Semi-Supervised Learning of Named Entity Substructure

Alden Timme
aotimme@stanford.edu
CS229 Final Project

Advisor: Richard Socher
richard@socher.org

Abstract

The goal of this project was two-fold: (1) to provide an algorithm to correctly find and label named entities in text, and (2) to uncover substructure in the named entities (such as a first name, last name distinction among person entities). The underlying algorithm used is a Class Hidden Markov Model (CHMM), a Hidden Markov Model with hidden states that emit observed words as well as observed classes. This algorithm is further bolstered by incorporating features into the model, substituting the multinomial probability distributions for transitions and emissions in the model with the outputs of logistic regressions using the features.

1 Introduction

Named-Entity Recognition (NER) is a task in Natural Language Processing (NLP) which aims to identify entity types of interest in a collection of text. For example, one might want to find entities corresponding to a person (*PER*), location (*LOC*), or organization (*ORG*). Given a set of entity types, the task of NER is to find and correctly label entities of these types within text, and correctly label other words as other. Supervised training of a model to perform NER gives the model a set of labeled training data and aims to learn enough to correctly label entities on the test set. NER can be seen as a very useful task because such entities are often some of the most important words for determining the content of a document. As such, one of the obvious goals of this project was to maximize the precision and recall of NE assignments. Precision is the accuracy of predicted entities (e.g. the probability that an entity labeled *PER* is actually a person), and recall is the ability to recognize an entity (e.g. the probability that a person entity is actually labeled *PER*).

The other goal of this project, and a different objective than other NLP algorithms that are used for NER solely, is to see if the model could uncover a latent substructure to the named entities encountered. That is, we wanted to see if the model would be able to, for example, discriminate between the first name and last name of a person while labeling the entire entity as a person, and maybe even be able to discriminate a third subset which corresponds to an abbreviation of the first name. For instance, take the person *John Ratcliffe*, which could appear in its full form as *John Ratcliffe* or in other forms such as *John*, *Ratcliffe*, or *J. Ratcliffe*. An idea of the project was to see if the model could identify all occurrences as *PER* but also identify the differences in the words that make up the same entity - i.e. that *J.* and *Ratcliffe* are two different parts of the same entity.

Evaluation of the system is done on two different data sets. They are the CoNLL-2003 English NE data set [1] and the MUC-7 English NE data set [2]. Labeling and evaluation of the NER task requires that the system correctly determines the entities' types as well as their start and end boundaries.

[1] *www.cnts.ua.ac.be/conll2003/ner/*

[2] *www-nlpir.nist.gov/related_projects/muc/proceedings/muc_7_toc.html*

2 The Model

The basic model used for the project is a Class Hidden Markov Model as proposed in Krogh (1994). To supplement the ability of the CHMM as a NE classifier, we then incorporate features in the model using the same system as Berg-Kirkpatrick, et al (2010).

2.1 Class Hidden Markov Model

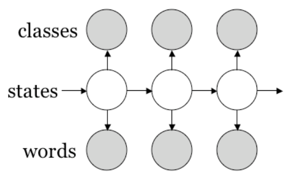

Figure 1: The graphical structure of a Class Hidden Markov Model for NER, in which classes and words are observed.

This is a Hidden Markov Model where there are two observed states for each hidden state. In particular, these observed states are the words of the text and the classes of entity to which they belong. In reality, a CHMM can be thought of as a regular HMM which emits a pair of values. However, thinking of it as a model with two separate emission types helps for inference, in which only the words are observed. One can either allow a probability distribution over classes for each state or assign a class deterministically to each state. In this project we make the simplifying assumption that each state is assigned to only one class. We also assign the same number of states to each entity type, including *other*.

Learning and inference with a CHMM is done in much the same way as with a normal HMM. In the learning period we wish to determine the transition and emission probabilities given the observed classes and words. This is done via a modified version of the forward-backward equations using the training data. With the one-class per state assumption, this amounts to finding looking at only valid paths through the model, where valid paths are those where the state labels agree with the observed class labels.

Inference on the test data is then done simply by performing the standard viterbi algorithm to find the hidden states given the observed words. The classes (named entities) are then deterministically given by the state assignments.

2.2 Feature-Enhanced CHMM

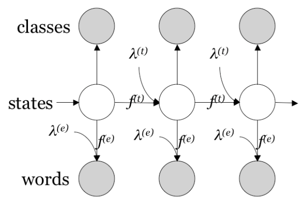

Figure 2: Feature-enhanced Class Hidden Markov Model for NER, where transitions and word emissions are feature-based, with features f and feature-weights λ. Superscript (t) denotes transition features and weights, and superscript (e) denotes emission features and weights.

The feature-enhanced Class Hidden Markov Model makes use of word features to enhance the performance of the CHMM. The idea for feature incorporation comes from Berg-Kirkpatrick et al. (2010), in which each component multinomial of the model become the outputs of local multi-class logistic regressions. That is, the transition probability between two states y and y' and the emission probability of word x from state y are given by, respectively,

$$\begin{aligned} p(y'|y, \lambda^{(t)}, f^{(t)}) &= \frac{\exp\left(\sum_i \lambda_i^{(t)} f_i^{(t)}(y, y')\right)}{\sum_{y''} \exp\left(\sum_i \lambda_i^{(t)} f_i^{(t)}(y, y'')\right)} \\ p(x|y, \lambda^{(e)}, f^{(e)}) &= \frac{\exp\left(\sum_i \lambda_i^{(e)} f_i^{(e)}(y, x)\right)}{\sum_{x'} \exp\left(\sum_i \lambda_i^{(e)} f_i^{(e)}(y, x')\right)} \end{aligned}$$

where superscripts (e) and (t) correspond to emission and transition features and weights.

Given a normal prior on the feature weights, we get the regularized log-likelihood of the observed data as

$$\mathcal{L}(\lambda) = \log P(X = x, C = c|\lambda, f) - \kappa||\lambda||_2^2$$

where x and c are the word and class sequences.

In order to find the optimal feature weights λ, we optimize the regularized log-likelihood by direct gradient ascent using any Hessian free optimizer (in this case, LBFS). In order to climb $\mathcal{L}$, we need a formula for its gradient. However, it turns out that its gradient is equal to the gradient of the regularized *expected* log likelihood,

$$\begin{aligned}\nabla\ell(\lambda,e) &= \sum_{y,y'} e_{y,y'}\left\{f(y,y') - \sum_{z''} a_{y,y''} f(y,y'')\right\}\\ &+ \sum_{y,x} e_{y,x}\left\{f(y,x) - \sum_{x'} b_{y,x'} f(y,x')\right\}\\ &- 2\kappa\lambda\end{aligned}$$

where

$$\begin{aligned}a_{y,y'} &= p(y'|y,\lambda^{(t)},f^{(t)})\\ b_{y,x} &= p(x|y,\lambda^{(e)},f^{(e)})\end{aligned}$$

as computed above, and $e_{y,x}, e_{y,y'}$ are the expected counts of the number of emissions of word x from state y and the expected number of transitions from y to y'. These expected counts are calculated through the modified viterbi algorithm where only valid paths are considered. At $e = e(\lambda_0)$,

$$\nabla_\lambda \ell(\lambda, e(\lambda_0)) = \nabla_\lambda \mathcal{L}(\lambda)$$

so we can use this gradient for the direct gradient ascent optimizations, recalculating $e(\lambda)$ at every step.

3 Results and Evaluation

We ran the model on two well known NER data sets, the CoNLL-2003 English data set and the MUC-7 English data set. The CoNLL data set has four different named entities, *location* (LOC), *miscellaneous* (MISC), *organization* (ORG), and *person* (PER), as well as the *other* (O) class. The MUC-7 data set has seven different entities, *date* (DATE), *location* (LOC), *money* (MONEY), *organization* (ORG), *percent* (PERCENT), *person* (PERSON), *location* (LOC), and *time* (TIME), along with the *other* (O) class. Results for each entity type (and all entity types) are measured in terms of precision, recall, and F_1,

$$\begin{aligned}\text{Precision:}\quad & P = \frac{TP}{TP+FP}\\ \text{Recall:}\quad & R = \frac{TP}{TP+FN}\\ F_1:\quad & F_1 = 2\left(\frac{P\cdot R}{P+R}\right)\end{aligned}$$

Feature	Example	Explanation
OneDigitNum	9	Number
TwoDigitNum	90	Year
FourDigitNum	1990	Year
YearDecade	1990s	Decade
ContainsDigitAndDash	03-90	Date
ContainsDigitAndOneSlash	3/4	Date
ContainsDigitAndComma	10,000	Money
ContainsDigitAndPeriod	10.00	Money/Percent
AllCaps	IBM	Organization
CapPeriod	J.	Name Abbr.
CapOtherPeriod	St.	Location Abbr.
CapPeriods	N.Y.	Loc or Org
InitialCap	Ratcliffe	Capitalized

Table 1: Features used for the feature-enhanced model on the MUC-7 data set.

where TP is the number of true positives (correct classifications), FP is the number of false positives (incorrectly classified the entity as its type), and FN is the number of false negatives (incorrectly classified entity as not its type).

For each data set, the model was first run without features, where the only "features" are the basic features. A basic feature is the baseline in which the only feature for a word is the word itself. This corresponds to what is the original Class Hidden Markov Model without any features. For all models, the only features used for transitions are the basic features, since there do not seem to be any notable features of states that would help with classification.

Runs were then done with features. For the CoNLL data set, I used two different feature sets. One feature set only includes two features: (1) all letters in a word are capitalized, and (2) the initial letter is capitalized. The second feature set was the "word shape", where there are only three word shapes: (1) all letters are capitalized (XX), (2) only the first letter is capitalized (Xx), and (3) all letters are lower case (xx). The MUC-7 data set feature-enhanced runs incorporated many more features (Table 1), culled from Zhou (2002). Figure 3 shows the F_1 scores for the basic and feature-enhanced runs on the CoNLL data set and MUC-7 data set with different values of the regularization parameter κ and different values for the number of states per class

SPC	Type	LOC	MISC	ORG	PER	OVERALL
1	Basic	0.7451	0.6500	0.5129	0.5732	0.6276
1	Capitalization	0.8135	0.7386	0.6654	0.8297	0.7744
1	Word Shape	0.7999	0.7301	0.6443	0.8191	0.7607
2	Basic	0.7247	0.6039	0.3959	0.6355	0.6105
2	Capitalization	0.7809	0.6517	0.5934	0.7989	0.7197
2	Word Shape	0.7844	0.7383	0.6662	0.8124	0.7584
5	Basic	0.7082	0.6364	0.4664	0.6630	0.6342
5	Capitalization	0.7565	0.6700	0.6172	0.7938	0.7228
5	Word Shape	0.7537	0.6724	0.6440	0.7987	0.7277

Table 2: F_1 scores for best regularization parameter ($\kappa = 0.25$) on CoNLL data set with different feature sets (SPC is the number of states per NE class).

SPC	Type	DATE	LOC	MONEY	ORG	PERCENT	PERSON	TIME	OVERALL
1	Basic	0.5986	0.5846	0.4673	0.4041	0.3582	0.3960	0.2297	0.4903
1	Features	0.6876	0.6991	0.5617	0.4518	0.8374	0.5129	0.2299	0.4582
2	Basic	0.5526	0.5193	0.5782	0.3699	0.8911	0.2759	0.3277	0.4706
2	Features	0.6833	0.6250	0.5899	0.5035	0.8812	0.4937	0.2275	0.5822
5	Basic	0.5662	0.5315	0.4593	0.3705	0.8469	0.3983	0.2793	0.5818
5	Features	0.7119	0.6669	0.5446	0.5848	0.8641	0.5517	0.2791	0.6274

Table 3: F_1 scores for best regularization parameter ($\kappa = 0.25$) on MUC-7 data set with and without features (SPC is the number of states per NE class).

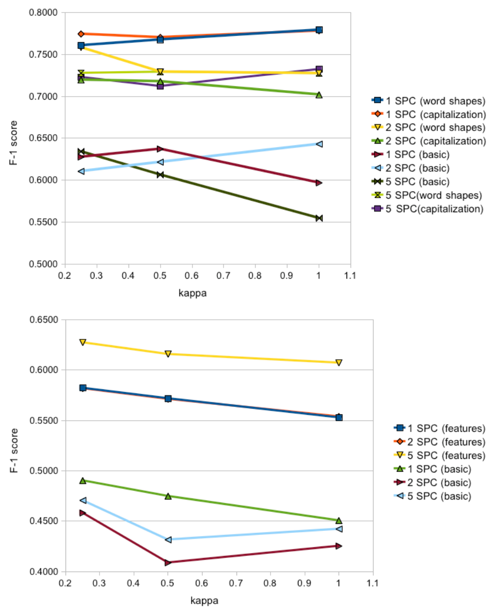

Figure 3: F_1 scores on the CoNLL data set (top) and MUC-7 data set (bottom) with different feature sets and different numbers of states per NE class.

(SPC). Tables 2 and 3 contain the results in terms of F_1 scores for the CoNLL and MUC-7 data sets respectively.

4 Conclusion

As can be seen fairly obviously from Figure 2 and Figure 3, incorporating features helps quite a bit. With both data sets, the F_1 scores are considerably lower when the model considers only the basic features. Unfortunately, however, there does not seem to be anything added by the Class HMM for the CoNLL data set. Having one state per class almost always does best, suggesting that a feature-enhanced regular HMM would do better than a Class HMM with a number of states per class. The MUC-7 data set does benefit from including a number of states per class. However, the observation on the CoNLL data set also suggests that the CHMM does not exploit a substructure to the named entities. And, in fact, from analysis of the inference on both the CoNLL and MUC-7 data sets, I cannot find any evidence of the model finding a substructure within the named entities it classifies.

The model also does not do a very good job on the NER task itself, falling well short of state-of-the-art numbers, which have F_1 scores around 0.89 for the CoNLL data set and 0.94 for the MUC-7 data set. Perhaps including more features would help, but I do not think that better features would account for a $20 - 30\%$ jump in F_1.

References

Anders Krogh. 1994. Hidden Markov Models for Labeled Sequences. *Proceedings of the 12th IAPR International Conference on Pattern Recognition*. Los Alamitos IEEE Computer Society Press, California.

GuoDong Zhou and Jian Su. 2002. Named Entity Recognition using and HMM-based Chunk Tagger. *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 473-480. Philadelphia, PA, July 2002.

Taylor Berg-Kirkpatrick, Bouchard-Cote Alexandre, DeNero John, Klein Dan. 2010. Painless Unsupervised Learning with Features. *The 2010 Conference of the North American Chapter of the ACL*, pages 582-590. Los Angeles, California, June 2010.