Early Results for Named Entity Recognition with Conditional Random Fields, Feature Induction and Web-Enhanced Lexicons

Andrew McCallum
Department of Computer Science
University of Massachusetts Amherst
Amherst, MA 01003
mccallum@cs.umass.edu

Wei Li
Department of Computer Science
University of Massachusetts Amherst
Amherst, MA 01003
weili@cs.umass.edu

1 Introduction

Models for many natural language tasks benefit from the flexibility to use overlapping, non-independent features. For example, the need for labeled data can be drastically reduced by taking advantage of domain knowledge in the form of word lists, part-of-speech tags, character n-grams, and capitalization patterns. While it is difficult to capture such inter-dependent features with a generative probabilistic model, conditionally-trained models, such as conditional maximum entropy models, handle them well. There has been significant work with such models for greedy sequence modeling in NLP (Ratnaparkhi, 1996; Borthwick et al., 1998).

Conditional Random Fields (CRFs) (Lafferty et al., 2001) are undirected graphical models, a special case of which correspond to conditionally-trained finite state machines. While based on the same exponential form as maximum entropy models, they have efficient procedures for complete, non-greedy finite-state inference and training. CRFs have shown empirical successes recently in POS tagging (Lafferty et al., 2001), noun phrase segmentation (Sha and Pereira, 2003) and Chinese word segmentation (McCallum and Feng, 2003).

Given these models' great flexibility to include a wide array of features, an important question that remains is what features should be used? For example, in some cases capturing a word tri-gram is important, however, there is not sufficient memory or computation to include all word tri-grams. As the number of overlapping atomic features increases, the difficulty and importance of constructing only certain feature combinations grows.

This paper presents a feature induction method for CRFs. Founded on the principle of constructing only those feature conjunctions that significantly increase log-likelihood, the approach builds on that of Della Pietra *et al* (1997), but is altered to work with conditional rather than joint probabilities, and with a mean-field approximation and other additional modifications that improve efficiency specifically for a sequence model. In comparison with traditional approaches, automated feature induction offers both improved accuracy and significant reduction in feature count; it enables the use of richer, higher-order Markov models, and offers more freedom to liberally guess about which atomic features may be relevant to a task.

Feature induction methods still require the user to create the building-block atomic features. Lexicon membership tests are particularly powerful features in natural language tasks. The question is where to get lexicons that are relevant for the particular task at hand?

This paper describes *WebListing*, a method that obtains seeds for the lexicons from the labeled data, then uses the Web, HTML formatting regularities and a search engine service to significantly augment those lexicons. For example, based on the appearance of *Arnold Palmer* in the labeled data, we gather from the Web a large list of other golf players, including *Tiger Woods* (a phrase that is difficult to detect as a name without a good lexicon).

We present results on the CoNLL-2003 named entity recognition (NER) shared task, consisting of news articles with tagged entities PERSON, LOCATION, ORGANIZATION and MISC. The data is quite complex; for example the English data includes foreign person names (such as *Yayuk Basuki* and *Innocent Butare*), a wide diversity of locations (including sports venues such as *The Oval*, and rare location names such as *Nirmal Hriday*), many types of organizations (from company names such as *3M*, to acronyms for political parties such as *KDP*, to location names used to refer to sports teams such as *Cleveland*), and a wide variety of miscellaneous named entities (from software such as *Java*, to nationalities such as *Basque*, to sporting competitions such as *1,000 Lakes Rally*).

On this, our first attempt at a NER task, with just a few person-weeks of effort and little work on development-set error analysis, our method currently obtains overall English F1 of 84.04% on the test set by using CRFs, feature induction and Web-augmented lexicons. German F1 using very limited lexicons is 68.11%.

2 Conditional Random Fields

Conditional Random Fields (CRFs) (Lafferty et al., 2001) are undirected graphical models used to calculate the conditional probability of values on designated output nodes given values assigned to other designated input nodes.

In the special case in which the output nodes of the graphical model are linked by edges in a *linear chain*, CRFs make a first-order Markov independence assumption, and thus can be understood as conditionally-trained finite state machines (FSMs). In the remainder of this section we introduce the likelihood model, inference and estimation procedures for CRFs.

Let $\mathbf{o} = \langle o_1, o_2, ...o_T \rangle$ be some observed input data sequence, such as a sequence of words in text in a document, (the values on n input nodes of the graphical model). Let $\mathcal{S}$ be a set of FSM states, each of which is associated with a label, $l \in \mathcal{L}$, (such as ORG). Let $\mathbf{s} = \langle s_1, s_2, ...s_T \rangle$ be some sequence of states, (the values on T output nodes). By the Hammersley-Clifford theorem, CRFs define the conditional probability of a state sequence given an input sequence to be

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

where $Z_\mathbf{o}$ is a normalization factor over all state sequences, $f_k(s_{t-1}, s_t, \mathbf{o}, t)$ is an arbitrary feature function over its arguments, and λ_k is a learned weight for each feature function. A feature function may, for example, be defined to have value 0 in most cases, and have value 1 if and only if s_{t-1} is state #1 (which may have label OTHER), and s_t is state #2 (which may have label LOCATION), and the observation at position t in $\mathbf{o}$ is a word appearing in a list of country names. Higher λ weights make their corresponding FSM transitions more likely, so the weight λ_k in this example should be positive. More generally, feature functions can ask powerfully arbitrary questions about the input sequence, including queries about previous words, next words, and conjunctions of all these, and $f_k(\cdot)$ can range $-\infty ... \infty$.

CRFs define the conditional probability of a label sequence based on total probability over the state sequences, $P_\Lambda(\mathbf{l}|\mathbf{o}) = \sum_{\mathbf{s}:l(\mathbf{s})=\mathbf{l}} P_\Lambda(\mathbf{s}|\mathbf{o})$, where $l(\mathbf{s})$ is the sequence of labels corresponding to the labels of the states in sequence $\mathbf{s}$.

Note that the normalization factor, $Z_\mathbf{o}$, is the sum of the "scores" of all possible state sequences, $Z_\mathbf{o} = \sum_{\mathbf{s} \in \mathcal{S}^T} \exp\left(\sum_{t=1}^{T} \sum_k \lambda_k f_k(s_{t-1}, s_t, \mathbf{o}, t) \right)$, and that the number of state sequences is exponential in the input sequence length, T. In arbitrarily-structured CRFs, calculating the normalization factor in closed form is intractable, but in linear-chain-structured CRFs, as in *forward-backward* for hidden Markov models (HMMs), the probability that a particular transition was taken between two CRF states at a particular position in the input sequence can be calculated efficiently by dynamic programming. We define slightly modified *forward values*, $\alpha_t(s_i)$, to be the "unnormalized probability" of arriving in state s_i *given* the observations $\langle o_1, ...o_t \rangle$. We set $\alpha_0(s)$ equal to the probability of starting in each state s, and recurse:

$$\alpha_{t+1}(s) = \sum_{s'} \alpha_t(s') \exp\left(\sum_k \lambda_k f_k(s', s, \mathbf{o}, t) \right).$$

The backward procedure and the remaining details of Baum-Welch are defined similarly. $Z_\mathbf{o}$ is then $\sum_s \alpha_T(s)$. The Viterbi algorithm for finding the most likely state sequence given the observation sequence can be correspondingly modified from its HMM form.

2.1 Training CRFs

The weights of a CRF, $\Lambda = \{\lambda, ...\}$, are set to maximize the conditional log-likelihood of labeled sequences in some training set, $\mathcal{D} = \{\langle \mathbf{o}, \mathbf{l} \rangle^{(1)}, ...\langle \mathbf{o}, \mathbf{l} \rangle^{(j)}, ...\langle \mathbf{o}, \mathbf{l} \rangle^{(N)}\}$:

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

where the second sum is a Gaussian prior over parameters (with variance σ) that provides smoothing to help cope with sparsity in the training data.

When the training labels make the state sequence unambiguous (as they often do in practice), the likelihood function in exponential models such as CRFs is convex, so there are no local maxima, and thus finding the global optimum is guaranteed. It has recently been shown that quasi-Newton methods, such as L-BFGS, are significantly more efficient than traditional iterative scaling and even conjugate gradient (Malouf, 2002; Sha and Pereira, 2003). This method approximates the second-derivative of the likelihood by keeping a running, finite-sized window of previous first-derivatives.

L-BFGS can simply be treated as a black-box optimization procedure, requiring only that one provide the first-derivative of the function to be optimized. Assuming that the training labels on instance j make its state path unambiguous, let $\mathbf{s}^{(j)}$ denote that path, and then the first-derivative of the log-likelihood is

$$\begin{aligned} \frac{\delta L}{\delta \lambda_k} &= \left(\sum_{j=1}^{N} C_k(\mathbf{s}^{(j)}, \mathbf{o}^{(j)}) \right) - \\ &\quad \left(\sum_{j=1}^{N} \sum_{\mathbf{s}} P_\Lambda(\mathbf{s}|\mathbf{o}^{(j)}) C_k(\mathbf{s}, \mathbf{o}^{(j)}) \right) - \frac{\lambda_k}{\sigma^2} \end{aligned}$$

where $C_k(\mathbf{s}, \mathbf{o})$ is the "count" for feature k given $\mathbf{s}$ and $\mathbf{o}$, equal to $\sum_{t=1}^{T} f_k(s_{t-1}, s_t, \mathbf{o}, t)$, the sum of

$f_k(s_{t-1}, s_t, \mathbf{o}, t)$ values for all positions, t, in the sequence $\mathbf{s}$. The first two terms correspond to the difference between the empirical expected value of feature f_k and the model's expected value: $(\tilde{E}[f_k] - E_\Lambda[f_k])N$. The last term is the derivative of the Gaussian prior.

3 Efficient Feature Induction for CRFs

Typically the features, f_k, are based on some number of hand-crafted atomic observational tests (such as *word is capitalized* or *word is "said"*, or *word appears in lexicon of country names*), and a large collection of features is formed by making conjunctions of the atomic tests in certain user-defined patterns; (for example, the conjunctions consisting of all tests at the current sequence position conjoined with all tests at the position one step ahead—specifically, for instance, *current word is capitalized and next word is "Inc"*). There can easily be over 100,000 atomic tests (mostly based on tests for the identity of words in the vocabulary), and ten or more shifted-conjunction patterns—resulting in several million features (Sha and Pereira, 2003). This large number of features can be prohibitively expensive in memory and computation; furthermore many of these features are irrelevant, and others that are relevant are excluded.

In response, we wish to use just those time-shifted conjunctions that will significantly improve performance. We start with no features, and over several rounds of feature induction: (1) consider a set of proposed new features, (2) select for inclusion those candidate features that will most increase the log-likelihood of the correct state path $\mathbf{s}^{(j)}$, and (3) train weights for all features. The proposed new features are based on the hand-crafted observational tests—consisting of singleton tests, and binary conjunctions of tests with each other and with features currently in the model. The later allows arbitrary-length conjunctions to be built. The fact that not all singleton tests are included in the model gives the designer great freedom to use a very large variety of observational tests, and a large window of time shifts.

To consider the effect of adding a new feature, define the new sequence model with additional feature, g, having weight μ, to be

$$P_{\Lambda+g,\mu}(\mathbf{s}|\mathbf{o}) = \frac{P_\Lambda(\mathbf{s}|\mathbf{o}) \exp\left(\sum_{t=1}^{T} \mu\, g(s_{t-1}, s_t, \mathbf{o}, t)\right)}{Z_\mathbf{o}(\Lambda, g, \mu)};$$

$Z_\mathbf{o}(\Lambda, g, \mu) \overset{\text{def}}{=} \sum_{\mathbf{s}'} P_\Lambda(\mathbf{s}'|\mathbf{o}) \exp(\sum_{t=1}^{T} \mu\, g(s'_{t-1}, s'_t, \mathbf{o}, t))$ in the denominator is simply the additional portion of normalization required to make the new function sum to 1 over all state sequences.

Following (Della Pietra et al., 1997), we efficiently assess many candidate features in parallel by assuming that the λ parameters on all included features remain fixed while estimating the *gain*, $G(g)$, of a candidate feature, g, based on the improvement in log-likelihood it provides,

$$G_\Lambda(g) = \max_\mu\ G_\Lambda(g, \mu) = \max_\mu\ L_{\Lambda+g\mu} - L_\Lambda.$$

where $L_{\Lambda+g\mu}$ includes $-\mu^2/2\sigma^2$.

In addition, we make this approach tractable for CRFs with two further reasonable and mutually-supporting approximations specific to CRFs. (1) We avoid dynamic programming for inference in the gain calculation with a mean-field approximation, removing the dependence among states. (Thus we transform the gain from a sequence problem to a token classification problem. However, the original posterior distribution over states given each token, $P_\Lambda(s|\mathbf{o}) = \alpha_t(s|\mathbf{o})\beta_{t+1}(s|\mathbf{o})/Z_\mathbf{o}$, is still calculated by dynamic programming without approximation.) Furthermore, we can calculate the gain of aggregate features irrespective of transition source, $g(s_t, \mathbf{o}, t)$, and expand them after they are selected. (2) In many sequence problems, the great majority of the tokens are correctly labeled even in the early stages of training. We significantly gain efficiency by including in the gain calculation only those tokens that are mislabeled by the current model. Let $\{o(i) : i = 1...M\}$ be those tokens, and $\mathbf{o}(i)$ be the input sequence in which the ith error token occurs at position $t(i)$. Then algebraic simplification using these approximations and previous definitions gives $G_\Lambda(g, \mu) =$

$$\sum_{i=1}^{M} \log\left(\frac{\exp\left(\mu\, g(s_{t(i)}, \mathbf{o}(i), t(i))\right)}{Z_{o(i)}(\Lambda, g, \mu)}\right) - \frac{\mu^2}{2\sigma^2}$$

$$= M\mu\, \tilde{E}[g] - \sum_{i=1}^{M} \log(E_\Lambda[\exp(\mu\, g)|\mathbf{o}(i)] - \frac{\mu^2}{2\sigma^2},$$

where $Z_{o(i)}(\Lambda, g, \mu)$ (with non-bold o) is simply $\sum_s P_\Lambda(s|\mathbf{o}(i)) \exp(\mu g(s, \mathbf{o}(i), t(i)))$. The optimal values of the μ's cannot be solved in closed form, but Newton's method finds them all in about 12 quick iterations.

There are two additional important modeling choices: (1) Because we expect our models to still require several thousands of features, we save time by adding *many* of the features with highest gain each round of induction rather than just one; (including a few redundant features is not harmful). (2) Because even models with a small select number of features can still severely overfit, we train the model with just a few BFGS iterations (not to convergence) before performing the next round of feature induction. Details are in (McCallum, 2003).

4 Web-augmented Lexicons

Some general-purpose lexicons, such a surnames and location names, are widely available, however, many natural language tasks will benefit from more task-specific lexicons, such as lists of soccer teams, political parties, NGOs and English counties. Creating new lexicons entirely by hand is tedious and time consuming.

English	Development			Test		
	Prec	Recall	F1	Prec	Recall	F1
LOC	93.82	91.78	92.79	87.23	87.65	87.44
MISC	83.99	78.52	81.17	74.44	71.37	72.87
ORG	84.23	82.03	83.11	79.52	78.33	78.92
PER	92.64	93.65	93.14	91.05	89.98	90.51
Overall	89.84	88.10	**88.96**	84.52	83.55	**84.04**

German	Development			Test		
	Prec	Recall	F1	Prec	Recall	F1
LOC	68.55	68.84	68.69	71.92	69.28	70.57
MISC	72.66	45.25	55.77	69.59	42.69	52.91
ORG	70.64	54.88	61.77	63.85	48.90	55.38
PER	82.21	64.31	72.17	90.04	74.14	81.32
Overall	73.60	59.01	**65.50**	75.97	61.72	**68.11**

Figure 1: English and German named entity extraction.

Using a technique we call *WebListing*, we build lexicons automatically from HTML data on the Web. Previous work has built lexicons from fixed corpora by determining linguistic patterns for the context in which relevant words appear (Collins and Singer, 1999; Jones et al., 1999). Rather than mining a small corpus, we gather data from nearly the entire Web; rather than relying on fragile linguistic context patterns, we leverage robust formatting regularities on the Web. WebListing finds co-occurrences of seed terms that appear in an identical HTML formatting pattern, and augments a lexicon with other terms on the page that share the same formatting. Our current implementation uses GoogleSets, which we understand to be a simple implementation of this approach based on using HTML list items as the formatting regularity. We are currently building a more sophisticated replacement.

5 Results

To perform named entity extraction on the news articles in the CoNLL-2003 English shared task, several families of features are used, all time-shifted by -2, -1, 0, 1, 2: (a) the word itself, (b) 16 character-level regular expressions, mostly concerning capitalization and digit patterns, such as `A`, `A+`, `Aa+`, `Aa+Aa*`, `A.`, `D+`, where `A`, `a` and `D` indicate the regular expressions `[A-Z]`, `[a-z]` and `[0-9]`, (c) 8 lexicons entered by hand, such as honorifics, days and months, (d) 15 lexicons obtained from specific web sites, such as countries, publicly-traded companies, surnames, stopwords, and universities, (e) 25 lexicons obtained by WebListing (including people names, organizations, NGOs and nationalities), (f) all the above tests with prefix firstmention from any previous duplicate of the current word, (if capitalized). A small amount of hand-filtering was performed on some of the WebListing lexicons. Since GoogleSets' support for non-English is severely limited, only 5 small lexicons were used for German; but character bi- and tri-grams were added.

A Java-implemented, first-order CRF was trained for about 12 hours on a 1GHz Pentium with a Gaussian prior variance of 0.5, inducing 1000 or fewer features (down to a gain threshold of 5.0) each round of 10 iterations of L-BFGS. Candidate conjunctions are limited to the 1000 atomic and existing features with highest gain. Performance results for each of the entity classes can be found in Figure 1. The model achieved an overall F1 of 84.04% on the English test set using 6423 features. (Using a set of fixed conjunction patterns instead of feature induction results in F1 73.34%, with about 1 million features; trial-and-error tuning the fixed patterns would likely improve this.) Accuracy gains are expected from experimentation with the induction parameters and improved WebListing.

Acknowledgments

We thank John Lafferty, Fernando Pereira, Andres Corrada-Emmanuel, Drew Bagnell and Guy Lebanon, for helpful input. This work was supported in part by the Center for Intelligent Information Retrieval, SPAWARSYSCEN-SD grant numbers N66001-99-1-8912 and N66001-02-1-8903, Advanced Research and Development Activity under contract number MDA904-01-C-0984, and DARPA contract F30602-01-2-0566.

References

A. Borthwick, J. Sterling, E. Agichtein, and R. Grishman. 1998. Exploiting diverse knowledge sources via maximum entropy in named entity recognition. In *Proceedings of the Sixth Workshop on Very Large Corpora, Association for Computational Linguistics*.

M. Collins and Y. Singer. 1999. Unsupervised models for named entity classification. In *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*.

Stephen Della Pietra, Vincent J. Della Pietra, and John D. Lafferty. 1997. Inducing features of random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(4):380–393.

Rosie Jones, Andrew McCallum, Kamal Nigam, and Ellen Riloff. 1999. Bootstrapping for text learning tasks. In *IJCAI-99 Workshop on Text Mining: Foundations, Techniques and Applications*.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. ICML*.

Robert Malouf. 2002. A comparison of algorithms for maximum entropy parameter estimation. In *Sixth Workshop on Computational Language Learning (CoNLL-2002)*.

Andrew McCallum and Fang-Fang Feng. 2003. Chinese word segmentation with conditional random fields and integrated domain knowledge. In *Unpublished Manuscript*.

Andrew McCallum. 2003. Efficiently inducing features of conditional random fields. In *Nineteenth Conference on Uncertainty in Artificial Intelligence (UAI03)*. (Submitted).

Adwait Ratnaparkhi. 1996. A maximum entropy model for part-of-speech tagging. In Eric Brill and Kenneth Church, editors, *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 133–142. Association for Computational Linguistics.

Fei Sha and Fernando Pereira. 2003. Shallow parsing with conditional random fields. In *Proceedings of Human Language Technology, NAACL*.