Learning a Named Entity Tagger from Gazetteers with the Partial Perceptron

Andrew Carlson
Machine Learning Department
Carnegie Mellon University
5000 Forbes Ave, Pittsburgh, PA 15213
acarlson@cs.cmu.edu

Scott Gaffney and **Flavian Vasile**
Yahoo! Labs
701 First Ave, Sunnyvale, CA 94089
{gaffney,flavian}@yahoo-inc.com

Abstract

While gazetteers can be used to perform named entity recognition through lookup-based methods, ambiguity and incomplete gazetteers lead to relatively low recall. A sequence model which uses more general features can achieve higher recall while maintaining reasonable precision, but typically requires expensive annotated training data. To circumvent the need for such training data, we bootstrap the learning of a sequence model with a gazetteer-driven labeling algorithm which only labels tokens in unlabeled data that it can label confidently. We present an algorithm, called the Partial Perceptron, for discriminatively learning the parameters of a sequence model from such partially labeled data. The algorithm is easy to implement and trains much more quickly than a state-of-the-art algorithm based on Conditional Random Fields with equivalent performance. Experimental results show that the learned model yields a substantial relative improvement in recall (77.3%) with some loss in precision (a 28.7% relative decrease) when compared to the gazetteer-driven method.

Introduction

The problem of named entity recognition is that of tagging sequences of words that represent interesting entities, such as people, places, and organizations. Such tags are useful in many applications, such as semantic role labeling, question answering, or relation extraction (Punyakanok et al. 2004; Rozenfeld and Feldman 2008).

Much previous work has studied supervised training of such taggers, where a large hand-annotated corpus is used to train a model (Bikel, Schwartz, and Weischedel 1999; Borthwick 1999; McCallum and Li 2003). Such approaches can achieve good performance, but annotating data requires a large amount of human effort. Other previous work has used lists of entities (commonly called *gazetteers*) with lookup-based methods to recognize entities (Nadeau, Turney, and Matwin 2006). These approaches can require much less effort because lists of entities can be created using automated or semi-automated techniques (Etzioni et al. 2005; Wang and Cohen 2007), but they suffer from limited recall and have problems with ambiguity, e.g, where words like 'Washington' can have several different meanings depending on the context.

In this work, we bootstrap a sequence model with no hand-annotated data by using lists of entities of each type of interest and a large collection of unlabeled text. Our method first labels tokens in unlabeled data that it can label confidently, while abstaining from labeling tokens for which it is not confident. Our method then discriminatively learns the parameters of a sequence model from such partially labeled training data. This algorithm, which we call the *Partial Perceptron*, is based on the *Structured Perceptron* (Collins 2002) but is modified to allow for unlabeled tokens in the training data. We present theoretical justifications for our modifications, showing that our algorithm retains the properties that justify the original algorithm.

Empirical results demonstrate that the learned tagger yields a substantial lift in recall (77.3%) with some relative loss in precision (28.7%), when compared to the gazetteer-based tagger. In a comparison to previous work on bootstrapping a named entity tagger, we show that it is better to use all confidently labeled tokens in a sentence to train a sequence model, rather than a small window of context around confidently recognized entities. In addition, a comparison to a CRF-based method for learning from partially labeled data shows that the Partial Perceptron achieves equivalent levels of accuracy but is several orders of magnitude faster to train.

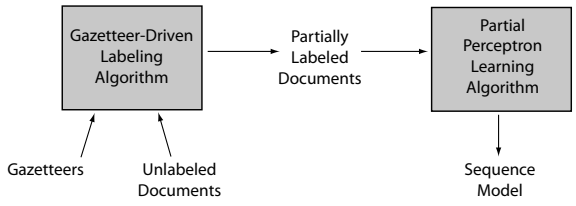

Figure 1: A high-level view of learning in our system.

Related Work

For a survey of previous work on named entity recognition, refer to Nadeau and Sekine (2007).

Correct labels	O	O	B-ORG	I-ORG	O	O	B-ORG	O	O
Partial labels	O	O	B-ORG	I-ORG	O	O	UNK	O	O
POS Tag	NNS	IN	NNP	NNP	CC	VBG	NNS	NN	NNS
Token	talks	between	Boeing	Co.	and	striking	Machinists	union	members

Figure 2: An excerpt of a sentence with true named entity labels and labels assigned after partial labeling.

Class	Gazetteer excerpt
Locations	sunderland, sundsvall, sundsvl, sunnyvale, sunrise, sunset beach, sunshine coast, suourland, suouroy
Organizations	xyratex, y e data, ya, yadkin, yahoo, yahoo maps, yahoo personal email, yahoo personals, yahoo sbc
Person Names	jerrine, jerrod, jerrol duane, jerrold, jerrome, jerry, jerry ben, jerry brown, jerry claude, jerry coleman
Other	january, february, sunday, monday, toyota prius, ford focus, apple ipod, microsoft zune, tylenol, advil

Table 1: Excerpts from gazetteers used in our experiments.

Collins and Singer (1999) presented an unsupervised approach to named entity classification which started from seven rules, used these rules to label noun phrases, used the labeled noun phrases to learn new rules, and repeated this loop in an iterative fashion. While their approach classified proper noun phrases with slightly better than 80% accuracy, it ignored all proper nouns that did not occur in specific types of informative context. Evaluated in the same fashion as traditional named entity taggers, we would expect the recall of their method to be low.

Niu et al. (2003) present an alternative method for bootstrapping named entity recognition. They start with some seed examples of entities, and learn rules to recognize additional entities. The rules are used to label unlabeled text, and the recognized entities are used to train a Hidden Markov Model. Their model uses only the context near an entity to classify it. We compare to such an approach in our experiments.

In the biomedical domain, Morgan et al. (2004) and Vlachos et al. (2006) bootstrapped a named entity tagger to recognize gene names. For a set of articles, both the abstract and a list of genes known to appear in that article were available. The gene list for each article was used to noisily annotate gene mentions in the abstract of that article. An HMM-based tagger was then trained with this noisily annotated data. For general named entity recognition, lists of entities mentioned in documents are not typically available, so this approach is not practical.

An alternative method for learning a sequence model from training data with missing labels is presented by Bellare and McCallum (2007). Their method is based on Conditional Random Fields (Lafferty, Mccallum, and Pereira 2001), and is described in detail in the 'Comparisons to Previous Work' section. We present an experimental comparison to their method later in the paper.

Recently, Whitelaw et al. used an approach similar to our work to perform web-scale named entity recognition (Whitelaw et al. 2008). Their method started with massive gazetteers for 33 different categories and generated a very large training set of trusted mentions. This set was used to train a machine learning model for classification. They report high accuracies, but their results are difficult to compare to. Their work used private corpora and gazetteers, and tested on automatically generated test data from the web. Furthermore, while the overall results that are presented are quite good (94.7% precision and 93.7% recall), the four individual category results that are presented are all worse than these numbers. For example, precision and recall for the 'company' category are only 42.6% and 62.4%, respectively, much lower than our reported results. Also, they do not compare to a gazetteer-driven approach, so it is unclear how much of an improvement is provided by their machine learned model. Our work in this paper used a publicly available set of gazetteers, and evaluated on publicly available test corpora. We demonstrate considerable improvement over a gazetteer-driven method.

Approach

Our approach is illustrated in a high-level diagram in Figure 1. Our method requires as input a collection of gazetteers, one for each named entity class of interest, and one for an 'other' class that gives examples of entities that we do not want to extract. It also takes a collection of unlabeled documents. We then apply a gazetteer-driven named entity recognition algorithm that labels the documents. The result is a high-precision, low-recall partial labeling of the tokens in the documents, where only some of the tokens are labeled. These partially labeled documents are then used to train a sequence model. The result of our approach is a model that can induce named entity labels for new documents with much higher recall than a gazetteer-driven method.

Problem Formulation

We formulate the named entity recognition problem as the task of predicting labels for the tokens in some target sentence. We assume that we are given a vector $\mathbf{x}$ that represents the tokens, part-of-speech tags, and other features of the tokens in that sentence. We must produce a label vector $\mathbf{y}$ that contains a label for each token. The labels are BIO-style labels, where 'O' means that a token is not part of an entity, 'B-TYPE' means that a token is at the beginning of an entity of type 'TYPE', and 'I-TYPE' means that a token is inside an entity of type 'TYPE'. We are given a set of entity types which we must recognize, so that 'TYPE' must refer to one of these given entity types. Figure 2 shows an example

Input: Sentence s, gazetteers $\mathcal{G}$
Output: Labels for s
$\mathcal{C} = \text{RECOGNIZECANDIDATEENTITIES}(s)$;
$\text{LABELNONENTITYTOKENS}(\text{'O'})$;
foreach *Candidate* $c \in \mathcal{C}$ **do**
  if $\text{FULLYCOVERED}(c, \mathcal{G})$ **then**
    $\text{LABELWITHGAZETTEERS}(c, \mathcal{G})$;
  else
    $\text{LABEL}(c, \text{'UNK'})$
  end
end

Algorithm 1: The gazetteer-driven partial labeling algorithm.

excerpt from a sentence with its tokens, part-of-speech tags, correct named entity labels, and the partial labels output by our training data generation method.

Training Data Generation

This section describes the process of turning unlabeled documents into partially labeled training data for a sequence model. We first describe the input to the system, which is a set of gazetteers and a collection of unlabeled documents. We then describe the algorithm used to produce high-precision labels for those documents.

Gazetteers Gazetteers are provided to the system and assumed to contain unambiguous examples of each named entity class. Excerpts of such gazetteers are shown in Table 1. Entries in the gazetteer need not be whole entities. For example, 'jerry' and 'yang' can be separate entries and used to recognize the string 'Jerry Yang' as a person. An extra gazetteer is provided for an 'Other' named entity class, which provides additional negative examples for the classes of interest.

Unlabeled Data Unlabeled data provided to the system consists of a collection of documents that have been segmented into sentences, tokenized, and tagged with part-of-speech tags.

Partial Labeling Algorithm Our high-precision partial labeling algorithm is presented as Algorithm 1. To label a sentence s, we first recognize *candidate entities* in the sentence. Candidate entities consist of sequences of capitalized words. However, we also use statistics drawn from web text to decide if candidate entities that are connected by "connector words" (*of, for, the, and, in, and &*) should be joined into a single candidate by using an independence test with a manually chosen threshold. For example, "United States of America" should be one candidate entity, but "Europe and Asia" should be two. All tokens that are not part of candidate entities are labeled with 'O' labels, indicating non-entity tokens. Next, we consider each candidate entity. The FULLYCOVERED function tests whether or not all tokens in a candidate entity can be labeled using the gazetteers. If so, the candidate entity is labeled according to the classes in the gazetteers. If any tokens cannot be labeled with gazetteer matches, then the whole candidate is marked with 'UNK' labels, indicating that we do not know the label of those tokens.

Input: Training examples $(\mathbf{x}_i, \mathbf{y}_i^{partial})$
Output: Parameters $\boldsymbol{\alpha}$
$\boldsymbol{\alpha} = 0$;
for $j = 1 \ldots T$ **do**
  $\text{SHUFFLEEXAMPLES}(\mathbf{x})$;
  for $i = 1 \ldots n$ **do**
    $\mathbf{z}_i = \arg\max_{\mathbf{z} \in \text{GEN}(\mathbf{x}_i)} \Phi(\mathbf{x}_i, \mathbf{z}) \cdot \boldsymbol{\alpha}$;
    if $(\neg \text{AGREEONKNOWNLABELS}(\mathbf{z}_i, \mathbf{y}_i^{partial}))$ **then**
      $\boldsymbol{\alpha} = \boldsymbol{\alpha} +$
      $\Phi_{OBS}(\mathbf{x}_i, \mathbf{y}_i^{partial}, \text{KNOWN}(\mathbf{y}_i^{partial})) -$
      $\Phi_{OBS}(\mathbf{x}_i, \mathbf{z}_i, \text{KNOWN}(\mathbf{y}_i^{partial}))$;
    end
  end
end

Algorithm 2: The Partial Perceptron algorithm.

In addition to this algorithm, we employ some heuristics with the aim of increasing the precision of our partial labels. First, if the first token in a sentence is not used capitalized elsewhere in the article, it is always labeled as a non-entity token. This heuristic is inspired by Nadeau et al. (2006). Second, if a single token is matched as a person name, it is still marked as unknown. Many person names, when used alone, can be ambiguous. For example, 'Merrill' will often refer to 'Merrill Lynch,' the financial firm. Finally, person name titles like 'Mrs.' and 'Dr.' are considered non-entity tokens to match conventions of our test sets.

To increase the recall of our labels, we employ an 'alias resolution' heuristic. Shorter substrings of labeled phrases are given the same label. For example, if 'George Washington' is recognized as a person in a document, 'George' or 'Washington' would also be matched as a person. This method was also inspired by Nadeau et al. (2006).

Learning the Sequence Model

We formulate the sequence model learning task as follows: given training data consisting of $(\mathbf{x}_i, \mathbf{y}_i^{partial})$ pairs, learn a model that produces the correct label vector $\mathbf{y}$ for a new observation vector $\mathbf{x}$. In the training data, each $\mathbf{x}_i$ is a vector of tokens and part-of-speech tag observations, and each $\mathbf{y}_i^{partial}$ gives labels for those tokens. The labels might be special 'UNK' labels indicating that the true label for that token is unknown.

We assume a local feature representation ϕ that generates a d-dimensional Boolean vector, given the observed features for a token, a label for that token, and a label for the previous token. ϕ_i refers to the ith entry in the vector. For example, $\phi_{1000}(y, y_{prev}, x)$ might be 1 if the word represented by x is 'Frank' and the label y is 'B-PER' and 0 otherwise.

The local feature representation is used in a global feature representation Φ. Φ generates a d-dimensional feature vector for a sequence of observations $\mathbf{x}$ and a corresponding sequence of labels $\mathbf{y}$ by summing over each position in the

sequence:

$$\Phi(\mathbf{x}, \mathbf{y}) = \sum_{j=1}^{|\mathbf{y}|} \phi(y_{j-1}, y_j, x)$$

Our learning task is to learn a d-dimensional weight vector $\boldsymbol{\alpha}$ that we use to produce a label vector $\mathbf{y}$ for an input vector $\mathbf{x}$:

$$\hat{\mathbf{y}}(\mathbf{x}) = \arg\max_{\mathbf{y} \in \text{GEN}(\mathbf{x})} \Phi(\mathbf{x}, \mathbf{y}) \cdot \boldsymbol{\alpha}$$

where $\text{GEN}(\mathbf{x})$ is the set of possible labelings of the sequence represented by $\mathbf{x}$.

To learn from partially labeled training data, we use an alternative feature representation, Φ_{OBS}. Informally, Φ_{OBS} only sums over local features ϕ_{OBS} which do not depend on any unknown labels. Φ_{OBS} takes an additional Boolean vector $\mathbf{k}$ as a parameter, which indicates which labels in $\mathbf{y}$ are known. k_j is 1 if the value of y_j is known, and 0 otherwise.

$$\Phi^i_{OBS}(\mathbf{x}, \mathbf{y}, \mathbf{k}) = \sum_{j=1}^{|\mathbf{y}|} \phi^i_{OBS}(y_{j-1}, y_j, x, k_{j-1}, k_j)$$

$\phi^i_{OBS}(y_{j-1}, y_j, x, k_{j-1}, k_j)$ returns 0 if it depends on the values of y_{j-1} and y_j and at least one of k_{j-1} or k_j are 0 (for transition features), or if it depends on the value of y_j and k_j is 0 (for observation features). In these cases, at least one of the labels that the local feature depends on is unknown, so the correct value of that feature is unknown. Otherwise, it returns the value of $\phi_i(y_{j-1}, y_j, x)$.

The Partial Perceptron, our algorithm for learning $\boldsymbol{\alpha}$ from partially labeled examples, is shown in Algorithm 2. We are given a collection of partially labeled training examples that consist of $(\mathbf{x}_i, \mathbf{y}_i^{partial})$ pairs. We define a function $\text{KNOWN}(\mathbf{y})$ which returns a binary vector of the same length as $\mathbf{y}$ where the ith entry in that vector is 1 if the value of y_i is known and 0 otherwise. We initialize the weight vector $\boldsymbol{\alpha}$ to 0. We loop T times over the training examples, randomly shuffling their order each time. For each training example, we find the labeling $\mathbf{z}_i$ which maximizes the score $\Phi(\mathbf{x}_i, \mathbf{z}) \cdot \boldsymbol{\alpha}$. If $\mathbf{z}_i$ has labels that disagree with known labels in $\mathbf{y}_i^{partial}$ (determined by the AGREEONKNOWNLABELS function), we update $\boldsymbol{\alpha}$ to make the incorrect predictions less likely and the correct predictions more likely.

The original Structured Perceptron algorithm (due to Collins (2002)) would add $\Phi(\mathbf{x}_i, \mathbf{y}_i)$ to $\boldsymbol{\alpha}$ and subtract $\Phi(\mathbf{x}_i, \mathbf{z}_i)$, but we do not know the correct values for all labels in $\mathbf{y}_i$. Instead, we only update using features where we observed the correct value using $\Phi_{OBS}(\mathbf{x}, \mathbf{y}, \text{KNOWN}(\mathbf{y}))$.

We based our method on the Structured Perceptron because it is inexpensive to train, has been shown to be competitive with other state-of-the-art learning methods (Sha and Pereira 2003), and has desirable theoretical properties.

Similar to Collins, we use an averaged version of this learning algorithm where the parameters are averaged across all iterations and examples. This helps to combat overfitting in the common case where data is not perfectly separable.

A later section presents a theoretical justification for a close variant of the Partial Perceptron, showing that it retains the same theoretical properties that justify the Structured Perceptron algorithm.

Tagging Named Entities in Test Documents

Once we have learned a sequence model, we can apply it to new test documents to induce named entity labels for all of the tokens in the target documents. Each sentence $\mathbf{x}$ can be fully labeled by finding the label sequence which maximizes the $\boldsymbol{\alpha} \cdot \Phi(\mathbf{x}, \mathbf{y})$ score, using the Viterbi algorithm.

Evaluation

In our evaluation, we sought to answer the following questions:

1. How accurate are the gazetteer-driven labels?
2. Does our Partial Perceptron algorithm learn a sequence model which improves on the gazetteer-driven method?
3. How does the model learned by the Partial Perceptron compare to alternative methods for learning a model from partially labeled data?

Experimental Methodology

Unlabeled Data We collected the bodies of Yahoo! News articles from 16 March 2008 to 18 May 2008. Articles were split into sentences, tokenized, and POS-tagged with the OpenNLP package[1]. Poorly formed sentences were filtered out (those consisting of all capital letters, mostly punctuation, or having no verbs). These unlabeled sentences were then tagged with our partial labeling algorithm. Sentences that had no recognized entities were discarded.

Gazetteers We used the gazetteers included in version 1.81 of the BALIE system (Nadeau 2005). The BALIE system is open source and available for download[2]. These gazetteers are large but noisy because they were generated using a bootstrapping approach that started with a small seed list for each category and expanded that list using wrapper induction from web pages. Nadeau estimated that they are roughly 90% precise. Our 'organization' gazetteer was created by merging the 'association', 'company', 'military', 'company designator' and 'military' gazetteers included with the BALIE system. Similarly, our 'person' gazetteer merged the 'first name', 'celebrity', and 'last name' BALIE gazetteers, and our 'location' gazetteer merged the 'state/province', 'city', and 'country' BALIE gazetteers. Any ambiguous entries were removed. For example, 'Charlotte' occurred in both the 'location' and 'person' gazetteers, and was thus removed from both. Refer to Table 1 for examples of entries in the gazetteers.

To provide examples of entities that were not members of any of our three target classes, we included lists of days of

[1] Available at `http://opennlp.sourceforge.net`

[2] Available at `http://balie.sourceforge.net/`

the week, months, and nationalities. Occurrences of these entries were marked as 'O', meaning non-entity tokens.

After experimentation with development data, we determined that a small amount of effort in removing incorrect gazetteer entries and providing additional negative examples would provide significantly cleaner partially labeled data. One of the authors spent one hour removing noisy entries from the gazetteers, and one hour adding new negative examples. This process was driven by looking at mistakes in development data. For example, 'Vietnam War' and 'American Revolution' were removed from the 'person' gazetteer, and 'Accord' and 'Congressional Medal of Honor' were added as extra negative entities.

Test Corpora We used three test sets: the development and test portions of the Wall Street Journal articles in the Penn Treebank, annotated with entity type labels (Weischedel and Brunstein 2005), and the CoNLL 2003 Shared Task English portion (Erik and De Meulder 2003), which consists of news articles from Reuters. In all cases we used the 'person', 'location', and 'organization' annotations, and discarded the rest. For the WSJ corpora, this required a simple mapping from the finer-grained classes to our three higher level classes. We used the POS tags provided in the data sets.

Evaluation Metrics We measured performance using the traditional metrics of precision, recall, and F1. We used the `conlleval`[3] script to calculate these metrics. This script requires exact matches of entities, and gives no credit unless the boundaries of an entity are correctly matched.

Features In our model, we used the following features of the current token, the previous token, and the following token:

1. **tokens**: The token
2. **pos tags**: The part-of-speech tag of the token
3. **shape**: The capitalization pattern of the token, for example "XxXx" for 'McDonald'
4. **prefix**: Two- and three-character token prefixes
5. **suffix**: Two- and three-character token suffixes

We also used a feature that was the predicted label of the previous token, which provided state transition features.

Following standard practice in structured output problems, these features were used in the feature representation mapping $\Phi(\mathbf{x}, \mathbf{y})$ by generating a binary indicator for every observed value of a feature conjoined with every possible label for a token.

Implementation of Partial Perceptron We based our implementation of the Partial Perceptron off of existing open source software, SuperSenseTagger, which is available for download online[4] and is described by Ciaramita and Altun (2006). We used the default feature extraction code provided with the package.

[3] Distributed with the CoNLL 2003 data

[4] `http://sourceforge.net/projects/supersensetag`

Comparisons to Previous Work We compared our approach to two other sequence modeling candidates: *Local Context Bootstrapping*, and the *Partial CRF*. The Local Context Bootstrapping method as defined here was inspired by Niu et al. (2003). This involves learning a sequence model for subsequences that surround proper noun phrases. For each entity in the partially labeled data, we extracted a training sequence of that entity phrase with two words of context to the left and right. We trained a Structured Perceptron model on these sequences. To label test data in the same manner as the previous work, we extract all proper noun phrases (tagged NNP or NNPS) with two context words to the left and right, and label these subsequences. This provides a comparison that shows whether or not learning our model from entire partially labeled sentences improves performance, as compared to learning from only words near entities.

The Partial CRF is based on the adapted CRF models of Bellare and McCallum (2007). Their work adapted traditional Conditional Random Fields to learn from training data where not all tokens have labels. Their method for training marginalized out the unknown labels in the computation of the gradient. This effectively learns parameters that maximize the conditional likelihood of the observed labels. This work provides us with a comparison with an algorithm that fills in missing labels during training, as opposed to our method which ignores the missing labels. In our experiments with Partial CRF, we used software provided by the original authors. We also tried software from the authors that used stochastic gradient descent to find parameters, but we found that the optimization procedure was highly unstable. Correcting this could yield a faster training procedure for this method.

Experimental Results

The top row of Table 2 shows the performance of the gazetteer-based partial labeling method when applied to the test sets. This gives an indication of the quality of the training data that was then used to train our sequence models. As expected, the gazetteer-based method has high precision (83-93%) but relatively low recall (27-46%).

The partial labeling results produced with the unedited gazetteers had 1-2% lower precision and 0-1% better recall on the test corpora.

The bottom two rows of Table 2 show the performance of the Local Context Bootstrapping method and the Partial Perceptron method. The results are averaged across five random training samples of 128,000 sentences from our partially labeled news articles. Each method was trained with three iterations through the data. Both methods give a lift in recall over the gazetteer-based extraction method with some loss in precision. On average, the Partial Perceptron gives an 77.3% relative improvement in recall, with a 28.7% relative drop in precision over the gazetteer-driven labeling. Compared to Local Context Bootstrapping, the Partial Perceptron gives an average 16.1% relative improvement in recall and a 2.2% relative drop in precision.

The sequence models learned by the Partial Perceptron when using the unedited gazetteers had negligibly lower re-

	Test Data Set								
	WSJ Dev			WSJ Test			CoNLL		
Method	P	R	F1	P	R	F1	P	R	F1
Gazetteer-based	90.6	40.7	56.2	92.9	45.6	61.1	83.1	27.0	40.7
Local context bootstrapping	66.4	62.9	64.6	66.8	62.9	64.8	61.2	47.2	53.3
Partial Perceptron	66.0	73.2	69.4	67.0	74.0	70.3	57.1	53.7	55.3

Table 2: Results for gazetteer-based labeling method, bootstrapping with local context around entities, and bootstrapping with partially labeled sentences with the Partial Perceptron.

Method	Training time (m)	P	R	F1
Partial CRF	717.0	64.9	71.9	68.2
Partial Perceptron	1.4	65.4	72.0	68.5

Table 3: Results for Partial CRF and Partial Perceptron on the WSJ Development data set. Both models were trained on random samples of 25600 partially labeled training sentences. Results are averaged across three samples.

call (0 - 0.5%) with higher precision (2.5-3.9%). Since two hours of work editing the gazetteers produced significantly higher precision, we suspect that further time editing the gazetteers would result in further improvements.

Table 3 compares our Partial Perceptron method with the Partial CRF method. Both methods were trained on 25,600 sentences of partially labeled news articles and evaluated on the WSJ Development corpus. The results are averaged across three random samples of training data. The methods give equivalent performance, but there is a large difference in training time. The Partial CRF takes 717 minutes to train to convergence, while the Partial Perceptron takes less than 2 minutes. We verified that on a typical run, the Partial CRF asymptotes with respect to test set performance about halfway through this time, so some time saving could be achieved using a held-out validation set, but the training time difference would still be two orders of magnitude. These results support our intuition that the Partial Perceptron provides a fast method for learning accurate models from partially labeled data.

Theoretical Justification for our Algorithm

Convergence and generalization results for the Perceptron applied to classification were given by Freund and Schapire (1999). These results were later adapted to structured output problems such as tagging by Collins (2002). In this section, we informally summarize these results and explain how to adapt them to apply to the Partial Perceptron algorithm.

The first result is that if the Perceptron is presented with a sequence of training examples that is separable with margin δ, then the Perceptron will make a number of mistakes inversely proportional to δ^2 on those training examples.

If a training sequence is not separable (that is, there is no weight vector that perfectly classifies it), we can measure how close that sequence is to being separable with margin δ by penalizing margin violations with slack terms. The second result is that one can relate the number of mistakes that the Perceptron will make in learning from that data to how close that data is to being separable.

The final result is related to how well a voted version of the Perceptron (called the *Voted Perceptron*) generalizes to unseen test examples. It states that if a training set of examples and a single test example are drawn i.i.d. from some probability distribution, the probability of making an error on the test example is related to the separability of the training and test data. If, according to our probability distribution over examples, we expect the training examples and the test examples to be close to separable, we expect to have a higher probability of correctly classifying the test example than if the examples were far from separable. In practice, the average values of the weights during training are used as an approximation to the Voted Perceptron.

Consider a slight variant of Algorithm 2 where we predict the $\arg\max$ label sequence using the restricted feature representation Φ_{OBS} rather that the full feature representation Φ. All results and proofs in the work by Collins apply to this variant with the difference that the feature representation mapping Φ is replaced by Φ_{OBS} (defined previously) that only sums over features for which the corresponding correct labels are known, and that the comparison that decides whether or not a mistake is made considers only the known labels. In Collins' setting, the theoretical results concerning a set of $(\mathbf{x}_i, \mathbf{y}_i)$ examples consider the separability of those examples after they have been mapped to a d-dimensional space using $\Phi(\mathbf{x}, \mathbf{y})$. In our setting, we consider the separability of a set of $(\mathbf{x}_i, \mathbf{y}_i^{partial})$ pairs in the d-dimensional space using $\Phi_{OBS}(\mathbf{x}_i, \mathbf{y}_i^{partial}, \textsc{Known}(\mathbf{y}_i^{partial}))$.

Conclusion

We have shown that an accurate named entity tagger can be learned using only gazetteers and unlabeled data, resources that are much less costly than hand-annotated training text. We presented a high-precision gazetteer-driven partial labeling algorithm, and a learning algorithm, called the Partial Perceptron, which learns a sequence model from partially labeled data. The Partial Perceptron learns models with much higher recall than a gazetteer-driven method with some loss in precision. Also, the Partial Perceptron outperforms a bootstrapping method that learns using local context around entities, and learns models that are just as accurate as

a CRF-based method, but is much faster to train. The Partial Perceptron also retains the desirable theoretical properties of the Perceptron.

Acknowledgments

Much of this work was done while the primary author was an intern with Yahoo!. The primary author gratefully acknowledges support from a Yahoo! Fellowship. The authors wish to acknowledge Kedar Bellare for providing his PartialCRF code, and Massi Ciaramita for help with the SuperSenseTagger code.

References

Bellare, K., and McCallum, A. 2007. Learning extractors from unlabeled text using relevant databases. In *Proceedings of IIWeb-AAAI'07 Workshop*.

Bikel, D. M.; Schwartz, R.; and Weischedel, R. M. 1999. An algorithm that learns what's in a name. *Machine Learning* 34(1-3):211–231.

Borthwick, A. E. 1999. *A maximum entropy approach to named entity recognition*. Ph.D. Dissertation, New York University, New York, NY, USA.

Ciaramita, M., and Altun, Y. 2006. Broad-coverage sense disambiguation and information extraction with a supersense sequence tagger. In *Proceedings of EMNLP*.

Collins, M., and Singer, Y. 1999. Unsupervised models for named entity classification. In *Proceedings of EMNLP*.

Collins, M. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of EMNLP*.

Erik, and De Meulder, F. 2003. Introduction to the conll-2003 shared task: language-independent named entity recognition. In *Proceedings of CoNLL*.

Etzioni, O.; Cafarella, M. J.; Downey, D.; Popescu, A.; Shaked, T.; Soderland, S.; Weld, D. S.; and Yates, A. 2005. Unsupervised named-entity extraction from the web: An experimental study. *Artificial Intelligence* 165(1):91–134.

Freund, Y., and Schapire, R. E. 1999. Large margin classification using the perceptron algorithm. In *Machine Learning*, 277–296.

Lafferty, J.; Mccallum, A.; and Pereira, F. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of ICML*.

McCallum, A., and Li, W. 2003. Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. In *Proceedings of CoNLL*.

Morgan, A. A.; Hirschman, L.; Colosimo, M.; Yeh, A. S.; and Colombe, J. B. 2004. Gene name identification and normalization using a model organism database. *J. of Biomedical Informatics* 37(6):396–410.

Nadeau, D., and Sekine, S. 2007. A survey of named entity recognition and classification. *Journal of Linguisticae Investigationes* 30(1).

Nadeau, D.; Turney, P. D.; and Matwin, S. 2006. Unsupervised named-entity recognition: Generating gazetteers and resolving ambiguity. In *Proceedings of CCAI*.

Nadeau, D. 2005. Balie - baseline information extraction : Multilingual information extraction from text with machine learning and natural language techniques. Technical report, University of Ottawa.

Niu, C.; Li, W.; Ding, J.; and Srihari, R. K. 2003. A bootstrapping approach to named entity classification using successive learners. In *Proceeding of ACL*.

Punyakanok, V.; Roth, D.; tau Yih, W.; and Zimak, D. 2004. Semantic role labeling via integer linear programming inference. In *Proceedings of COLING*.

Rozenfeld, B., and Feldman, R. 2008. Self-supervised relation extraction from the web. *Knowl. Inf. Syst.* 17(1):17–33.

Sha, F., and Pereira, F. 2003. Shallow parsing with conditional random fields. In *Proceedings of NAACL-HLT*.

Vlachos, A.; Gasperin, C.; Lewin, I.; and Briscoe, T. 2006. Bootstrapping the recognition and anaphoric linking of named entities in drosophila articles. *Pac Symp Biocomput* 100–111.

Wang, R. C., and Cohen, W. W. 2007. Language-independent set expansion of named entities using the web. In *Proceedings of ICDM*.

Weischedel, R., and Brunstein, A. 2005. Bbn pronoun coreference and entity type corpus. Linguistic Data Consortium, Philadelphia. LDC2005T13.

Whitelaw, C.; Kehlenbeck, A.; Petrovic, N.; and Ungar, L. 2008. Web-scale named entity recognition. In *Proceedings of CIKM*.