Named Entity Recognition using Hundreds of Thousands of Features

James Mayfield and **Paul McNamee** and **Christine Piatko**
The Johns Hopkins University Applied Physics Laboratory
11100 Johns Hopkins Road, Laurel, Maryland 20723-6099 USA
{mayfield,mcnamee,piatko}@jhuapl.edu

Abstract

We present an approach to named entity recognition that uses support vector machines to capture transition probabilities in a lattice. The support vector machines are trained with hundreds of thousands of features drawn from the CoNLL-2003 Shared Task training data. Margin outputs are converted to estimated probabilities using a simple static function. Performance is evaluated using the CoNLL-2003 Shared Task test set; Test B results were $F_{\beta=1}$ = 84.67 for English, and $F_{\beta=1}$ = 69.96 for German.

1 Introduction

Language independence is difficult to achieve in named entity recognition (NER) because different languages appear to require different features. Most NER systems (or *taggers*) are severely limited in the number of features they may consider, because the computational expense of handling large numbers of features is high, and because the risk of overtraining increases with the number of features. Thus, the feature set must be finely tuned to be effective. Such constrained feature sets are naturally language dependent.

Increasing the number of features that a tagger can handle would ameliorate this problem, because the designer could select many relatively simple features in lieu of a few highly tuned features. Because support vector machines (SVMs) (Vapnik, 1995) can handle large numbers of parameters efficiently while simultaneously limiting overtraining, they are good candidates for application to named entity recognition. This paper proposes a novel way to use SVMs for named entity recognition called *SVM-Lattice*, describes a large feature space that we used on the CoNLL-2003 Shared Task (Tjong Kim Sang and De Meulder, 2003), and presents results from that task.

2 Model

We are interested in a lattice-based approach to named entity recognition. In this approach, each sentence is processed individually. A lattice is built with one column per word of the sentence (plus a start state). Each column contains one vertex for each possible tag. Each vertex in one column is connected by an edge to every vertex in the next column that may legitimately follow it (some transitions, such as from I-LOC to B-PER are disallowed). Given such a lattice, our task is first to assign probabilities to each of the arcs, then to find the highest likelihood path through the lattice based on those probabilities. This path corresponds to the highest likelihood tagging of the sentence.

Hidden Markov models break the probability calculations into two pieces: transition probabilities (the probability of moving from one vertex to another independent of the word at the destination node), and emission probabilities (the probability that a given word would be generated from a certain state independent of the path taken to get to that state). These probability distributions are calculated separately because the training data are typically too sparse to support a reasonable maximum likelihood estimate of the joint probability. However, there is no reason that these two distributions could not be combined given a suitable estimation technique.

A support vector machine is a binary classifier that uses supervised training to predict whether a given vector is in a target class. All SVM training and test data occupy a single high-dimensional vector space. In its simplest form, training an SVM amounts to finding the hyperplane that separates the positive training samples from the negative samples by the largest possible margin. This hyperplane is then used to classify the test vectors; those that lie on one side of the hyperplane are classified as members of the positive class, while others are classified as members of the negative class. In addition to the classification decision, the SVM also produces a *margin* for each vector–its distance from the hyperplane.

SVMs have two useful properties for our purposes. First, they can handle very high dimensional spaces, as long as individual vectors are sparse (*i.e.*, each vector has extent along only a small subset of the dimensions). Secondly, SVMs are resistant to overtraining, because only the training vectors that are closest to the hyperplane (called *support vectors*) dictate the parameters for the hyperplane. So SVMs would seem to be ideal candidates for estimating lattice probabilities.

Unfortunately, SVMs do not produce probabilities, but rather margins. In fact, one of the reasons that SVMs work so well is precisely because they do not attempt to model the entire distribution of training points. To use SVMs in a lattice approach, then, a mechanism is needed to estimate probability of category membership given a margin.

Platt (1999) suggests such a method. If the range of possible margins is partitioned into bins, and positive and negative training vectors are placed into these bins, each bin will have a certain percentage of positive examples. These percentages can be approximated by a sigmoid function: $P(y = 1 \mid f) = 1/(1 + exp(Ax + b))$. Platt gives a simple iterative method for estimating sigmoid parameters A and B, given a set of training vectors and their margins.

This approach can work well if a sufficient number of positive training vectors are available. Unfortunately, in the CoNLL-2003 shared task, many of the possible label transitions have few exemplars. Two methods are available to handle insufficient training data: smoothing, and guessing.

In the smoothing approach, linear interpolation is used to combine the model for the source to target pair that lacks sufficient data with the model made from a combination of all transitions going to the target label. For example, we could smooth the probabilities derived for the I-ORG to I-LOC transition with the probability that *any* tag would transition to the I-LOC state at the same point in the sentence.

The second approach is to guess at an appropriate model without examining the training data. While in theory this could prove to be a terrible approach, in practice for the Shared Task, selection of fixed sigmoid parameters works better than using Platt's method to train the parameters. Thus, we fix $A = -2$ and $b = 0$. We continue to believe that Platt's method or something like it will ultimately lead to superior performance, but our current experiments use this untrained model.

Our overall approach then is to use SVMs to estimate lattice transition probabilities. First, due to the low frequency of B-XXX tags in the training data, we convert each B-XXX tags to the corresponding I-XXX tag; thus, our system never predicts B-XXX tags. Then, we featurize the training data, forming sparse vectors suitable for input to our SVM package, SVMLight 5.00 (Joachims, 1999). Our feature set is described in the following section. Next, we train one SVM for each transition type seen in the training data. We used a cubic kernel for all of our experiments; this kernel gives a consistent boost over a linear kernel, while still training in a reasonable amount of time. If we were to use Platt's approach, the resulting classifiers would be applied to further (preferably held-out) training data to produce a set of margins, which would be used to estimate appropriate sigmoid parameters for each classifier. Sigmoid estimates that suffered from too few positive input vectors would be replaced by static estimates, and the sigmoids would optionally be smoothed.

To evaluate a test set, the test input is featurized using the same features as were used with the training data, resulting in a separate vector for each word of the input. Each classifier built during the training phase is then applied to each test vector to produce a margin. The margin is mapped to a probability estimate using the static sigmoid described above. When all of the probabilities have been estimated and applied to the lattice, a Viterbi-like algorithm is used to find the most likely path through the lattice. This path identifies the final tag for each word of the input sentence.

3 Features

The advantage of the ability to handle large numbers of features is that we do not need to consider how well a feature is likely to work in a particular language before proposing it. We use the following features:

1. the word itself, both unchanged and lower-cased;
2. the character 3-grams and 4-grams that compose the word;
3. the word's capitalization pattern and digit pattern;
4. the inverse of the word's length;
5. whether the word contains a dash;
6. whether the word is inside double quote marks;
7. the inverse of the word's position in the sentence, and of the position of that sentence in the document;
8. the POS, CHUNK and LEMMA features from the training data;
9. whether the word is part of any entity, according to a previous application of the TnT-Subcat tagger (Brants, 2000) (see below) trained on the tag set $\{$O, I-ENTITY$\}$ (Test A $F_{\beta=1}$ performance was 94.70 English and 74.33 German on this tag set); and

Run Description	Test	LOC	MISC	ORG	PER	Overall
1. Tnt	Test A	86.67	79.60	73.04	88.54	82.90
	Test B	81.28	68.98	65.71	82.84	75.54
2. Tnt + subcat	Test A	91.46	81.41	80.63	91.64	87.49
	Test B	85.71	68.41	73.82	87.95	80.68
3. SVM-Lattice	Test A	92.14	84.86	83.70	93.73	89.63
	Test B	87.09	72.81	78.84	90.40	83.92
4. SVM-Lattice+	Test A	93.75	86.02	85.90	93.91	90.85
	Test B	88.77	74.19	79.00	90.67	84.67

Table 1: English evaluation results. $F_{\beta=1}$ measures for subcategories, and overall.

Run Description	Test	LOC	MISC	ORG	PER	Overall
1. Tnt	Test A	59.51	49.58	48.71	53.77	53.29
	Test B	66.16	46.45	50.00	64.51	59.01
2. Tnt + subcat	Test A	67.62	54.97	56.18	65.04	61.46
	Test B	66.13	46.01	55.35	74.07	62.90
3. SVM-Lattice	Test A	67.04	54.18	65.77	64.01	63.48
	Test B	68.47	51.88	60.67	73.07	65.47
4. SVM-Lattice+	Test A	72.58	58.13	65.76	74.92	68.72
	Test B	73.60	50.98	63.69	80.20	69.96

Table 2: German evaluation results. $F_{\beta=1}$ measures for subcategories, and overall.

10. the maximum likelihood estimate, based on the training data, of the word's prior probability of being in each class.

In some runs, we also use:

11. the tag assigned by a previous application of the SVM-Lattice tagger, or by another tagger.

Each of these features is applied not just to the word being featurized, but also to a range of words on either side of it. We typically use a range of three (or, phrased differently, a centered window of seven). We also applied some of these features to the environment of the first occurrence of the word in the document. For example, if the first occurrence of 'Bush' in the document were followed by 'League,' then the second occurrence of 'Bush' would receive the feature 'first-occurrence-is-followed-by-league.'

Some values of the above features will be encountered during testing but not during training. For example, a word that occurs in the test set but not the training set will lack a known value for the first feature in the list above. To handle these cases, we assign any feature that appears only once in the training data to a special 'never-before-seen' class. This gives us examples at training time of unseen features, which we can then train on.

Using the Shared Task English training data, this approach to featurization leads to a feature space of well over 600,000 features, while the German data results in over a million features. Individual vectors typically have extent along a few hundred of these features.

There is a significant practical consideration in applying the method. The vectors produced by the featurizer for input to the SVM package are voluminous, leading to significant I/O costs, and slowing tag assignment. Two methods might ameliorate this problem. First, simple compression techniques would be quite effective in reducing file sizes, if the SVM package would support them. Secondly, most vectors represent negative examples; a portion of these could probably be eliminated entirely without significantly affecting system performance.

We have done no tuning of our feature set, preferring to spend our time adding new features and relying on the SVMs to ignore useless features. This is advantageous when applying the technique to a language that we do not understand (such as any of the world's various non-English languages).

4 Results

We evaluated our approach using the CoNLL-2003 English and German training and test sets, and the *conlleval* scoring software. We ran two baseline tests using Thorsten Brants' TnT tagger (2000), and two tests of SVM-Lattice:

1. **TnT**: The TnT tagger applied as distributed.

2. **TnT+subcat**: The TnT tagger applied to a refined tag set. Each tag type was subcategorized into about forty subtag types; each instance of a tag in the text was then replaced by the appropriate subtag. For example, a number (e.g., 221) that was part of a location received an I-LOC-alldigits tag; a location with an initial capital letter (*e.g.*, Baker) received an I-LOC-initcap tag; and one of the 30 most common words (e.g., of) that was part of a location received a (word-specific) I-LOC-of tag. This run served both to calibrate the SVM-Lattice performance scores, and to provide input for the **SVM-Lattice+** run below.

3. **SVM-Lattice**: Features 1-10 (listed above in the Features section)

4. **SVM-Lattice+**: Features 1-11, using the output of runs **SVM-Lattice** and **TnT+subcat** as input features.

Scores for each English test are shown in Table 1; German tests are shown in Table 2. Table 3 shows the results of the **SVM-Lattice+** run in more detail. The results show that the technique performs well, at least compared with the baseline technique provided with the CoNLL-2003 data (whose English Test B $F_{\beta=1}$ measure is 59.61 English and 30.30 German).

5 Conclusion

The SVM-Lattice approach appears to give good results without language-specific tuning; it handily outperforms the CoNLL-2003 Shared Task baseline, and beats a basic HMM tagger as well. Use of SVMs allows the introduction of a large number of features. These features can be introduced with little concern for dependency among features, and without significant knowledge of the target language. It is likely that our results reflect some degree of overfitting, given the large number of parameters we use; however, we suspect this effect is not large. Thus, the SVM-Lattice technique is particularly well suited to language-neutral entity recognition. We expect it will also perform well on other tasks that can be cast as tagging problems, such as part-of-speech tagging and syntactic chunking.

Acknowledgments

Significant theoretical and implementation contributions were made to this work by Claudia Pearce, for which we are grateful.

We gratefully acknowledge the provision of the *Reuters Corpus Vol. 1: English language, 1996-08-20 to 1997-08-19* by Reuters Limited.

English devel.	Precision	Recall	$F_{\beta=1}$
LOC	94.42%	93.09%	93.75
MISC	88.80%	83.41%	86.02
ORG	85.24%	86.58%	85.90
PER	92.79%	95.06%	93.91
overall	90.97%	90.73%	90.85

English test	Precision	Recall	$F_{\beta=1}$
LOC	88.22%	89.33%	88.77
MISC	74.89%	73.50%	74.19
ORG	79.31%	78.69%	79.00
PER	89.71%	91.65%	90.67
overall	84.45%	84.90%	84.67

German devel.	Precision	Recall	$F_{\beta=1}$
LOC	72.77%	72.40%	72.58
MISC	71.00%	49.21%	58.13
ORG	72.57%	60.11%	65.76
PER	83.70%	67.81%	74.92
overall	75.48%	63.07%	68.72

German test	Precision	Recall	$F_{\beta=1}$
LOC	75.08%	72.17%	73.60
MISC	63.62%	42.54%	50.98
ORG	69.20%	58.99%	63.69
PER	86.53%	74.73%	80.20
overall	75.97%	64.82%	69.96

Table 3: Results for the development and test evaluations for the English and German tasks.

References

Thorsten Brants. 2000. TnT-A statistical part-of-speech tagger. In *Proceedings of ANLP-2000*. Seattle, Washington.

Thorsten Joachims. 1999. Making large-scale SVM learning practical. In C. Burges B. Schölkopf and A. Smola, editors, *Support Vector Learning*. MIT Press.

John C. Platt. 1999. Probabilistic Outputs for Support Vector Machines and Comparisons to Regularized Likelihood Methods. In B. Scholkopf A. Smola, P. Bartlett and D. Schuurmans, editors, *Advances in Large Margin Classifiers*. MIT Press.

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

Vladimir N. Vapnik. 1995. *The Nature of Statistical Learning Theory*. Springer-Verlag.