Name Tagging with Word Clusters and Discriminative Training

Scott Miller, Jethran Guinness, Alex Zamanian
BBN Technologies
10 Moulton Street
Cambridge, MA 02138
szmiller@bbn.com

Abstract

We present a technique for augmenting annotated training data with hierarchical word clusters that are automatically derived from a large unannotated corpus. Cluster membership is encoded in features that are incorporated in a discriminatively trained tagging model. Active learning is used to select training examples. We evaluate the technique for named-entity tagging. Compared with a state-of-the-art HMM-based name finder, the presented technique requires only 13% as much annotated data to achieve the same level of performance. Given a large annotated training set of 1,000,000 words, the technique achieves a 25% reduction in error over the state-of-the-art HMM trained on the same material.

1 Introduction

At a recent meeting, we presented name-tagging technology to a potential user. The technology had performed well in formal evaluations, had been applied successfully by several research groups, and required only annotated training examples to configure for new name classes. Nevertheless, it did not meet the user's needs.

To achieve reasonable performance, the HMM-based technology we presented required roughly 150,000 words of annotated examples, and over a million words to achieve peak accuracy. Given a typical annotation rate of 5,000 words per hour, we estimated that setting up a name finder for a new problem would take four person days of annotation work – a period we considered reasonable. However, this user's problems were too dynamic for that much setup time. To be useful, the system would have to be trainable in minutes or hours, not days or weeks.

We left the meeting thinking about ways to reduce training requirements to no more than a few hours. It seemed that three existing ideas could be combined in a way that might reduce training requirements sufficiently to achieve the objective.

First were techniques for producing word clusters from large unannotated corpora (Brown et al., 1990; Pereira et al., 1993; Lee and Pereira, 1999). The resulting clusters appeared to contain a great deal of implicit semantic information. This implicit information, we believed, could serve to augment a small amount of annotated data. Particularly promising were techniques for producing hierarchical clusters at various scales, from small and highly specific to large and more general. To benefit from such information, however, we would need an automatic learning mechanism that could effectively exploit it.

Fortunately, a second line of recent research provided a potential solution. Recent work in discriminative methods (Lafferty et al., 2001; Sha and Pereira, 2003, Collins 2002) suggested a framework for exploiting large numbers of arbitrary input features. These methods seemed to have exactly the right characteristics for incorporating the statistically-correlated hierarchical word clusters we wished to exploit.

Combining these two methods, we suspected, would be sufficient to drastically reduce the number of annotated examples required. However, we also hoped that a third technique, active learning (Cohn et al., 1996;

McCallum and Nigam, 1998), would be particularly effective when used in conjunction with hierarchical word clusters. Specifically, active learning attempts to select examples for annotation by estimating the system's certainty about the answer, requesting a human judgment only for those cases where it is most uncertain. Unfortunately, the issue often comes down to whether a specific word has previously been observed in training: if the system has seen the word, it is certain, if not, it is uncertain. Word clusters at various scales, we hoped, would permit more subtle distinctions to influence the system's certainty, increasing the method's effectiveness earlier in the process when fewer training examples have been annotated.

2 Word Clustering

We view clustering here as a method for estimating the probabilities of low frequency events, particularly events that are likely to go unobserved in a small annotated training corpus. For example, a clustering mechanism may choose to place *AT&T* in the same cluster as other company names based on contextual similarity. Then, even if the word *AT&T* was not previously annotated as a company, it may nonetheless be possible to infer that *AT&T* indeed is a company because it occupies a cluster that is populated mostly by other company names. Likewise, cluster membership can be used to exploit information from neighboring words. For example, if the word *reported* has previously been observed to follow person names, but the word *announced* has not yet been seen, it may be possible to guess that the word preceding *announced* is a person based on the fact that *reported* and *announced* occupy the same cluster.

A practical obstacle to using clusters for this purpose is selecting an appropriate level of granularity: too small, and the clusters provide insufficient generalization; too large, and they inappropriately overgeneralize. Hierarchical clusters provide one way around the problem by avoiding commitment to any particular granularity in advance.

However, the dominant trend during the past decade toward generative models has made integration of such hierarchical clusters difficult. Because the nested clusters surrounding each word are highly correlated, it is unreasonable to treat them as independent. Unfortunately, any treatment in a generative framework other than independent requires considerable ingenuity.

Interestingly, before generative models began to dominate parsing, the Spatter parser (Magerman, 1995) achieved extremely promising results using a non-generative statistical model. Of particular interest is the fact that Spatter used hierarchical word clusters for estimating its lexical attachment probabilities. However, the statistical decision trees underlying Spatter's probability model never gained widespread acceptance, and indeed, our own limited experience with them yielded mixed results.

In the past few years, researchers have begun to view generative models as instances of a broader class of linear (or log-linear) models, and have introduced discriminative methods (e.g. conditional random fields) to estimate the model parameters. These estimation methods do not impose the same strict independence conditions as generative models.

Armed with modern discriminative training methods, it seemed reasonable to us to revisit hierarchical clustering.

Specifically, we picked up where Spatter left off, with the clustering algorithm of (Brown et al., 1990). We implemented this algorithm twice as part of our work. The first implementation derived directly from the description given in the Brown paper. Then, in the hope of achieving greater efficiency, we reverse-engineered the clustering software in Spatter. While the mathematical details differ slightly between the two algorithms, both aim to cluster together words so as to minimize the bigram language-model perplexity of the unsupervised corpus. In practice, we observed no significant differences in accuracy when using one or the other in our experiments. All experimental results given in this paper are with the Spatter clustering algorithm.

The result of running the clustering algorithm is a binary tree, where each word occupies a single leaf node, and where each leaf node contains a single word. The root node defines a cluster containing the entire vocabulary. Interior nodes represent intermediate size clusters containing all of the words that they dominate. Thus, nodes higher in the tree correspond to larger word clusters, while lower nodes correspond to smaller clusters.

A particular word can be assigned a binary string by following the traversal path from the root to its leaf, assigning a 0 for each left branch, and a 1 for each right branch. The following are example bit strings from the Spatter clustering algorithm:

lawyer	1000001101000
newspaperman	100000110100100
stewardess	100000110100101
toxicologist	10000011010011
slang	1000001101010
babysitter	100000110101100
conspirator	1000001101011010
womanizer	1000001101011011
mailman	10000011010111
salesman	100000110110000
bookkeeper	1000001101100010
troubleshooter	10000011011000110
bouncer	10000011011000111
technician	1000001101100100
janitor	1000001101100101
saleswoman	1000001101100110
...	
Nike	1011011100100101011100
Maytag	10110111001001010111010
Generali	10110111001001010111011
Gap	1011011100100101011110
Harley-Davidson	10110111001001010111110
Enfield	101101110010010101111110
genus	101101110010010101111111
Microsoft	10110111001001011000
Ventritex	10110111001001010010
Tractebel	101101110010010100110
Synopsys	10110111001001011000111
WordPerfect	101101110010010110100 0
....	
John	101110010000000000
Consuelo	101110010000000001
Jeffrey	101110010000000010
Kenneth	10111001000000001100
Phillip	101110010000000011010
WILLIAM	101110010000000011011
Timothy	10111001000000001110
Terrence	101110010000000011110
Jerald	101110010000000011111
Harold	101110010000000100
Frederic	101110010000000101
Wendell	10111001000000011

Table 1: Sample bit strings

3 Discriminative Name Tagger

To implement discriminative training, we followed the *averaged perceptron* approach of (Collins, 2002). Our decision was based on three criteria. First, the method performed nearly as well as the currently best global discriminative model (Sha and Pereira, 2003), as evaluated on one of the few tasks for which there are any published results (noun phrase chunking). Second, convergence rates appeared favorable, which would facilitate multiple experiments. Finally, and most important, the method appeared far simpler to implement than any of the alternatives.

We implemented the averaged perceptron training algorithm exactly as described by Collins. However, we did not implement cross-validation to determine when to stop training. Instead, we simply iterated for 5 epochs in all cases, regardless of the training set size or number of features used. Furthermore, we did not implement features that occurred in no training instances, as was done in (Sha and Pereira, 2003). We suspect that these simplifications may have cost several tenths of a point in performance.

A set of 16 tags was used to tag 8 name classes (the seven MUC classes plus the additional null class). Two tags were required per class to account for adjacent elements of the same type. For example, the string *Betty Mary and Bobby Lou* would be tagged as PERSON-START PERSON-START NULL-START PERSON-START PERSON-CONTINUE.

Our model uses a total of 19 classes of features. The first seven of these correspond closely to features used in a typical HMM name tagger. The remaining twelve encode cluster membership. Clusters of various granularity are specified by *prefixes* of the bit strings. Short prefixes specify short paths from the root node and therefore large clusters. Long prefixes specify long paths and small clusters. We used 4 different prefix lengths: 8 bit, 12 bit, 16 bit, and 20 bit. Thus, the clusters decrease in size by about a factor of 16 at each level. The complete set of features is given in Table 2.

1. Tag + PrevTag
2. Tag + CurWord
3. Tag + CapAndNumFeatureOfCurWord
4. ReducedTag + CurWord
   //collapse start and continue tags
5. Tag + PrevWord
6. Tag + NextWord
7. Tag + DownCaseCurWord
8. Tag + Pref8ofCurrWord
9. Tag + Pref12ofCurrWord
10. Tag + Pref16ofCurrWord
11. Tag + Pref20ofCurrWord
12. Tag + Pref8ofPrevWord
13. Tag + Pref12ofPrevWord
14. Tag + Pref16ofPrevWord
15. Tag + Pref20ofPrevWord
16. Tag + Pref8ofNextWord
17. Tag + Pref12ofNextWord
18. Tag + Pref16ofNextWord
19. Tag + Pref20ofNextWord

Table 2: Feature Set

4 Active Learning

We used only a rudimentary confidence measure to

perform active learning, introducing no additional features beyond those used in training and decoding. The confidence score we assign to a sentence is just the un-normalized difference in perceptron scores between the highest scoring theory and the second highest scoring alternative. To apply active learning, we simply

1. Compute the confidence score for every sentence in the available pool.

2. Sort the results into ascending order.

3. Request annotations for a block of sentences beginning at the top of the list.

To compute the confidence scores efficiently, we use a combination of the forward Viterbi and backward Viterbi scores at each word. We define the confidence at a word to be the difference between the summed forward and backward scores of the best and second best tags for that word. The confidence for the entire sentence is then just the minimum of the scores at each word position.

5 Experimental Results

We performed our experiments using the seven MUC-6 name categories: person, organization, location, date, time, percent, and monetary amount. For annotated data, we used text from Sections 02-23 of the Wall Street Journal Treebank corpus that had previously been annotated with the MUC name classes. Sections 02-21 were used as training material, and Section 23 was used as test (note that the syntactic trees were not used in any way). Scoring was performed using the MUC scorer.

For unsupervised clustering data, we used the Wall Street Journal subset of the Continuous Speech Recognition (CSR-III) collection (LDC catalog # LDC95T6). This portion of the collection contains approximately 100 million words.

Active learning experiments were performed by permitting the system to choose examples from among the pool of annotated data, rather than presenting the examples in their natural chronological order. This approach, previously used in [Boschee et al, 2002], permits simulation of human-in-the-loop experiments that are inexpensive to run and repeatable because they don't actually involve a human annotator. However, because the pool of pre-annotated examples is limited, the results are most meaningful for small training sets. Once the system has selected the most useful examples from the pool, it is forced to choose among the remainder that it previously rejected as less useful. At the extreme where all available examples are used, our experimental framework prevents active learning from exhibiting any benefit whatsoever since the system is left no choice in selecting examples.

Before considering the impact of word clustering on system performance, we first evaluate the discriminative tagger relative to the baseline HMM. For this experiment, we used all of the features described in Section 3 except word cluster features. The remaining features encode essentially the same information used in the HMM, although in a slightly different form. Results are shown in Figure 1. For very small and very large training sets, the systems perform about the same. Between these extremes, the discriminative tagger exhibits somewhat, though not distressingly, worse performance. We conjecture that lack of smoothing in the discriminative tagger may account for the difference.

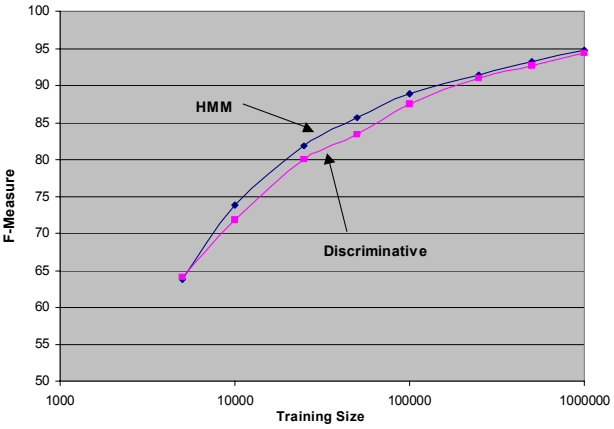

Figure 1: Impact of Discriminative Training

Second, we consider the impact of word clusters. Figure 2 compares performance of the discriminative tagger, now with cluster features included, to the baseline HMM. Immediately, with only 5,000 words of training, the discriminative model significantly outperforms the HMM. With 50,000 words of training, performance for the discriminative model exceeds 90F, a level not reached by the HMM until it has observed 150,000 words of training. Somewhat surprisingly, the clusters continue to provide some benefit even with 1,000,000 words of training. At this operating point, the discriminative tagger achieves an F-score of 96.08 compared to 94.72 for the HMM, a 25% reduction in error.

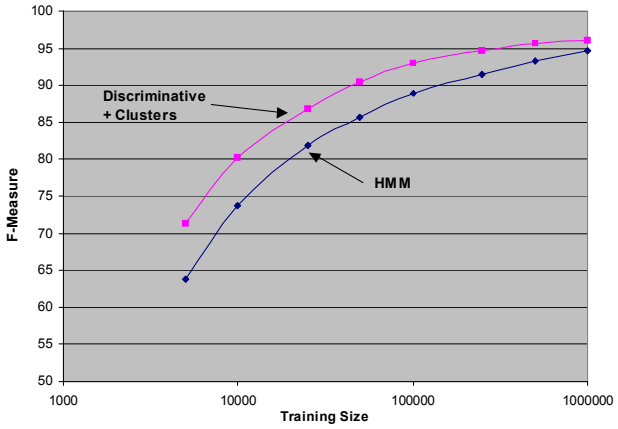

Figure 2: Impact of Word Clustering

Third, we consider the impact of active learning. Figure 3 shows (a) discriminative tagger performance without cluster features, (b) the same tagger using active learning, (c) the discriminative tagger with cluster features, and (d) the discriminative tagger with cluster features using active learning. Both with and without clusters, active learning exhibits a noticeable increase in learning rates. However, the increase in learning rate is significantly more pronounced when cluster features are introduced. We attribute this increase to better confidence measures provided by word clusters – the system is no longer restricted to whether or not it knows a word; it now can know something about the clusters to which a word belongs, even if it does not know the word.

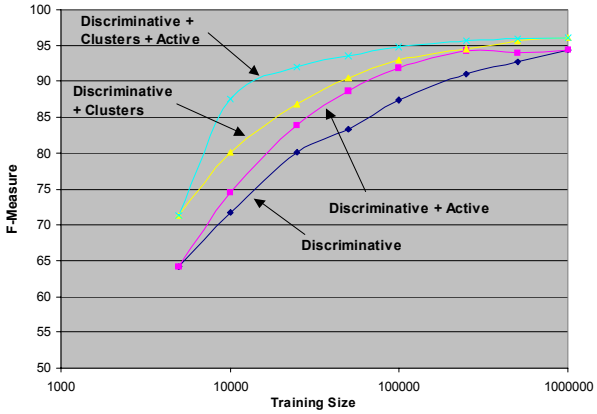

Figure 3: Impact of Active Learning

Finally, Figure 4 shows the impact of consolidating the gains from both cluster features and active learning compared to the baseline HMM. This final combination achieves an F-score of 90 with less than 20,000 words of training – a quantity that can be annotated in about 4 person hours – compared to 150,000 words for the HMM – a quantity requiring nearly 4 person days to annotate. At 1,000,000 word of training, the final combination continues to exhibit a 25% reduction in error over the baseline system (because of limitations in the experimental framework discussed earlier, active learning can provide no additional gain at this operating point).

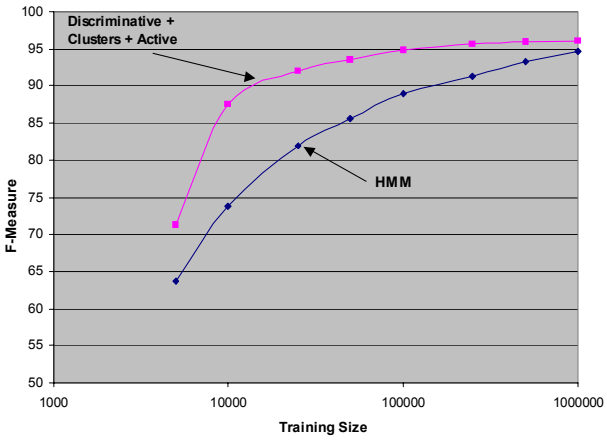

Figure 4: Cumulative Impact of Discriminative Training, Clustering, and Active Learning

6 Discussion

The work presented here extends a substantial body of previous work (Blum and Mitchell, 1998; Riloff and Jones, 1999; Lin et al., 2003; Boschee et al, 2002; Collins and Singer, 1999; Yarowsky, 1995) that all focuses on reducing annotation requirements through a combination of (a) seed examples, (b) large un-annotated corpora, and (c) training example selection. Moreover, our work is based largely on existing techniques for word clustering (Brown et al., 1990), discriminative training (Collins 2002), and active learning.

The synthesis of these techniques, nevertheless, proved highly effective in achieving our primary objective of reducing the need for annotated data.

Much work remains to be done. In an effort to move rapidly toward our primary objective, we investigated only one type of discriminative training (averaged perceptron), only one type of clustering (bigram mutual information), and only one simple confidence measure for active learning. It seems likely that some additional gains could be realized by alternative discriminative methods (e.g. conditional random fields estimated with conjugate-gradient training). Similarly, alternative clustering techniques, perhaps based on different contextual features or different distance measures,

could further improve performance.

On the application side, it would be interesting to apply the technique to other language problems. Applying it to parsing would yield a rare sense of closure, knitting together the word clustering of Magerman's (1995) Spatter parser – arguably the first successful broad-coverage statistical parser – with structural elements of the now-dominant Collins (1997) style parsers.

Because our combined method promises to require substantially less training data, it may also prove useful for so-called low-density languages, where limited resources – and even more limited numbers of native speakers – are available.

For the moment, we find the initial results encouraging. We achieved a 25% reduction in error on a standard named-entity problem, compared to a state-of-the-art HMM. Our main objective, though, was not reducing error rates but rather reducing the amount of annotation required. At least for the named-entity task we studied, using the method described, a single annotator could begin work after breakfast and, by lunchtime, have enough data annotated to achieve an F-score of 90.

References

Avrim Blum, Tom Mitchell. 1998. *Combining Labeled and Unlabeled Data with Co-Training.* COLT: Proceedings of the Workshop on Computational Learning Theory, Morgan Kaufmann Publishers

Elizabeth Boschee, Scott Miller, Ralph Weischedel. 2002. *Rapid Annotation through Human-Machine Collaboration.* HLT-2002

Peter F. Brown, Vincent J. Della Pietra, Peter V. deSouza, Jenifer C. Lai, Robert L. Mercer. 1990. *Class-based n-gram models of natural language.* Computational Linguistics

Michael Collins. 2002. *Discriminative Training Methods for Hidden Markov Models: Theory and Experiments with Perceptron Algorithms.* EMNLP 2002

Michael Collins. 1997. *Three Generative, Lexicalised Models for Statistical Parsing.* Proceedings of the 35th Annual Meeting of the ACL, Madrid.

Michael Collins and Yoram Singer. 1999. *Unsupervised Models for Named Entity Classification.* EMNLP/VLC-99

David A. Cohn, Zoubin Ghahramani, Michael I. Jordan. 1996. *Active Learning with Statistical Models* Advances in Neural Information Processing Systems.

Ellen Riloff, Rosie Jones. 1999. *Learning Dictionaries for Information Extraction by Multi-Level Bootstrapping* AAAI/IAAI

John Lafferty, Andrew McCallum, Fernando Pereira. 2001. *Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data.* Proc. 18th International Conf. on Machine Learning

Lillian Lee and Fernando Pereira. 1999. *Distributional similarity models: Clustering vs. nearest neighbors.* Proceedings of the 37th ACL, pp 33--40.

Winston Lin, Roman Yangarber and Ralph Grishman. 2003. *Bootstrapped Learning of Semantic Classes from Positive and Negative Examples.* Proceedings of the ICML-2003 Workshop on The Continuum from Labeled to Unlabeled Data, Washington, D.C.

David M. Magerman. 1995. *Statistical Decision-Tree Models for Parsing.* Proceedings, ACL Conference.

Andrew McCallum, Kamal Nigam. 1998. *Employing EM in Pool-Based Active Learning for Text Classification.* Proceedings of ICML-98, 15th International Conference on Machine Learning

Fernando Pereira, Naftali Tishby, and Lillian Lee. 1993. *Distributional Clustering of English Words.* Proceedings of the 31st ACL, pp 183--190

Fei Sha and Fernando Pereira. 2003. *Shallow Parsing with Conditional Random Fields.* Proceedings of Human Language Technology-NAACL, Edmonton, Canada.

David Yarowsky. 1995. *Unsupervised Word Sense Disambiguation Rivaling Supervised Methods.* Meeting of the Association for Computational Linguistics