

Distributional Representations for Handling Sparsity in Supervised Sequence-Labeling

Fei Huang

Temple University
1805 N. Broad St.
Wachman Hall 324
tub58431@temple.edu

Alexander Yates

Temple University
1805 N. Broad St.
Wachman Hall 324
yates@temple.edu

Abstract

Supervised sequence-labeling systems in natural language processing often suffer from data sparsity because they use word types as features in their prediction tasks. Consequently, they have difficulty estimating parameters for types which appear in the test set, but seldom (or never) appear in the training set. We demonstrate that distributional representations of word types, trained on unannotated text, can be used to improve performance on rare words. We incorporate aspects of these representations into the feature space of our sequence-labeling systems. In an experiment on a standard chunking dataset, our best technique improves a chunker from 0.76 F1 to 0.86 F1 on chunks beginning with rare words. On the same dataset, it improves our part-of-speech tagger from 74% to 80% accuracy on rare words. Furthermore, our system improves significantly over a baseline system when applied to text from a different domain, and it reduces the sample complexity of sequence labeling.

1 Introduction

Data sparsity and high dimensionality are the twin curses of statistical natural language processing (NLP). In many traditional supervised NLP systems, the feature space includes dimensions for each word type in the data, or perhaps even combinations of word types. Since vocabularies can be extremely large, this leads to an explosion in the number of parameters. To make matters worse, language is Zipf-distributed, so that a large fraction of any training data set will be *hapax legomena*, very many word types will appear only a few times, and many word types will be left out of the training set altogether. As a consequence, for

many word types supervised NLP systems have very few, or even zero, labeled examples from which to estimate parameters.

The negative effects of data sparsity have been well-documented in the NLP literature. The performance of state-of-the-art, supervised NLP systems like part-of-speech (POS) taggers degrades significantly on words that do not appear in the training data, or out-of-vocabulary (OOV) words (Lafferty et al., 2001). Performance also degrades when the domain of the test set differs from the domain of the training set, in part because the test set includes more OOV words and words that appear only a few times in the training set (henceforth, *rare* words) (Blitzer et al., 2006; Daumé III and Marcu, 2006; Chelba and Acero, 2004).

We investigate the use of distributional representations, which model the probability distribution of a word’s context, as techniques for finding *smoothed* representations of word sequences. That is, we use the distributional representations to share information across unannotated examples of the same word type. We then compute features of the distributional representations, and provide them as input to our supervised sequence labelers. Our technique is particularly well-suited to handling data sparsity because it is possible to improve performance on rare words by supplementing the training data with additional unannotated text containing more examples of the rare words. We provide empirical evidence that shows how distributional representations improve sequence-labeling in the face of data sparsity.

Specifically, we investigate empirically the effects of our smoothing techniques on two sequence-labeling tasks, POS tagging and chunking, to answer the following:

1. *What is the effect of smoothing on sequence-labeling accuracy for rare word types?* Our best smoothing technique improves a POS tagger by 11% on OOV words, and a chunker by an impressive 21% on OOV words.

2. *Can smoothing improve adaptability to new domains?* After training our chunker on newswire text, we apply it to biomedical texts. Remarkably, we find that the smoothed chunker achieves a higher F1 on the new domain than the baseline chunker achieves on a test set from the original newswire domain.

3. *How does our smoothing technique affect sample complexity?* We show that smoothing drastically reduces sample complexity: our smoothed chunker requires under 100 labeled samples to reach 85% accuracy, whereas the unsmoothed chunker requires 3500 samples to reach the same level of performance.

The remainder of this paper is organized as follows. Section 2 discusses the smoothing problem for word sequences, and introduces three smoothing techniques. Section 3 presents our empirical study of the effects of smoothing on two sequence-labeling tasks. Section 4 describes related work, and Section 5 concludes and suggests items for future work.

2 Smoothing Natural Language Sequences

To *smooth* a dataset is to find an approximation of it that retains the important patterns of the original data while hiding the noise or other complicating factors. Formally, we define the smoothing task as follows: let $\mathcal{D} = \{(\mathbf{x}, \mathbf{z}) | \mathbf{x} \text{ is a word sequence, } \mathbf{z} \text{ is a label sequence}\}$ be a labeled dataset of word sequences, and let \mathcal{M} be a machine learning algorithm that will learn a function f to predict the correct labels. The smoothing task is to find a function g such that when \mathcal{M} is applied to $D' = \{(g(\mathbf{x}), \mathbf{z}) | (\mathbf{x}, \mathbf{z}) \in D\}$, it produces a function f' that is more accurate than f .

For supervised sequence-labeling problems in NLP, the most important “complicating factor” that we seek to avoid through smoothing is the data sparsity associated with word-based representations. Thus, the task is to find g such that for every word x , $g(x)$ is much less sparse, but still retains the essential features of x that are useful for predicting its label.

As an example, consider the string “Researchers test reformulated gasolines on newer engines.” In a common dataset for NP chunking, the word “reformulated” never appears in the training data, but appears four times in the test set as part of the NP “reformulated gasolines.” Thus, a learning algorithm supplied with word-level features would

have a difficult time determining that “reformulated” is the start of a NP. Character-level features are of little help as well, since the “-ed” suffix is more commonly associated with verb phrases. Finally, context may be of some help, but “test” is ambiguous between a noun and verb, and “gasolines” is only seen once in the training data, so there is no guarantee that context is sufficient to make a correct judgment.

On the other hand, some of the other contexts in which “reformulated” appears in the test set, such as “testing of reformulated gasolines,” provide strong evidence that it can start a NP, since “of” is a highly reliable indicator that a NP is to follow. This example provides the intuition for our approach to smoothing: we seek to share information about the contexts of a word across multiple instances of the word, in order to provide more information about words that are rarely or never seen in training. In particular, we seek to represent each word by a distribution over its contexts, and then provide the learning algorithm with features computed from this distribution. Importantly, we seek distributional representations that will provide features that are common in both training and test data, to avoid data sparsity. In the next three sections, we develop three techniques for smoothing text using distributional representations.

2.1 Multinomial Representation

In its simplest form, the context of a word may be represented as a multinomial distribution over the terms that appear on either side of the word. If \mathcal{V} is the vocabulary, or the set of word types, and \mathbf{X} is a sequence of random variables over \mathcal{V} , the left and right context of $X_i = v$ may each be represented as a probability distribution over \mathcal{V} : $P(X_{i-1} | X_i = v)$ and $P(X_{i+1} | X_i = v)$ respectively.

We learn these distributions from unlabeled texts in two different ways. The first method computes word count vectors for the left and right contexts of each word type in the vocabulary of the training and test texts. We also use a large collection of additional text to determine the vectors. We then normalize each vector to form a probability distribution. The second technique first applies TF-IDF weighting to each vector, where the context words of each word type constitute a document, before applying normalization. This gives greater weight to words with more idiosyncratic distributions and may improve the informativeness of a distributional representation. We refer to these techniques as TF and TF-IDF.

To supply a sequence-labeling algorithm with information from these distributional representations, we compute real-valued features of the context distributions. In particular, for every word x_i in a sequence, we provide the sequence labeler with a set of features of the left and right contexts indexed by $v \in \mathcal{V}$: $F_v^{left}(x_i) = P(X_{i-1} = v|x_i)$ and $F_v^{right}(x_i) = P(X_{i+1} = v|x_i)$. For example, the left context for “reformulated” in our example above would contain a nonzero probability for the word “of.” Using the features $\mathbf{F}(x_i)$, a sequence labeler can learn patterns such as, if x_i has a high probability of following “of,” it is a good candidate for the start of a noun phrase. These features provide smoothing by aggregating information across multiple unannotated examples of the same word.

2.2 LSA Model

One drawback of the multinomial representation is that it does not handle sparsity well enough, because the multinomial distributions themselves are so high-dimensional. For example, the two phrases “red lamp” and “magenta tablecloth” share no words in common. If “magenta” is never observed in training, the fact that “tablecloth” appears in its right context is of no help in connecting it with the phrase “red lamp.” But if we can group similar context words together, putting “lamp” and “tablecloth” into a category for household items, say, then these two adjectives will share that category in their context distributions. Any patterns learned for the more common “red lamp” will then also apply to the less common “magenta tablecloth.” Our second distributional representation aggregates information from multiple context words by grouping together the distributions $P(x_{i-1} = v|x_i = w)$ and $P(x_{i-1} = v'|x_i = w)$ if v and v' appear together with many of the same words w . Aggregating counts in this way smooths our representations even further, by supplying better estimates when the data is too sparse to estimate $P(x_{i-1}|x_i)$ accurately.

Latent Semantic Analysis (LSA) (Deerwester et al., 1990) is a widely-used technique for computing dimensionality-reduced representations from a bag-of-words model. We apply LSA to the set of right context vectors and the set of left context vectors separately, to find compact versions of each vector, where each dimension represents a combination of several context word types. We normalize each vector, and then calculate features as

above. After experimenting with different choices for the number of dimensions to reduce our vectors to, we choose a value of 10 dimensions as the one that maximizes the performance of our supervised sequence labelers on held-out data.

2.3 Latent Variable Language Model Representation

To take smoothing one step further, we present a technique that aggregates context distributions both for similar context words $x_{i-1} = v$ and v' , and for similar words $x_i = w$ and w' . Latent variable language models (LVLMs) can be used to produce just such a distributional representation. We use Hidden Markov Models (HMMs) as the main example in the discussion and as the LVLMs in our experiments, but the smoothing technique can be generalized to other forms of LVLMs, such as factorial HMMs and latent variable maximum entropy models (Ghahramani and Jordan, 1997; Smith and Eisner, 2005).

An HMM is a generative probabilistic model that generates each word x_i in the corpus conditioned on a latent variable Y_i . Each Y_i in the model takes on integral values from 1 to S , and each one is generated by the latent variable for the preceding word, Y_{i-1} . The distribution for a corpus $\mathbf{x} = (x_1, \dots, x_N)$ given a set of state vectors $\mathbf{y} = (y_1, \dots, y_N)$ is given by:

$$P(\mathbf{x}|\mathbf{y}) = \prod_i P(x_i|y_i)P(y_i|y_{i-1})$$

Using Expectation-Maximization (Dempster et al., 1977), it is possible to estimate the distributions for $P(x_i|y_i)$ and $P(y_i|y_{i-1})$ from unlabeled data. We use a trained HMM to determine the optimal sequence of latent states \hat{y}_i using the well-known Viterbi algorithm (Rabiner, 1989). The output of this process is an integer (ranging from 1 to S) for every word x_i in the corpus; we include a new boolean feature for each possible value of y_i in our sequence labelers.

To compare our models, note that in the multinomial representation we directly model the probability that a word v appears before a word w : $P(x_{i-1} = v|x_i = w)$. In our LSA model, we find latent categories of context words z , and model the probability that a category appears before the current word w : $P(x_{i-1} = z|x_i = w)$. The HMM finds (probabilistic) categories Y for both the current word x_i and the context word x_{i-1} , and models the probability that one category follows the

other: $P(Y_i|Y_{i-1})$. Thus the HMM is our most extreme smoothing model, as it aggregates information over the greatest number of examples: for a given consecutive pair of words x_{i-1}, x_i in the test set, it aggregates over all pairs of consecutive words x'_{i-1}, x'_i where x'_{i-1} is similar to x_{i-1} and x'_i is similar to x_i .

3 Experiments

We tested the following hypotheses in our experiments:

1. Smoothing can improve the performance of a supervised sequence labeling system on words that are *rare or nonexistent* in the training data.
2. A supervised sequence labeler achieves greater accuracy on *new domains* with smoothing.
3. A supervised sequence labeler has a better *sample complexity* with smoothing.

3.1 Experimental Setup

We investigate the use of smoothing in two test systems, conditional random field (CRF) models for POS tagging and chunking. To incorporate smoothing into our models, we follow the following general procedure: first, we collect a set of unannotated text from the same domain as the test data set. Second, we train a smoothing model on the text of the training data, the test data, and the additional collection. We then automatically annotate both the training and test data with features calculated from the distributional representation. Finally, we train the CRF model on the annotated training set and apply it to the test set.

We use an open source CRF software package designed by Sunita Sajarwal and William W. Cohen to implement our CRF models.¹ We use a set of boolean features listed in Table 1.

Our baseline CRF system for POS tagging follows the model described by Lafferty *et al.* (2001). We include transition features between pairs of consecutive tag variables, features between tag variables and words, and a set of orthographic features that Lafferty *et al.* found helpful for performance on OOV words. Our smoothed models add features computed from the distributional representations, as discussed above.

Our chunker follows the system described by Sha and Pereira (2003). In addition to the transition, word-level, and orthographic features, we include features relating automatically-generated POS tags and the chunk labels. Unlike Sha and

CRF Feature Set	
Transition	$z_i=z$ $z_i=z$ and $z_{i-1}=z'$
Word	$x_i=w$ and $z_i=z$
POS	$t_i=t$ and $z_i=z$
Orthography	for every $s \in \{-ing, -ogy, -ed, -s, -ly, -ion, -tion, -ity\}$, suffix(x_i) = s and $z_i=z$ x_i is capitalized and $z_i = z$ x_i has a digit and $z_i = z$
TF, TF-IDF, and LSA features	for every context type v , $F_v^{left}(x_i)$ and $F_v^{right}(x_i)$
HMM features	$y_i=y$ and $z_i = z$

Table 1: **Features used in our CRF systems.** z_i variables represent labels to be predicted, t_i represent tags (for the chunker), and x_i represent word tokens. All features are boolean except for the TF, TF-IDF, and LSA features.

Pereira, we exclude features relating consecutive pairs of words and a chunk label, or features relating consecutive tag labels and a chunk label, in order to expedite our experiments. We found that including such features does improve chunking F1 by approximately 2%, but it also significantly slows down CRF training.

3.2 Rare Word Accuracy

For these experiments, we use the Wall Street Journal portion of the Penn Treebank (Marcus *et al.*, 1993). Following the CoNLL shared task from 2000, we use sections 15-18 of the Penn Treebank for our labeled training data for the supervised sequence labeler in all experiments (Tjong *et al.*, 2000). For the tagging experiments, we train and test using the gold standard POS tags contained in the Penn Treebank. For the chunking experiments, we train and test with POS tags that are automatically generated by a standard tagger (Brill, 1994). We tested the accuracy of our models for chunking and POS tagging on section 20 of the Penn Treebank, which corresponds to the test set from the CoNLL 2000 task.

Our distributional representations are trained on sections 2-22 of the Penn Treebank. Because we include the text from the train and test sets in our training data for the distributional representations, we do not need to worry about smoothing them — when they are decoded on the test set, they

¹Available from <http://sourceforge.net/projects/crf/>

Freq:	0	1	2	0-2	all
#Samples	438	508	588	1534	46661
Baseline	.62	.77	.81	.74	.93
TF	.76	.72	.77	.75	.92
TF-IDF	.82	.75	.76	.78	.94
LSA	.78	.80	.77	.78	.94
HMM	.73	.81	.86	.80	.94

Table 2: POS tagging accuracy: our HMM-smoothed tagger outperforms the baseline tagger by 6% on rare words. Differences between the baseline and the HMM are statistically significant at $p < 0.01$ for the OOV, 0-2, and all cases using the two-tailed Chi-squared test with 1 degree of freedom.

will not encounter any previously unseen words. However, to speed up training during our experiments and, in some cases, to avoid running out of memory, we replaced words appearing twice or fewer times in the data with the special symbol *UNKNOWN*. In addition, all numbers were replaced with another special symbol. For the LSA model, we had to use a more drastic cutoff to fit the singular value decomposition computation into memory: we replaced words appearing 10 times or fewer with the *UNKNOWN* symbol. We initialize our HMMs randomly. We run EM ten times and take the model with the best cross-entropy on a held-out set. After experimenting with different variations of HMM models, we settled on a model with 80 latent states as a good compromise between accuracy and efficiency.

For our POS tagging experiments, we measured the accuracy of the tagger on “rare” words, or words that appear at most twice in the training data. For our chunking experiments, we focus on chunks that begin with rare words, as we found that those were the most difficult for the chunker to identify correctly. So we define “rare” chunks as those that begin with words appearing at most twice in training data. To ensure that our smoothing models have enough training data for our test set, we further narrow our focus to those words that appear rarely in the labeled training data, but appear at least ten times in sections 2-22. Tables 2 and 3 show the accuracy of our smoothed models and the baseline model on tagging and chunking, respectively. The line for “all” in both tables indicates results on the complete test set.

Both our baseline tagger and chunker achieve respectable results on their respective tasks for all words, and the results were good enough for

Freq:	0	1	2	0-2	all
#Samples	133	199	231	563	21900
Baseline	.69	.75	.81	.76	.90
TF	.70	.82	.79	.77	.89
TF-IDF	.77	.77	.80	.78	.90
LSA	.84	.82	.83	.84	.90
HMM	.90	.85	.85	.86	.93

Table 3: Chunking F1: our HMM-smoothed chunker outperforms the baseline CRF chunker by 0.21 on chunks that begin with OOV words, and 0.10 on chunks that begin with rare words.

us to be satisfied that performance on rare words closely follows how a state-of-the-art supervised sequence-labeler behaves. The chunker’s accuracy is roughly in the middle of the range of results for the original CoNLL 2000 shared task (Tjong et al., 2000). While several systems have achieved slightly higher accuracy on supervised POS tagging, they are usually trained on larger training sets.

As expected, the drop-off in the baseline system’s performance from all words to rare words is impressive for both tasks. Comparing performance on all terms and OOV terms, the baseline tagger’s accuracy drops by 0.31, and the baseline chunker’s F1 drops by 0.21. Comparing performance on all terms and rare terms, the drop is less severe but still dramatic: 0.19 for tagging and 0.15 for chunking.

Our hypothesis that smoothing would improve performance on rare terms is validated by these experiments. In fact, the more aggregation a smoothing model performs, the better it appears to be at smoothing. The HMM-smoothed system outperforms all other systems in all categories except tagging on OOV words, where TF-IDF performs best. And in most cases, the clear trend is for HMM smoothing to outperform LSA, which in turn outperforms TF and TF-IDF. HMM tagging performance on OOV terms improves by 11%, and chunking performance by 21%. Tagging performance on all of the rare terms improves by 6%, and chunking by 10%. In chunking, there is a clear trend toward larger increases in performance as words become rarer in the labeled data set, from a 0.02 improvement on words of frequency 2, to an improvement of 0.21 on OOV words.

Because the test data for this experiment is drawn from the same domain (newswire) as the

training data, the rare terms make up a relatively small portion of the overall dataset (approximately 4% of both the tagged words and the chunks). Still, the increased performance by the HMM-smoothed model on the rare-word subset contributes in part to an increase in performance on the overall dataset of 1% for tagging and 3% for chunking. In our next experiment, we consider a common scenario where rare terms make up a much larger fraction of the test data.

3.3 Domain Adaptation

For our experiment on domain adaptation, we focus on NP chunking and POS tagging, and we use the labeled training data from the CoNLL 2000 shared task as before. For NP chunking, we use 198 sentences from the biochemistry domain in the Open American National Corpus (OANC) (Reppen et al., 2005) as our test set. We manually tagged the test set with POS tags and NP chunk boundaries. The test set contains 5330 words and a total of 1258 NP chunks. We used sections 15-18 of the Penn Treebank as our labeled training set, including the gold standard POS tags. We use our best-performing smoothing model, the HMM, and train it on sections 13 through 19 of the Penn Treebank, plus the written portion of the OANC that contains journal articles from biochemistry (40,727 sentences). We focus on chunks that begin with words appearing 0-2 times in the labeled training data, and appearing at least ten times in the HMM’s training data. Table 4 contains our results. For our POS tagging experiments, we use 561 MEDLINE sentences (9576 words) from the Penn BioIE project (PennBioIE, 2005), a test set previously used by Blitzer *et al.* (2006). We use the same experimental setup as Blitzer *et al.*: 40,000 manually tagged sentences from the Penn Treebank for our labeled training data, and all of the unlabeled text from the Penn Treebank plus their MEDLINE corpus of 71,306 sentences to train our HMM. We report on tagging accuracy for all words and OOV words in Table 5. This table also includes results for two previous systems as reported by Blitzer *et al.* (2006): the semi-supervised Alternating Structural Optimization (ASO) technique and the Structural Correspondence Learning (SCL) technique for domain adaptation.

Note that this test set for NP chunking contains a much higher proportion of rare and OOV words: 23% of chunks begin with an OOV word, and 29% begin with a rare word, as compared with

Freq.	#	Baseline			HMM		
		R	P	F1	R	P	F1
0	284	.74	.70	.72	.80	.89	.84
1	39	.85	.87	.86	.92	.88	.90
2	39	.79	.86	.83	.92	.90	.91
0-2	362	.75	.73	.74	.82	.89	.85
all	1258	.86	.87	.86	.91	.90	.91

Table 4: **On biochemistry journal data from the OANC, our HMM-smoothed NP chunker outperforms the baseline CRF chunker by 0.12 (F1) on chunks that begin with OOV words, and by 0.05 (F1) on all chunks.** Results in bold are statistically significantly different from the baseline results at $p < 0.05$ using the two-tailed Fisher’s exact test. We did not perform significance tests for F1.

Model	All	Unknown
	words	words
Baseline	88.3	67.3
ASO	88.4	70.9
SCL	88.9	72.0
HMM	90.5	75.2

Table 5: **On biomedical data from the Penn BioIE project, our HMM-smoothed tagger outperforms the SCL tagger by 3% (accuracy) on OOV words, and by 1.6% (accuracy) on all words.** Differences between the smoothed tagger and the SCL tagger are significant at $p < .001$ for all words and for OOV words, using the Chi-squared test with 1 degree of freedom.

1% and 4%, respectively, for NP chunks in the test set from the original domain. The test set for tagging also contains a much higher proportion: 23% OOV words, as compared with 1% in the original domain. Because of the increase in the number of rare words, the baseline chunker’s overall performance drops by 4% compared with performance on WSJ data, and the baseline tagger’s overall performance drops by 5% in the new domain.

The performance improvements for both the smoothed NP chunker and tagger are again impressive: there is a 12% improvement on OOV words, and a 10% overall improvement on rare words for chunking; the tagger shows an 8% improvement on OOV words compared to our baseline and a 3% improvement on OOV words compared to the SCL model. The resulting performance of the smoothed NP chunker is almost identical to its performance on the WSJ data. Through smoothing, the chunker not only improves by 5%

in F1 over the baseline system on all words, it in fact outperforms our baseline NP chunker on the WSJ data. 60% of this improvement comes from improved accuracy on rare words.

The performance of our HMM-smoothed chunker caused us to wonder how well the chunker could work without some of its other features. We removed all tag features and all features for word types that appear fewer than 20 times in training. This chunker achieves 0.91 F1 on OANC data, and 0.93 F1 on WSJ data, outperforming the baseline system in both cases. It has only 20% as many features as the baseline chunker, greatly improving its training time. Thus our smoothing features are more valuable to the chunker than features from POS tags and features for all but the most common words. Our results point to the exciting possibility that with smoothing, we may be able to train a sequence-labeling system on a small labeled sample, and have it apply generally to other domains. Exactly what size training set we need is a question that we address next.

3.4 Sample Complexity

Our complete system consists of two learned components, a supervised CRF system and an unsupervised smoothing model. We measure the sample complexity of each component separately. To measure the sample complexity of the supervised CRF, we use the same experimental setup as in the chunking experiment on WSJ text, but we vary the amount of labeled data available to the CRF. We take ten random samples of a fixed size from the labeled training set, train a chunking model on each subset, and graph the F1 on the labeled test set, averaged over the ten runs, in Figure 1. To measure the sample complexity of our HMM with respect to unlabeled text, we use the full labeled training set and vary the amount of unlabeled text available to the HMM. At minimum, we use the text available in the labeled training and test sets, and then add random subsets of the Penn Treebank, sections 2-22. For each subset size, we take ten random samples of the unlabeled text, train an HMM and then a chunking model, and graph the F1 on the labeled test set averaged over the ten runs in Figure 2.

The results from our labeled sample complexity experiment indicate that sample complexity is drastically reduced by HMM smoothing. On rare chunks, the smoothed system reaches 0.78 F1 using only 87 labeled training sentences, a level that the baseline system never reaches, even with 6933

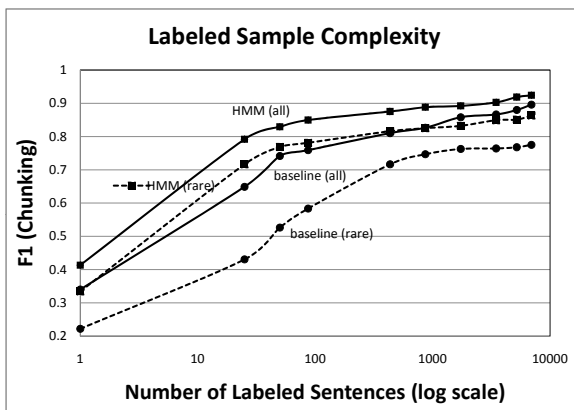


Figure 1: The smoothed NP chunker requires less than 10% of the samples needed by the baseline chunker to achieve .83 F1, and the same for .88 F1.

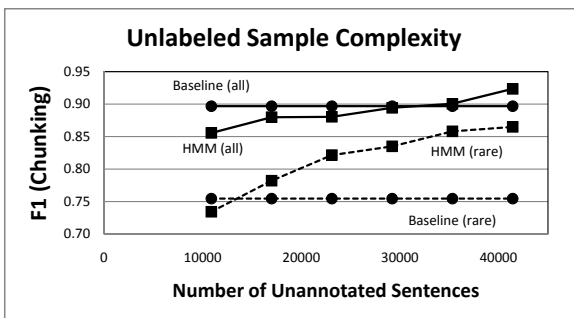


Figure 2: By leveraging plentiful unannotated text, the smoothed chunker soon outperforms the baseline.

labeled sentences. On the overall data set, the smoothed system reaches 0.83 F1 with 50 labeled sentences, which the baseline does not reach until it has 867 labeled sentences. With 434 labeled sentences, the smoothed system reaches 0.88 F1, which the baseline system does not reach until it has 5200 labeled samples.

Our unlabeled sample complexity results show that even with access to a small amount of unlabeled text, 6000 sentences more than what appears in the training and test sets, smoothing using the HMM yields 0.78 F1 on rare chunks. However, the smoothed system requires 25,000 more sentences before it outperforms the baseline system on all chunks. No peak in performance is reached, so further improvements are possible with more unlabeled data. Thus smoothing is optimizing performance for the case where unlabeled data is plentiful and labeled data is scarce, as we would hope.

4 Related Work

To our knowledge, only one previous system — the REALM system for sparse information extrac-

tion — has used HMMs as a feature representation for other applications. REALM uses an HMM trained on a large corpus to help determine whether the arguments of a candidate relation are of the appropriate type (Downey et al., 2007). We extend and generalize this smoothing technique and apply it to common NLP applications involving supervised sequence-labeling, and we provide an in-depth empirical analysis of its performance.

Several researchers have previously studied methods for using unlabeled data for tagging and chunking, either alone or as a supplement to labeled data. Ando and Zhang develop a semi-supervised chunker that outperforms purely supervised approaches on the CoNLL 2000 dataset (Ando and Zhang, 2005). Recent projects in semi-supervised (Toutanova and Johnson, 2007) and unsupervised (Biemann et al., 2007; Smith and Eisner, 2005) tagging also show significant progress. Unlike these systems, our efforts are aimed at using unlabeled data to find distributional representations that work well on rare terms, making the supervised systems more applicable to other domains and decreasing their sample complexity.

HMMs have been used many times for POS tagging and chunking, in supervised, semi-supervised, and in unsupervised settings (Banko and Moore, 2004; Goldwater and Griffiths, 2007; Johnson, 2007; Zhou, 2004). We take a novel perspective on the use of HMMs by using them to compute features of each token in the data that represent the distribution over that token’s contexts. Our technique lets the HMM find parameters that maximize cross-entropy, and then uses labeled data to learn the best mapping from the HMM categories to the POS categories.

Smoothing in NLP usually refers to the problem of smoothing n -gram models. Sophisticated smoothing techniques like modified Kneser-Ney and Katz smoothing (Chen and Goodman, 1996) smooth together the predictions of unigram, bigram, trigram, and potentially higher n -gram sequences to obtain accurate probability estimates in the face of data sparsity. Our task differs in that we are primarily concerned with the case where even the unigram model (single word) is rarely or never observed in the labeled training data.

Sparsity for low-order contexts has recently spurred interest in using latent variables to represent distributions over contexts in language models. While n -gram models have traditionally dominated in language modeling, two recent efforts de-

velop latent-variable probabilistic models that rival and even surpass n -gram models in accuracy (Blitzer et al., 2005; Mnih and Hinton, 2007). Several authors investigate neural network models that learn not just one latent state, but rather a vector of latent variables, to represent each word in a language model (Bengio et al., 2003; Emami et al., 2003; Morin and Bengio, 2005).

One of the benefits of our smoothing technique is that it allows for domain adaptation, a topic that has received a great deal of attention from the NLP community recently. Unlike our technique, in most cases researchers have focused on the scenario where labeled training data is available in both the source and the target domain (*e.g.*, (Daumé III, 2007; Chelba and Acero, 2004; Daumé III and Marcu, 2006)). Our technique uses unlabeled training data from the target domain, and is thus applicable more generally, including in web processing, where the domain and vocabulary is highly variable, and it is extremely difficult to obtain labeled data that is representative of the test distribution. When labeled target-domain data is available, instance weighting and similar techniques can be used in combination with our smoothing technique to improve our results further, although this has not yet been demonstrated empirically. HMM-smoothing improves on the most closely related work, the Structural Correspondence Learning technique for domain adaptation (Blitzer et al., 2006), in experiments.

5 Conclusion and Future Work

Our study of smoothing techniques demonstrates that by aggregating information across many unannotated examples, it is possible to find accurate distributional representations that can provide highly informative features to supervised sequence labelers. These features help improve sequence labeling performance on rare word types, on domains that differ from the training set, and on smaller training sets.

Further experiments are of course necessary to investigate distributional representations as smoothing techniques. One particularly promising area for further study is the combination of smoothing and instance weighting techniques for domain adaptation. Whether the current techniques are applicable to structured prediction tasks, like parsing and relation extraction, also deserves future attention.

References

- Rie Kubota Ando and Tong Zhang. 2005. A high-performance semi-supervised learning method for text chunking. In *ACL*.
- Michele Banko and Robert C. Moore. 2004. Part of speech tagging in context. In *COLING*.
- Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155.
- C. Biemann, C. Giuliano, and A. Gliozzo. 2007. Unsupervised pos tagging supporting supervised methods. *Proceeding of RANLP-07*.
- J. Blitzer, A. Globerson, and F. Pereira. 2005. Distributed latent variable models of lexical cooccurrences. In *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics*.
- John Blitzer, Ryan McDonald, and Fernando Pereira. 2006. Domain adaptation with structural correspondence learning. In *EMNLP*.
- E. Brill. 1994. Some Advances in Rule-Based Part of Speech Tagging. In *AAAI*, pages 722–727, Seattle, Washington.
- Ciprian Chelba and Alex Acero. 2004. Adaptation of maximum entropy classifier: Little data can help a lot. In *EMNLP*.
- Stanley F. Chen and Joshua Goodman. 1996. An empirical study of smoothing techniques for language modeling. In *Proceedings of the 34th annual meeting on Association for Computational Linguistics*, pages 310–318, Morristown, NJ, USA. Association for Computational Linguistics.
- Hal Daumé III and Daniel Marcu. 2006. Domain adaptation for statistical classifiers. *Journal of Artificial Intelligence Research*, 26.
- Hal Daumé III. 2007. Frustratingly easy domain adaptation. In *ACL*.
- S. C. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American Society of Information Science*, 41(6):391–407.
- Arthur Dempster, Nan Laird, and Donald Rubin. 1977. Likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38.
- D. Downey, S. Schoenmackers, and O. Etzioni. 2007. Sparse information extraction: Unsupervised language models to the rescue. In *ACL*.
- A. Emami, P. Xu, and F. Jelinek. 2003. Using a connectionist model in a syntactical based language model. In *Proceedings of the International Conference on Spoken Language Processing*, pages 372–375.
- Zoubin Ghahramani and Michael I. Jordan. 1997. Factorial hidden markov models. *Machine Learning*, 29(2-3):245–273.
- Sharon Goldwater and Thomas L. Griffiths. 2007. A fully bayesian approach to unsupervised part-of-speech tagging. In *ACL*.
- Mark Johnson. 2007. Why doesn't EM find good HMM POS-taggers. In *EMNLP*.
- J. Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the International Conference on Machine Learning*.
- Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, 19(2):313–330.
- Andriy Mnih and Geoffrey Hinton. 2007. Three new graphical models for statistical language modelling. In *Proceedings of the 24th International Conference on Machine Learning*, pages 641–648, New York, NY, USA. ACM.
- F. Morin and Y. Bengio. 2005. Hierarchical probabilistic neural network language model. In *Proceedings of the International Workshop on Artificial Intelligence and Statistics*, pages 246–252.
- PennBioIE. 2005. Mining the bibliome project. <http://bioie ldc.upenn.edu/>.
- Lawrence R. Rabiner. 1989. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–285.
- Randi Reppen, Nancy Ide, and Keith Suderman. 2005. American national corpus (ANC) second release. Linguistic Data Consortium.
- F. Sha and Fernando Pereira. 2003. Shallow parsing with conditional random fields. In *Proceedings of Human Language Technology - NAACL*.
- Noah A. Smith and Jason Eisner. 2005. Contrastive estimation: Training log-linear models on unlabeled data. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 354–362, Ann Arbor, Michigan, June.
- Erik F. Tjong, Kim Sang, and Sabine Buchholz. 2000. Introduction to the CoNLL-2000 shared task: Chunking. In *Proceedings of the 4th Conference on Computational Natural Language Learning*, pages 127–132.
- Kristina Toutanova and Mark Johnson. 2007. A bayesian LDA-based model for semi-supervised part-of-speech tagging. In *NIPS*.
- GuoDong Zhou. 2004. Discriminative hidden Markov modeling with long state dependence using a kNN ensemble. In *COLING*.