weekend] [VP making] → [NP the weekend making]
- [NP the competition] [VP can] → [NP the competition can]

(lexicalized: 6 errors, pruned: 8 errors)

And splitting some NPs to VP+NP due to the same reasons:

- [VP operating] [NP profit]
- [VP improved] [NP average yield]

(lexicalized: 5 errors, pruned: 7 errors)

The word "that" is confused between SBAR and NP (5 mistakes for each model)

Erroneously splitting range NPs, *e.g.* :

- [about $115][to][$125] (2 cases for each model).

Where the pruned model is better

There are some cases where the pruned models is doing better than the lexicalized one:

VP wrongly split into VP and ADJP:

- [remains] [banned]

4 mistakes for lexicalized, 1 for pruned

VP wrongly split into VP and VP:

- [were scheduled] [to meet]
- [used] [to complain]

3 mistakes for lexicalized, 1 for pruned

VP wrongly split into ADVP and VP:

- [largly][reflecting]
- [selectively][leaking]

6 mistakes for lexicalized, 1 for pruned

PP and SBAR confusion:

- of, with, As, after

9 mistakes for lex, 5 for pruned

VP chunked as NP due to tagger mistake:

- [NP ruling], [NP drives], [NP cuts]

6 mistakes for lex, 2 for pruned

"that" tagged as NP instead of SBAR:

2 mistakes for lex, 0 for pruned

To conclude

Both the pruned and the fully lexicalized models have problems dealing with non-local phenomena such as coordination and relative clauses, as well as verb/adjective ambiguities and VBG/Noun ambiguities. They also perform poorly on embeded syntactic constructions (such as an NP containing an ADJP), and on identification of back-to-back NPs, which often requires semantic knowledge.

Both models suffer from tagging mistakes of the underlying tagger and systematic ambiguity between the morphological tag assigned by the tagger and the syntactic tag in which the word operates (e.g., "including" used as a preposition).

The main advantage of the fully lexcialized model is in dealing with:

- Some coordinated constructions.
- Some cases of verb/adjective ambiguities.
- Specific function words not seen much in training.
- Idiomatic usages of some VBG/VBN forms functioning as prepositions.

The first two items are semantic in nature, and hint that lexical features do capture some semantic information. While this might be true on the specific corpus, we believe that such corpus-derived semantic knowledge is very restricted, is not generalizable, and will not transfer well to other corpora, even on the same genre. We provide evidence for this claim in Section 7.

The last two items are syntactic. We address them by introducing a slightly modified feature model.

6.1 Another chunking Experiment

Based on the observations from the error analysis, we performed another pruned-chunking experiment, with the following features:

- Word and POS for a -2,+2 window around the current token, and 2-and-3-letter suffixes of the token to be classified (same as Experiment 2 in Section 5.2 above).
- Features of words appearing as a preposition (IN) anywhere in the training set are not pruned (this result in a model with 310 unique lexical items after $k = 100$ pruning).
- *An additional binary feature* indicating for each token whether it can function as a PP. The list of possible-PP forms is generated by considering all tokens seen inside a PP in the training corpus. It can be easily extended if additional lexicographic resources are available, without retraining the model.

This last proposed feature incorporates important lexical knowledge without relying on features for specific lexical forms, and is more generalizable. The accuracy of this new model on the development and test set with various pruning thresholds is presented in Table 7.

The addition of the CanBePrep feature improves the fully-lexicalized model accuracy on the development set (93.24 to 93.68), and does not affect fully lexicalized result on the test set (93.71

CORPUS	SOURCE	CONTENT	#TOKENS
WSJ	4 articles from wsj.com business	Magazine, business	2,671
Jaguar	Wikipedia page on Jaguar	Well edited text, animals	5,396
FreeWill	Wikipedia page on Free Will	Well edited text, philosophy	9,428
LJ-Life	4 LiveJournal posts	Noisy teenage writing, life	870

Table 8: Corpus Variation Text Sources

PRUNING	#FEATURES	SOFT-MARGIN	ANCHORED
Dev Set			
0	92,989	93.71	–
100	4,066	–	93.22
Test Set			
0	92,989	93.68	–
100	4,066	–	93.26

Table 7: Chunking results (F), with various pruning thresholds. Experiment 4. Features: POS, Word , Suff2, Suff3 for main word, CanBePrep .

vs. 93.73). The pruned model performance improves in both cases, more so on the development set (93.12 to 93.22 on the test set, 92.84 to 93.26 on the development set). The new model helps bridging the gap between the fully lexicalized and the pruned model, yet we still observe a lead of 0.4F for the fully lexicalized model. We now turn to explore how meaningful this difference is in real-world situation in which one does not operate on the Penn-WSJ corpus.

7 Corpus Variation and Model Performance

When tested on the exact same resource as the models are trained on, the fully lexicalized model still has a slight edge over the pruned ones. How well does this lexical knowledge transfer to different text genres? We compare the models' performance on text from various genres, ranging from very similar to the training material (recent articles from the WSJ Business section) to a well-edited but different domain text ("Featured-content" wikipedia pages) to a non-edited noisy text (live-journal blog posts from the "life" category). As we do not have gold-annotated data for these text genres, we analyze the few differences between the models, manually inspecting the instances on which the models disagree.

Table 8 describes our test corpora for this experiment. We applied the fully-lexicalized and the pruned ($k = 100$) anchored models described in Section 6.1 to these texts, and compared the chunking results. The results are presented in Table 9.

When moving outside of the canonic training

TEXT	#DIFF	PRUNED CORRECT	LEX CORRECT	BOTH WRONG
WSJ	13	**9**	4	0
Jaguar	45	20	20	7
FreeWill	118	**51**	38	29
LJ-Life	15	**8**	6	1

Table 9: Comparison of Models' performance on different text genres

corpus, the fully lexicalized model have no advantage over the heavily pruned one. On the contrary, the pruned models seem to have a small advantage in most cases (though it is hard to tell if the differences are significant). This is true even for texts in the very same domain, genre and editing guidelines as the training corpus was derived from.

8 Discussion

For all the sequence labeling tasks we analyzed, the anchored-SVM proved to be robust to feature pruning. The experiments support the claim that rare lexical features do not provide substantial information to the model, but instead play a role in maintaining separability. When this role is taken over by anchoring, we can obtain the same level of performance with very few robust lexical features. Yet, we cannot conclude that lexical information is not needed. There is a significant difference between the pruned and non-pruned models for the chunking task. We showed that this difference can be bridged to some extent by a binary feature relating to idiomatic word usage, and that the difference vanishes when testing outside of the annotated corpus. The high classification accuracies achieved with the heavily pruned anchored-SVM models sheds new light on the actual role of lexical features, and indicating that there is still a lot to be learned regarding the effective incorporation of lexical and semantic information into our models. It is our view that semantic knowledge should not be expected to be learned by inspection of raw lexical counts from an annotated text corpus, but instead collected from sources external to the annotated corpora – either based on a very large unannotated corpora, or on manually constructed lexical resources.

References

Daniel M. Bikel. 2004. Intricacies of collins' parsing model. *Computational Linguistics*, 30(4).

Richard P. Brent, 1973. *Algorithms for Minimization without Derivatives*, chapter 4. Prentice-Hall.

Claire Cardie and David Pierce. 1998. Error-driven pruning of treebank grammars for base noun phrase identification. In *ACL-1998*.

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *Proc of NAACL*.

Michael Collins. 1997. Three generative, lexicalized models for statistical parsing. In *Proc of EACL*.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proc. of ICML*.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proc of ACL*.

Daniel Gildea. 2001. Corpus variation and parser performance. In *Proc of EMNLP*.

Yoav Goldberg and Michael Elhadad. 2007. SVM Model Tampering and Anchored Learning: A Case Study in Hebrew. NP Chunking. In *ACL2007*.

Yoav Goldberg, Meni Adler, and Michael Elhadad. 2006. Noun Phrase Chunking in Hebrew: Influence of Lexical and Morphological Features. In *COLING/ACL2006*.

Hideki Isozaki and Hideto Kazawa. 2002. Efficient Support Vector Classifiers For Named Entity Recognition. In *COLING2002*.

Daisuke Kawahara and Kiyotaka Uchimoto. 2007. Miniamlly lexicalized dependency parsing. In *Proc of ACL (Short papers)*.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proc. of ACL*.

Yoshiaki Koshiba and Shigeo Abe. 2003. Comparison of L1 and L2 support vector machines. In *Proc. of the International Joint Conference on Neural Networks*, volume 3.

Taku Kudo and Yuji Matsumoto. 2000. Use of Support Vector Learning for Chunk Identification. In *CoNLL-2000*.

Taku Kudo and Yuji Matsumoto. 2001. Chunking with support vector machines. In *NAACL '01*.

Mitch P. Marcus and Lance A. Ramshaw. 1995. Text Chunking Using Transformation-Based Learning. In *3rd ACL Workshop on Very Large Corpora*.

Takuya Matsuzaki, Yusuke Miyao, and Jun'ichi Tsujii. 2005. Probabilistic cfg with latent annotations. In *Proc of ACL*.

Ryan McDonald. 2006. *Discriminative Training and Spanning Tree Algorithms for Dependency Parsing*. Ph.D. thesis, University of Pennsylvania.

Joakim Nivre, Johan Hall, and Jens Nillson. 2006. MaltParser: A Data-Driven Parser-Generator for Dependency Parsing. In *LREC2006*.

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *Proc of ACL*.

Lev Ratinov and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In *Proc of CONLL*.

Erik F. Tjong Kim Sang and S. Buchholz. 2000. Introduction to the CoNLL-2000 shared task: chunking. In *CoNLL-2000*.

Fei Sha and Fernando Pereira. 2003. Shallow parsing with conditional random fields. In *Proc of NAACL*.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition. In *CoNLL-2003*.

Erik F. Tjong Kim Sang. 2002. Introduction to the CoNLL-2002 Shared Task: Language-Independent Named Entity Recognition. In *CoNLL-2002*.

Vladimir Vapnik. 1995. *The nature of statistical learning theory*. Springer-Verlag New York, Inc.

Tong Zhang, Fred Damerau, and David Johnson. 2001. Text chunking using regularized winnow. In *Proc of ACL*.