Rapid Development of Hindi Named Entity Recognition using Conditional Random Fields and Feature Induction (Short Paper)

Wei Li
University of Massachusetts Amherst
and
Andrew McCallum
University of Massachusetts Amherst

This paper describes our application of Conditional Random Fields (CRFs) with feature induction to a Hindi named entity recognition task. With only five days development time and little knowledge of this language, we automatically discover relevant features by providing a large array of lexical tests and using feature induction to automatically construct the features that most increase conditional likelihood. In an effort to reduce overfitting, we use a combination of a Gaussian prior and early-stopping based on the results of 10-fold cross validation.

Categories and Subject Descriptors: I.2.7 [**Artificial Intelligence**]: Natural Language Processing - *Text analysis*; G.3 [**Mathematics of Computing**]: Probability and Statistics - *Probablistic algorithms*

General Terms: algorithms; experimentation; languages

Additional Key Words and Phrases: extraction; Conditional Random Fields; feature induction

1. INTRODUCTION

Conditional Random Fields (CRFs) [Lafferty et al. 2001] are undirected graphical models, a special case of which correspond to conditionally-trained probabilistic finite state automata. Being conditionally trained, these CRFs can easily incorporate a large number of arbitrary, non-independent features while still having efficient procedures for non-greedy finite-state inference and training. CRFs have shown success in various sequence modeling tasks including noun phrase segmentation [Sha and Pereira 2003] and table extraction [Pinto et al. 2003]. Following [McCallum 2003], we augment CRFs with feature induction to construct only those feature conjunctions that significantly increase the training label conditional likelihood.

In this paper, we apply CRFs with feature induction to the TIDES 2003 Surprise Language Hindi Named Entity Recognition task. Unlike English, for which

the authors know many helpful lexical structures such as capitalization patterns and word suffixes, the authors knew almost nothing about Hindi. However, given CRFs' tremendous freedom to include arbitrary features, and the ability of feature induction to automatically construct the most useful feature combinations, users of CRFs can simple provide a large menu of lexical feature tests consisting of anything they imagine might possibly be useful, and then let the training procedure automatically perform the feature engineering.

2. CONDITIONAL RANDOM FIELDS AND FEATURE INDUCTION

CRFs are undirected graphical models used to calculate the conditional probability of values on designated output nodes given values on other designated input nodes. In a special case where the output nodes form a linear chain, CRFs make a first-order Markov independence assumption, and can be viewed as conditionally-trained probabilistic finite state automata. (Second-order and higher-order models are also straightforward.) These models are analogous to maximum entropy / conditional log-linear finite state models, except that they are normalized over entire sequences rather than per-state. The conditional probability of a state sequence $s = \langle s_1, s_2, ...s_T \rangle$ given an observation sequence $o = \langle o_1, o_2, ...o_T \rangle$ is calculated as:

$$P_\Lambda(\mathbf{s}|\mathbf{o}) = \frac{1}{Z_\mathbf{o}} \exp\left(\sum_{t=1}^{T}\sum_{k} \lambda_k f_k(s_{t-1}, s_t, \mathbf{o}, t)\right),$$

where $f_k(s_{t-1}, s_t, \mathbf{o}, t)$ is a feature function whose weight λ_k is to be learned via training. Feature functions could ask arbitrary questions about the two consecutive states, any part of the observation sequence and the current position. Their values may range between $-\infty ... + \infty$, but typically they are binary. To make all conditional probabilities sum up to 1, we must calculate the normalization factor $Z_\mathbf{o} = \sum_\mathbf{s} \exp\left(\sum_{t=1}^{T}\sum_k \lambda_k f_k(s_{t-1}, s_t, \mathbf{o}, t)\right)$, which, as in HMMs, can be obtained efficiently by dynamic programming.

To train a CRF, the objective function to be maximized is the penalized log-likelihood of the state sequences given observation sequences:

$$L_\Lambda = \sum_{i=1}^{N} \log\left(P_\Lambda(\mathbf{s}^{(i)}|\mathbf{o}^{(i)})\right) - \sum_k \frac{\lambda_k^2}{2\sigma^2},$$

where $\{\langle \mathbf{o}^{(i)}, \mathbf{s}^{(i)} \rangle\}$ is the labeled training data. The second sum corresponds to a zero-mean, σ^2-variance Gaussian prior over parameters, which facilitates optimization by making the likelihood surface strictly convex. The Gaussian prior has also been thought to significantly reduce overfitting, however we find that its effectiveness for this purpose is less than widely believed. Tighter Gaussian priors slow convergence, and this seems to have imparted a certain "early stopping" effect to earlier experiments with inefficient training methods, such as Improved Iterative Scaling [Della Pietra et al. 1997]. Here, however, we set parameters λ to maximize this penalized log-likelihood using Limited-memory BFGS [Sha and Pereira 2003], a quasi-Newton method that is significantly more efficient, and which results in only minor changes in accuracy due to changes in σ. The Gaussian prior encourages smaller weights, but complex trade-offs in weights can still be made at a lower

resolution.

Since CRFs are log-linear models, and high accuracy may require complex decision boundaries that are non-linear in the space of original features, the expressive power of the models is often increased by adding new features that are *conjunctions* of the original features. For instance, a conjunction feature might ask if the current word is in a lexicon of organization names *and* the next word is *"spokesman"*.

With conjunctions such as this, one could create arbitrarily complicated features. However, it is infeasible to incorporate all possible conjunctions. For example, while certain word tri-grams are important, including all tri-grams will overflow memory and also exacerbate overfitting. So we turn to feature induction as described in [McCallum 2003], aiming to create only those feature conjunctions that will significantly improve performance. We start with no features at all and choose new features iteratively. In each iteration, some set of candidates are evaluated (also using the Gaussian prior), and the best ones are added to the model. It is not necessary that all atomic features are used. This allows us to liberally guess about which observational tests might be useful, without being concerned about forcing harmful features into the model. Conditional Random Fields and this feature induction method are described in significantly greater detail in [McCallum 2003].

3. EVALUATION

When applying CRFs to the named entity recognition problem, an observation sequence is the token sequence of a sentence or document of text and the state sequence is its corresponding label sequence. The Hindi task requires us to find all appearances of three types of entities: PERSON, LOCATION and ORGANIZATION. To recognize entity boundaries, we have two kinds of labels for each entity type: B-TYPE for the start of an entity and I-TYPE for the inner part. For example, "*New York City*" will be labeled as "B-LOCATION I-LOCATION I-LOCATION". For non-entities, we use the label O.

While CRFs generally can use real-valued feature functions, in our experiments, all features are binary. A feature function $f_k(s_{t-1}, s_t, \mathbf{o}, t)$ has a value of 0 for most cases and is only set to be 1 when s_{t-1}, s_t are certain states and the observation has certain properties. Unfamiliar with Hindi, the authors have little knowledge about what properties should be included. So we literally guess which features might be relevant and let CRFs and feature induction discover the useful ones. The atomic feature tests we have provide to the model include the entire word text, character n-grams (n=2, 3, 4), word prefix and suffix of lengths 2, 3 and 4, and 24 Hindi gazetteer lists provided at the Surprise Language resource website. We then make available to the feature induction procedure these atomic features at the current, previous and next sequence positions.

Our training set for the Hindi task is composed of 601 BBC and 27 EMI documents after we remove the ones with no tag files or containing the NO-ANNOTATION tags. It contains about 340k words, 4540 PERSON, 7342 LOCATION and 3181 ORGANIZATION entities. In the 25 documents in the NIST test set, there are 10k words and the entity counts are 152, 232 and 92 respectively. To train the CRF, we experimented with various options, such as first-order versus second-order models, using feature induction or not and using lexicons or not. In an effort to reduce

% training data	10	50	100	100	100	100	100	100	100	100
Markov order	1	1	1	2	1	1	1	1	1	1
feature induction	Y	Y	Y	Y	N	Y	Y	Y	Y	Y
using lexicons	Y	Y	Y	Y	Y	N	Y	Y	Y	Y
early-stopping	N	N	N	N	N	N	Y	N	N	N
Gaussian prior	100.0	100.0	100.0	100.0	100.0	100.0	100.0	10.0	1.0	0.1
validation set F1	65.82	77.13	81.16	79.51	-	81.31	**82.55**	80.73	80.66	78.80
test set F1	56.68	66.46	**71.50**	-	62.94	70.77	68.80	70.62	69.27	63.16

Table I. Experiment Results

overfitting, we have also tried different Gaussian priors and early-stopping. Finally, a first-order CRF is trained with the whole training set, inducing 500 or fewer features (down to a gain threshold of 5.0) every 10 iterations. Feature induction constructs 9697 features from an original set of 152,189 atomic features; many are position-shifted but only about 1% are conjunctions. Sample features include the Hindi word for "*in*" at position $t + 1$, which possibly indicates a LOCATION entity, and a typical suffix of country names followed by the word "*minister*".

The experiment results for validation and test sets are summarized in Figure 1. The first-order model performs slightly better than the second-order model on the validation set, and the testing performance is significantly better when using feature induction. Using lexicons or not does not make much difference, and tight Gaussian priors do not improve the performance. While an early-stopping point of 240 iterations of L-BFGS obtains the highest average F1 score for the 10-fold cross validation experiments, early-stopping actually hurts the performance on the test set. Although performance is similar to an HMM on a validation set [May *et. al*, this issue], our model does not perform as well on the test set. We hypothesize that both of these phenomena may be due to the significant mismatch between the training/validation data and the test data.

Acknowledgments

We thank Leah Larkey for help with data normalization, and to Hema Raghavan and Nasreen Abdul Jaleel, who provided helpful analysis after model development. This work was supported in part by the Center for Intelligent Information Retrieval; and SPAWARSYSCEN-SD grant numbers N66001-99-1-8912 and N66001-02-1-8903.

REFERENCES

DELLA PIETRA, S., DELLA PIETRA, V. J., AND LAFFERTY, J. D. 1997. Inducing features of random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence 19,* 4, 380–393.

LAFFERTY, J., MCCALLUM, A., AND PEREIRA, F. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. 18th International Conf. on Machine Learning*.

MCCALLUM, A. 2003. Efficiently inducing features of conditional random fields. In *Nineteenth Conference on Uncertainty in Artificial Intelligence (UAI03)*.

PINTO, D., MCCALLUM, A., WEI, X., AND CROFT, W. B. 2003. Table extraction using conditional random fields. In *Proceedings of SIGIR 03 Conference, Toronto, Canada*.

SHA, F. AND PEREIRA, F. 2003. Shallow parsing with conditional random fields. In *Proceedings of Human Language Technology, NAACL*.

Short Paper