Learning to Extract Information from Semi-structured Text using a Discriminative Context Free Grammar

Draft submitted to the conference SIGIR 2005 *

Paul Viola
viola@microsoft.com

Mukund Narasimhand
a-muknar@microsoft.com

ABSTRACT

In recent work, conditional Markov chain models (CMM) have been used to extract information from semi-structured text (one example is the Conditional Random Field [10]). Applications range from finding the author and title in research papers to finding the phone number and street address in a web page. The CMM framework combines a priori knowledge encoded as features with a set of labeled training data to learn an efficient extraction process. We will show that similar problems can be solved more effectively by learning a discriminative context free grammar from training data. The grammar has several distinct advantages: long range, even global, constraints can be used to disambiguate entity labels; training data is used more efficiently; and a set of new more powerful features can be introduced. The specific problem we consider is of extracting personal contact, or address, information from unstructured sources such as documents and emails. While linear-chain CMMs perform reasonably well on this task, we show that a statistical parsing approach results in a 50% reduction in error rate. This system also has the advantage of being interactive, similar to the system described in [9]. In cases where there are multiple errors, a single user correction can be propagated to correct multiple errors automatically. Using a discriminatively trained grammar, 93.71% of all tokens are labeled correctly (compared to 88.43% for a CMM) and 72.87% of records have *all* tokens labeled correctly (compared to 45.29% for the CMM).

Categories and Subject Descriptors

H.4.0 [**Information Systems and Applications**]: [General]; H.3.3 [**Information Storage and Retrieval**]: [Information search and retrieval]; H.3.5 [**Information Storage and Retrieval**]: [online information services]

General Terms

Algorithms, Experimentation

Keywords

Text Tagging, Discriminative Models, Discriminative Grammars, Conditional Random Fields, Perceptron Training, Information Retrieval

*Contact `viola@microsoft.com` for more information

1. INTRODUCTION

In this paper, we consider the problem of automatically populating forms and databases with information that is available in an electronic but *unstructured* format. While there has been a rapid growth of online and other computer accessible information, little of this information has been schematized and entered into databases so that it can be searched, integrated and reused. For example, a recent study shows that as part of the process of gathering and managing information, currently 70 million workers, or 59% of working adults in the U.S., complete forms on a regular basis as part of their job responsibilities.

One common example is the entry of customer information into an online customer relation management system. In many cases customer information is already available in an unstructured form on web sites and in email. The challenge is in converting this semi-structured information into the regularized or schematized form required by a database system. There are many related examples including the importation of bibliography references from research papers and extraction of resume information from job applications. For the applications considered in this paper, the source of the semi-structured information is "raw text". The same approach can be extended to work with semi-structured information derived from scanned documents (image based information) or voice recordings (audio based information).

Contact information appears routinely in the signature of emails, on web pages, and on fax cover sheets. The form of this information varies quite a bit; from a simple name and phone number to a complex multi-line block containing addresses, multiple phone numbers, emails, and web pages. Effective search and reuse of this information requires field extraction such as LastName, FirstName, StreetAddress, City, State, PostalCode, HomePhoneNumber etc. One way of doing this is to consider the text block as a sequence of words/tokens, and assign labels (fields of the database) to each of these tokens (see Figure 1). All the tokens corresponding to a particular label are then entered into the corresponding field of the database. In this simple

way a token classification algorithm can be used to perform schematization. Common approaches for classification include maximum entropy models and Markov models.

We present a classification algorithm based on discriminatively trained context free grammars (CFG) that significantly outperforms prior approaches. Besides achieving substantially higher accuracy rates, we show that a CFG based approach is better able to incorporate expert knowledge (such as the structure of the database or form), less likely to be overtrained, and is more robust to variations in the tokenization algorithm.

2. PREVIOUS WORK

Free-form contact information such as that found on web pages, emails and documents typically does not follow a rigid format, even though it often follows some conventions. The lack of a rigid format makes it hard to build a non-statistical system to recognize and extract various fields from this semi-structured data. Such a non-statistical system might be built for example by using regular expressions and lexicon lists to recognize fields. One such system is described in [20]. This system looks for individual fields such as phone numbers by matching regular expressions, and recognizing other fields by the presence of keywords such as "Fax", "Researcher", etc., and by their relative position within the block (for example, it looks in the beginning for a name). However, because of spelling (or optical character recognition) errors and incomplete lexicon lists, even the best of deterministic systems are relatively inflexible, and hence break rather easily. Further, there is no obvious way for these systems to incorporate and propagate user input or to estimate confidences in the labels. For these reasons, it makes sense to consider a statistical approach to the problem of extracting information from semi-structured sources.

A simple statistical approach might be to use a Naïve Bayes classifier to classify (label) each word individually. However such classifiers have difficulties using features which are not independent. Maximum entropy classifiers ([15]) can use arbitrarily complex, possibly dependent features, and tend to significantly outperform Naïve Bayes classifiers when there is sufficient data. However, a common weakness of both these approaches is that each word is classified independently of all others. Because of this, dependencies between labels cannot be used for classification purposes. To see that label dependencies can help improve recognition, consider the problem of assigning labels to the word sequence "GREWTER JONES". The correct label sequence is FirstName LastName. Because GREWTER is an unusual name, classifying it in isolation is difficult. But since JONES is very likely to be a LastName, this can be used to infer that GREWTER is probably a FirstName. Thus, a Markov dependency between the labels can be used to disambiguate the first token.

Markov models explicitly capture the dependencies between the labels. A Hidden Markov Model (HMM) [17] models the labels as the states of a Markov chain, with each token a probabilistic function of the corresponding label. A first order Markov chain models dependencies between the labels corresponding to adjacent tokens. While it is possible to use higher order Markov models, they are typically not used in practice because such models require much more data (as there are more parameters to estimate), and require more computational resources for learning and inference. A drawback of HMM based approaches is that the features used must be independent, and hence complex features (of more than one token) cannot be used. Some papers exploring these approaches include [1, 2, 4, 3, 18, 19].

A Conditional Markov Model (CMM) ([10, 5, 21]) is a discriminative model that is a generalization of both maximum entropy models and HMMs. Formally, they are undirected graphical models used to compute the joint score (sometimes as a conditional probability) of a set of nodes designated as hidden nodes given the values of the remaining nodes (designated as observed nodes). The observed nodes correspond to the tokens, while the hidden nodes correspond to the (unknown) labels corresponding to the tokens. As in the case of HMMs, the hidden nodes are sequentially ordered, with one link between successive hidden nodes. While a HMM model is generative, the conditional Markov model is discriminative. The conditional Markov model defines the joint score of the hidden nodes *given* the observed nodes. This provides the flexibility to use complex features which can be a function of any or all of the observed nodes, rather than just the observed node corresponding to the hidden node. Like the Maximum Entropy models the conditional Markov model uses complex features. Like the HMM the CMM can model dependencies between labels. In principle a CMMs can model third or fourth order dependencies between labels though most published papers use first order models because of data and computational restrictions.

Variants of conditional Markov models include Conditional Random Fields (CRFs) [10], voted perceptron models [5], and max-margin Markov models [21]. CRFs are the most mature and have shown to perform extremely well on information extraction tasks ([14, 16, 15, 13, 19]). A CRF model is used in [9] to label tokens corresponding to contact blocks, to achieve significantly better results than prior approaches to this problem.

3. GOING BEYOND CHAIN-MODELS

While CMMs can be very effective, there are clear limitations that arise from the "Markov" assumption. For example, a single "unexpected" state/label can throw the model off. Further, these models are incapable of encoding some types of complex relationships and constraints. For example, in a contact block, it may be quite reasonable to expect only one city name. However, since a Markov model can only encode constraints between adjacent labels, constraints on labels that are separated by a distance of more than one cannot be easily encoded without an explosion in the number of states (possible values of labels), which then complicates learning and decoding.

```
Fred Jones                  Boston College
10 Main St.                 10 Main St.
Cambridge, MA 02146         Cambridge MA 02146
(425) 994-8021              (425) 994-8021
```

Figure 2: Disambiguation of Phone Numbers

Modeling non-local constraints is very useful, for example, in the disambiguation of business phone numbers and personal phone numbers. To see this, consider the two contact blocks shown in Figure 2. In the first case, it is natural to label the phone number as a HomePhoneNumber. In the second case, it is more natural to label the phone

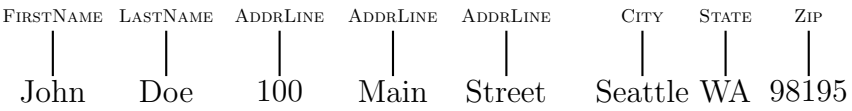

Figure 1: Assigning labels to unstructured text

number as a BusinessPhoneNumber. Humans tend to use the labels/tokens near the beginning to distinguish the two. Therefore, the label of the last token depends on the label of the first token. There is no simple way of encoding this very long-range dependence with any practical Markov model.

A grammar based model allows us to "escape the linear tyranny of these n-gram models and HMM tagging models" ([11]). A context-free grammar allows specification of more complex structure with long-range dependencies, while still allowing for relatively efficient labeling and learning from labeled data. One possible way to encode the long-range dependence required for the above example might be to use a grammar which contains different productions for business contacts, and personal contacts. The presence of the productions BizContact → BizName Address BizPhone and PersonalContact → Name Address HomePhone would allow the system to infer that the phone number in the first block is more likely to be a HomePhone while the phone number in the second is more likely to be a BusinessPhone. The correct/optimal parse of the blocks automatically takes the long-range dependencies into account naturally and efficiently.

As another example imagine a system which has a detailed database of city and zip code relationships. Given a badly misspelled city name, there may be many potential explanations (such as a first name or company name). If the address block contains an unambiguous zip code, this might provide the information necessary to realize that "Noo Yick" is actually the city "New York". This becomes especially important if there is some ambiguity with regards to the tokens themselves (which might occur for example if the tokens are outputs of a speech recognition system, or a image based system). Therefore, if the name of the city is misspelled, or incorrectly recognized, we can use the presence of an unambiguous zip code to make better predictions about the city. In a simple linear-chain Markov model, if the state appears between the city and the zip, the dependence between the zip and the city is lost.

Labeling using CMM's has been used as an approximation to, and as an intermediate step in, many important shallow parsing problems including NP-chunking. While CMMs achieve reasonably good accuracy, the accuracy provided by a full blown statistical parser is often higher. The main advantage of a CMM is computational speed and simplicity. We argue that it is more natural to model a contact block using a CFG than a CMM. This is because a contact block is more than just a sequence of words. There is clearly some hierarchical structure to the block. For example, the bigram FirstName LastName can be recognized as a Name as can LastName, FirstName. Similarly, a Address can be of the form StreetAddress, City State Zip and also of the form StreetAddress. It intuitively makes sense that these different forms occur (with different probabilities) independently of their context. While this is clearly an approximation to the reality, it is perhaps a better approximation than the Markov assumption underlying chain-models.

The grammatical parser accepts a sequence of tokens, and returns the optimal (lowest cost or highest probability) parse tree corresponding to the tokens. Figure 3 shows a parse tree for the sequence of tokens shown in Figure 1. The leaves of the parse tree are the tokens. Each leaf has exactly one parent, and parents of the leaves are the labels of the leaves. Therefore, going from a parse tree to the label sequence is very straightforward. Note that the parse tree represents a hierarchical structure beyond the labels. This hierarchy is not artificially imposed, but rather occurs naturally. Just like a language model, the substructure Name and Address can be arranged in different orders : both Name Address and Address Name are valid examples of a contact block. The reuse of components allows the grammar based approach to more efficiently generalize from limited data than than a linear-chain based model. This hierarchical structure is also useful when populating forms with more than one field corresponding to a single label. For example, a contact could have multiple addresses. The hierarchical structure allows a sequence of tokens to be aggregated into a single address, so that different addresses could be entered into different fields.

4. DISCRIMINATIVE CONTEXT-FREE GRAMMARS

A context free grammar (CFG) consists of a set of terminals $\{w^k\}_{k=1}^{V}$, a set of nonterminals $\{N^j\}_{i=1}^{n}$, a designated start symbol N^1, and a set of rules or productions $\{R_i : N^{j_i} \rightarrow \zeta^i\}_{i=1}^{r}$ where ζ^i is a sequence of terminals and nonterminals. We associate a score $S(R_i)$ with each rule R_i. A parse tree is a tree whose leaves are labeled by terminals and interior nodes are labeled by nonterminals. Further if a node N^{j_i} is the label of a interior node, then the child nodes are the terminals/nonterminals in ζ^i where $R_i : N^{j_i} \rightarrow \zeta^j$. The score of a parse tree T is given by $\sum_{\{R_i:N^{j_i}\rightarrow\zeta^i\}\in T} S\left(N^{j_i} \rightarrow \zeta^i\right)$. A parse tree for a sequence $w_1 w_2 \ldots w_m$ is a parse tree whose leaves are $w_1 w_2 \ldots w_m$. Given the scores associated with all the rules, and a given sequence of terminals $w_1 w_2 \ldots w_m$, the CKY algorithm can compute the highest scoring parse tree in time $O(m^3 \cdot n \cdot r)$, which is reasonably efficient when m is relatively small.

Generative models such as probabilistic CFGs can be described using this formulation by taking $S(R_i)$ to be the logarithm of the probability $P(R_i)$ associated with the rule. If the probability $P(R_i)$ is a log-linear model and N^{j_i} can be derived from the sequence $w_a w_{a+1} \ldots w_b$ (also denoted $N^{j_i} \stackrel{*}{\Rightarrow} w_a w_{a+1} \ldots w_b$), then $P(R_i)$ can be written as

$$\frac{1}{Z_{(\lambda(R_i),a,b,R_i)}} \exp \sum_{k=1}^{F} \lambda_k(R_i) f_k(w_a, w_{a+1}, \ldots, w_b, R_i)$$

$\{f_k\}_{k=1}^{F}$ is the set of features and $\lambda(R_i)$ is a vector of parameters representing feature weights (possibly chosen by training). $Z_{\left(\lambda,a,b,N^j\rightarrow\zeta^j\right)}$ is called the partition function and

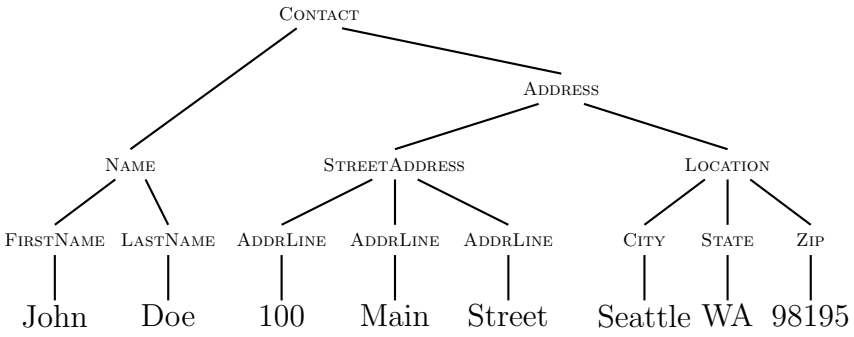

Figure 3: An example parse of a contact block

is chosen to ensure that the probabilities add up to 1.

In order to learn an accurate generative model, a lot of effort has to be spent learning the distribution of the generated leaf sequences. Since the set of possible leaf sequences are very large, this requires a large amount of training data. However, in the applications of interest, the leaves are typically fixed, and we are only interested in the conditional distribution of the rest of the parse tree given the leaves. Therefore, if we set out to only learn the conditional distribution (or scores) of the parse trees given the leaves, we can potentially manage with considerably less data (and less computational effort).

A similar observation has been made in the machine learning community. Many of the modern approaches for classification are discriminative (e.g. Support Vector Machines [6] and AdaBoost [8]). These techniques typically generalize better than generative techniques because they only model the boundary between classes (which is closely related to the conditional distribution of the class label), rather than the joint distribution of class label and observation.

A generative model defines a language, and associates probabilities with each sentence in the language. In contrast a discriminative model only associates scores with the different parses of a particular sequence of terminals. Computationally there is little difference between the generative and discriminative model - the complexity for finding the optimal parse tree (the inference problem) is identical in both cases. For our generative model, the scores associated with the rule $R_i : N^{j_i} \rightarrow \zeta_i$ is given by

$$S(R_i) = \sum_{k=1}^{F} \lambda_k(R_i) f_k(w_1 w_2 \dots w_m, a, b, R_i)$$

when applied to the sequence $w_a w_{a+1} \dots w_b$. Note that in this case the features can depend on all the tokens, not just the subsequence of tokens spanned by N^{j_i}. The discriminative model allows for a richer collection of features as we do not require independence between the features. Since a discriminative model can always use the set of features that a generative model can, there is always a discriminative model which performs at least well as the best generative model. In many experiments, discriminative models tend to outperform generative models.

4.1 Constructing a grammar

As we mentioned before, the hierarchical structure of contact blocks is not arbitrary. It is fairly natural to combine a FirstName and a LastName to come up with a Name. This leads to the rule Name → FirstName LastName. Other productions for Name include

- Name → LastName, FirstName
- Name → FirstName MiddleName LastName
- Name → FirstName NickName LastName

We can build on Name by modeling titles and suffixes using productions FullName → Name, FullName → Title Name Suffix. We can construct other rules based on commonly occurring idioms. For example, we might have Location → City State Zip. Such a grammar can be constructed by an "expert" after examining a number of examples.

Alternatively an automatic grammar induction technique may be used. In our case, we used a combination of the two. Based on a database of 1487 labeled examples of contact records drawn from a diverse collection of sources, we had a program extract commonly occurring "idioms" or patterns. A human expert then sifted through the generated patterns to decide which made sense and which did not. Most of the rules generated by the program, especially those which occurred with high frequency, made sense to the human expert. The human expert also took some other considerations into account, such as the requirement that the productions were to be binary (though the productions were automatically binarized by another program). Another requirement was imposed by training requirements mentioned in Section 4.4.

4.2 Selecting features

The features selected included easily definable functions like word count, regular expressions matching token text (like ContainsNewLine, ContainsHyphen, ContainsDigits, PhoneNumLike), tests for inclusion in lists of standard lexicons (for example, US first names, US last names, commonly occurring job titles, state names, street suffixes), etc. These features are mostly binary, and are definable with minimal effort. They are similar to those used by the CRF model described in [9]. However in the CRF model, and in all CMMs, the features can only relate the sequence of observations w_i, the current state s_t, the previous state s_{t-1}), and the current time t (i.e. $f_j(s_t, s_{t-1}, w_1, w_1, \dots, w_m, t)$).

In contrast the discriminative grammar admits additional features of the form $f_k(w_1, w_1, \dots, w_m, a, b, c, N^{j_i} \rightarrow \zeta^i)$,

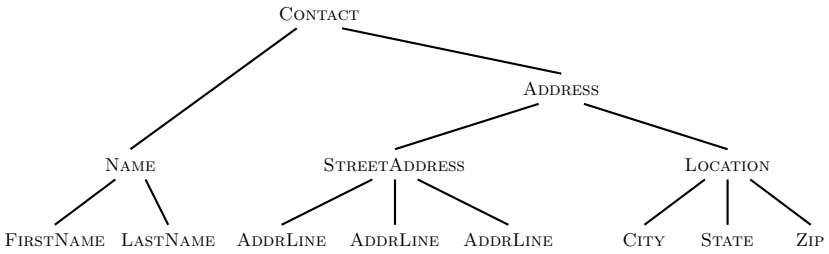

Figure 4: The reduced parse tree of the parse tree shown in Figure 3

where N^{j_i} spans $w_a w_{a+1} \dots w_b$. In principle, these features are much more powerful because they can analyze the *sequence* of words associated with the current non-terminal. For example, consider the sequence of tokens `Mavis Wood Products`. If the first and second tokens are on a line by themselves, then `Wood` is more likely to be interpreted as a LastName. However, if all three are on the same line, then they are more likely to be interpreted as part of the company name. Therefore, a feature AllOnTheSameLine (which when applied to any sequence of words returns 1 if they are on the same line) can help the CFG disambiguate between these cases. This type of feature cannot be included in a conditional Markov model.

4.3 Generating Labeled Data

The standard way of training a CFG is to use a corpus annotated with tree structure, such as the Penn Treebank[12]. Given such a corpus algorithms based on counting can be used to determine the probabilities (parameters) of the model. However, annotating the corpora with the tree-structure is typically done manually which is time consuming and expensive in terms of human effort. The only training data required for training the Markov models are the sequences of words and the corresponding label sequences. In this section we show that we can automatically generate the parse tree required for training the grammars from just the label sequences for a certain class of grammars.

Given a parse tree T for a sequence $w_1 w_2 \dots w_m$, let the reduced parse tree T' be the tree obtained by deleting all the leaves of T. Figure 4 shows the reduced parse tree obtained from Figure 3. In this reduced parse tree, the label sequence $\ell_1 \ell_2 \dots \ell_m$ corresponds to the leaves. We can think of this reduced tree as the parse tree of the sequence $\ell_1 \ell_2 \dots \ell_m$ over a different grammar in which the labels are the terminals. This new grammar is easily obtained from the original grammar by simply discarding all rules in which a label occurs on the LHS. If G' is the reduced grammar, then we can use G' to parse any sequence of labels. Note that G' can parse a sequence $\ell_1 \ell_2 \dots \ell_m$ if and only if there is a sequence of words $w_1 w_2 \dots w_m$ with ℓ_i being the label of w_i. We say that G is label-unambiguous if G' is unambiguous (i.e., for any sequence $\ell_1 \ell_2 \dots \ell_m$, there is at most one parse tree for this sequence in G'). To generate a parse tree for a label unambiguous grammar, given the label, we use the following two step method.

1. Generate a (reduced) parse tree for the label sequence using the reduced grammar G'.
2. Glue on the edges of the form $\ell_i \to w_i$ to the leaves of the reduced tree.

It is very easy to see that given any sequence of words $w_1 \dots w_m$, and their corresponding labels $\ell_1 \dots \ell_m$, this method gives us the unique parse tree for $w_1 \dots w_m$ which is compatible with the label sequence $\ell_1 \dots \ell_m$ (if one exists). Therefore, this method allows to generate a collection of parse trees given a collection of labeled sequences.

Doing this has at least two advantages. First, it allows for a apples-to-apples comparison with the CRF based methods since it requires no additional human effort to generate the parse trees (i.e., both models can work of exactly the same input). Secondly, it ensures that changes in grammar do not require human effort to generate new parse trees.

There is a natural extension of this algorithm to handle the case of grammars that are not label-unambiguous. If the grammar is not label-unambiguous, then there could be more than one tree corresponding to a particular labeled example. We could then pick an arbitrary tree, or possibly a tree that optimizes some other criterion. We could even use an EM-style algorithm to learn a probabilistic grammar for the reduced grammar. We experimented with some grammars which have a moderate amounts of label-ambiguity. In this case, we simply picked a tree with the smallest height. In our experiments, we did not observe any performance degradation by moderate amounts of ambiguity.

4.4 Training

The goal of training is to find the parameters λ that maximize some optimization criterion, which is typically taken to be the maximum likelihood criterion for generative models. A discriminative model assigns scores to each parse, and these scores need not necessarily be thought of as probabilities. A good set of parameters maximizes the "margin" between correct parses and incorrect parses. One way of doing this is using the technique of [21]. However, we use a simpler algorithm to train our discriminative grammar. This algorithm is a variant of the perceptron algorithm and is based on the algorithm for training Markov models proposed by Collins in [5]. Suppose that $\mathcal{T}$ is the collection of training data $\left\{(w^i, \ell^a, T^a) | 1 \le i \le m\right\}$, where $w^i = w_1^i w_2^i \dots w_{n_i}^i$ is the collection of words, $\ell^i = \ell_1^i \ell_2^i \dots \ell_{n_i}^i$ is the corresponding labels, and T^i is the parse tree. For each rule R in the grammar, we seek a setting of the parameters $\lambda(R)$ so that the resulting score is maximized for the correct parse T^i of w^i for $0 \le i \le m$. Our algorithm for training is shown in Figure 5. Convergence results for the perceptron algorithm appear in [7, 5] when the data is separable. In [5] some generalization results for the inseparable case are also given to

```
for r ← 1 ... numRounds do
  for i ← 1 ... m do
    T ← optimal parse of w^i with current parameters
    if T ≠ T^i then
      for each rule R used in T but not in T^i do
        if feature f_j is active in w^i then
          λ_j(R) ← λ_j(R) − 1;
        endif
      endfor
      for each rule R used in T^j but not in T do
        if feature f_j is active in w^i then
          λ_j(R) ← λ_j(R) + 1;
        endif
      endfor
    endif
  endfor
endfor
```

Figure 5: The Perceptron Training Algorithm

justify the application of the algorithm.

4.5 Correction Propagation

Kristjansson et. al introduced the notion of correction propagation for interactive form filling tasks [9]. In this scenario the user pastes unstructured data into the form filling system and observes the results. Errors are then quickly corrected using a drag and drop interface. After each correction the remaining observations can be relabeled so as to yield the labeling of lowest cost constrained to match the corrected field (i.e. the corrections can be *propagated*). For inputs containing multiple labeling errors correction propagation can save significant effort. Any score minimization framework such as a CMM or CFG can implement correction propagation. The main value of correction propagation can be observed on examples with two or more errors. In the ideal case, a single user correction should be sufficient to accurately label all the tokens correctly.

Suppose that the user has indicated that the token w_i actually has label ℓ_i. The CKY algorithm can be modified to produce the best parse consistent with this label. Such a constraint can actually accelerate parsing, since the search space is reduced from the set of all parses to the set of all parses in which w_i has label ℓ_i. CKY returns the optimal constrained parse in the case where all alternative non-terminals are removed from the cell associated with w_i.

5. EXPERIMENTAL RESULTS

For our experiments, we used 1487 contact records for training and 400 contact records for testing. The data was collected from web pages and email, and was hand-labeled with 24 labels (FirstName, MiddleName, LastName, NickName, Suffix, Title, JobTitle, CompanyName, Department, AddressLine, City, State, Country, PostalCode, HomePhone, Fax, CompanyPhone, DirectCompanyPhone, MobilePhone, Pager, Email, InstantMessage, WebPage). Only the labels for these examples was generated by hand. The complete parse tree for these examples were generated as described in the previous section.

5.1 Features

Each example token sequence was analyzed with a set of ad hoc features. Approximately 100 simple regular expression features were used, including IsCapitalized, AllCaps, IsDigit, Numeric, ContainsDash, EndsInPeriod, ContainsAtSign, etc. In addition there are 9 lexicon lists including: LastNames, FirstNames, States, Cities, Countries, JobTitles, CompanyNameComponents, Titles, StreetNameComponents.

Using these basic features as input, one addition high level feature is added: the output of a token classifier. The tokens in the sequence are labeled using a boosted collection of decision trees [8]. Each tree is of depth 3 and there are a total of 42 trees in the final classifier. The token classification performance of this algorithm is surprisingly high. Often it is within a few percent of the best Markov model.

5.2 The Collins Model

For these experiments we compare with the voted perceptron conditional Markov model due to Collins [5]. The form of the Collins model is very similar to the more well known CRF (the decoding algorithms are identical). While experiments show that the two systems are quite competitive, the implementation of the learning algorithm for the Collins model is much simpler. Our results using the Collins model are very similar to earlier results obtained by Kristjansson et. al on a similar problem [9].

The boosted classifier is used as an input feature both for the Collins model and the CFG model. The boosted classifier achieves quite a reasonable level of performance by itself. In the experiments corresponding to the results shown in Table 1, the boosted classifier achieves a word error rate of 13.49% and a record error rate of 65.86%. We have noticed that using the boosted classifier as an input feature speeds convergence of the learning by orders of magnitude. Our conjecture is that the boosted classifier frees the structural models from modeling the local evidence and allows them to focus on learning dependencies.

In addition the CFG model is given access to a set of region features which cannot be included in Collins model (as described above). There are a total of 80 regions features, including features such as ContainsNewLine and BeginsOnNaturalBoundary which test for natural end of field markers like new lines and commas. Other region features are extensions of the above mentioned token based regular expression and lexicon features to sequences of tokens.

5.3 Performance Metrics

There are a number of ways of comparing the quality of different labeling techniques. The most obvious is to examine the word labeling error rate. While this might be the most appropriate criterion for a batch setting, there are other appropriate criteria for interactive settings. In the interactive context, it is important to minimize the number of actions required of the user. Every time even one word of the record is incorrectly labeled, the user is forced to correct the labeling. Therefore, the percentage of records in which all labels are correctly assigned is perhaps a better metric. Both types or results are summarize in Table 1. Interestingly the Collins model has a fairly high word accuracy rate while performing poorly on record accuracy.

In cases where there is more than one mislabeled token in the record, it is possible to propagate the correction that the user enters for one of the mislabeled tokens to correct other tokens as well. The last row of Table 1 shows the percentage of records containing at least two mislabeled tokens that were completely corrected with just one user action. In summary, the CFG approach labels more than 70% of the

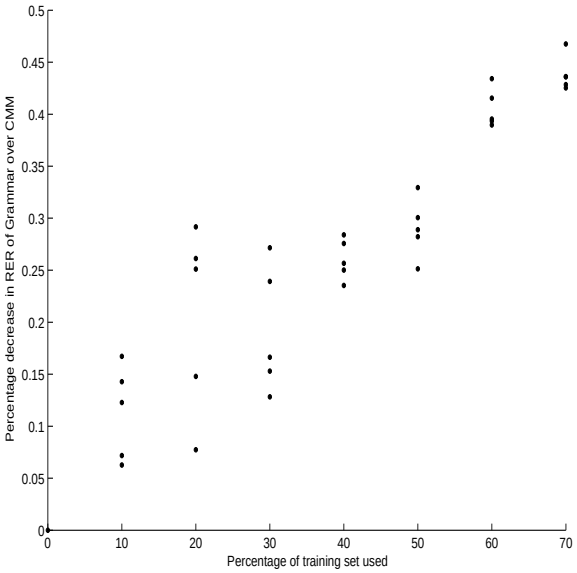

Figure 6: Record error-rate reduction using a discriminatively-trained grammar over a CMM

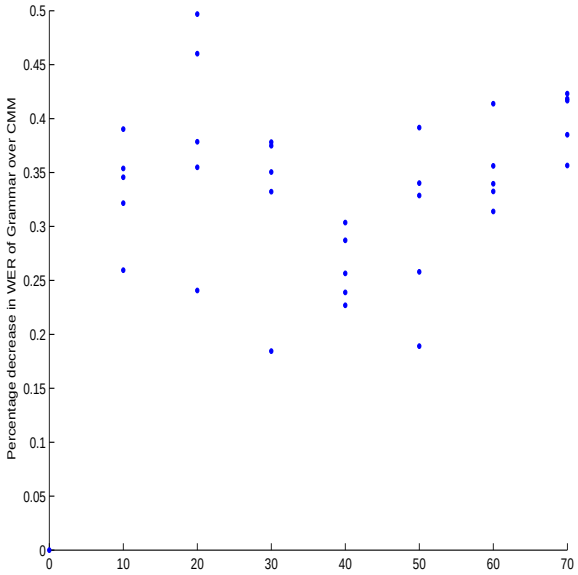

Figure 7: Word error-rate reduction using a discriminatively-trained grammar over a CMM

records completely correctly. For those records where there are more than 2 errors, the CFG corrects the entire record with a single word correction 50% of the time. This performance is significantly better than the best results obtained with a Markov model.

A number of experiments were performed to determine the relationship between training set size and testing error. Both the CMM and CFG were trained with different sized training sets. We experimented with training sets of size 10%, 20%, 30%, 40%, 50% 60% and 70% of our base training set. For each case we picked 5 random samples of the prescribed size and trained both the CMM and the grammar on this reduced training set. In every case, we observed that the grammar outperformed the Collins model and Figures 6 and 7 show the improvement of word error rate and the record error rate of the grammar over the CRF. It can be seen that there is some variability in the improvement, but the variability decreases with increase in the size of the training set. Because of the strong prior available to the grammar (the rules of the grammar), the grammar is able to better generalize even when there is just a little training data. The CMM is not able to do quite so well when the training size is small and hence the grammar shows a large improvement in word error rates over the CRF for small training sizes. However, even when there is a large amount of data, the grammar shows a large improvement over the CMM.

The CFG is also better able to propagate user corrections. As shown in Table 1, the CMM can use a single user correction to fix all errors in about 15% of the cases where there are at least two mislabeled tokens. In contrast, the CFG can propagate the user correction to fix over 50% of these cases.

6. CONCLUSIONS

This paper marries the powerful tools of statistical natural language processing to the analysis of non-natural language text. Experiments demonstrated that a discriminatively trained context free grammar can more accurately extract contact information than a similar conditional Markov model.

There are several advantages provided by the CFG system. The CFG, because its model is hierarchically structured, can generalize from less training data. What is learned about BusinessPhoneNumber can be shared with what is learned about HomePhoneNumber, since both are modeled as PhoneNumber. The CFG also allows for a rich collection of features which can measure properties of a sequence of tokens. The feature AllOnOneLine is a very powerful clue that an entire sequence of tokens has the same label (e.g. a title in a paper, or a street address). Another advantage is that the CFG can propagate long range label dependencies efficiently. This allows decisions regarding the first tokens in an input to effect the decisions made regarding the last tokens. This propagation can be quite complex and multi-faceted.

The effects of these advantages are many. For example a grammar based approach also allows for selective retraining of just certain rules to fit data from a different source. For example, Canadian contacts are reasonably similar to US contacts, but have different rules for postal codes and street addresses. In addition a grammatical model can encode a stronger set of constraints (e.g. there should be exactly one city, exactly one name, etc.). Grammars are much more robust to tokenization effects, since the two tokens which result from a word which which is split erroneously can be analyzed together by the grammar's sequence features

The application domain for discriminatively trained context free grammars is quite broad. It should be possible to analyze a wide variety of semi-structured forms such as resumes, tax documents, SEC filings, and research papers.

7. REFERENCES

[1] Vinajak R. Borkar, Kaustubh Deshmukh, and Sunita Sarawagi. Automatically extracting structure from

Performance Metric	Collins	Grammar	% Improvement
Word Error Rate	11.57%	6.29%	45.63%
Record Error Rate	54.71%	27.13%	50.41%
Percentage of records with at least 2 mislabeled tokens	32.34%	21.31%	34.09%
Record Error Rate of 2-error records with 1 user action	84.20%	46.38%	36.31%

Table 1: CMM/Grammar comparison

free text addresses. In *Bulletin of the IEEE Computer Society Technical committe on Data Engineering*. IEEE, 2000.

[2] Remco Bouckaert. Low level information extraction: A bayesian network based approach. In *Proc. TextML 2002*, Sydney, Australia, 2002.

[3] Claire Cardie and David Pierce. Proposal for an interactive environment for information extraction. Technical Report TR98-1702, 2, 1998.

[4] Rich Caruana, Paul Hodor, and John Rosenberg. High precision information extraction. In *KDD-2000 Workshop on Text Mining*, August 2000.

[5] M. Collins. Discriminative training methods for hidden markov models : Theory and experiments with perceptron algorithms. In *Proceedings of Empirical Methods in Natural Language Processing (EMNLP02)*, 2002.

[6] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

[7] Y. Freund and R. Schapire. Large margin classification using the perceptron algorithm. *Machine Learning*, 37(3):277–296.

[8] Y. Freund and R. E. Schapire. Experiments with a new boosting algorithm. In *International Conference on Machine Learning*, pages 148–156, 1996.

[9] T. Kristjansson, A. Culotta, P. Viola, and A. McCallum. Interactive information extraction with constrained conditional random fields. In *Proceedings of the 19th international conference on artificial intelligence, AAAI*, pages 412–418, 2004.

[10] John Lafferty, Andrew McCallum, and Fernando Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. 18th International Conf. on Machine Learning*, pages 282–289. Morgan Kaufmann, San Francisco, CA, 2001.

[11] C. D. Manning and H. Schütze. *Foundations of Statistical Natural Language Processing*. The MIT Press, 1999.

[12] M. Marcus, G. Kim, M. Marcinkiewicz, R. MacIntyre, A. Bies, M. Ferguson, K. Katz, and B. Schasberger. The penn treebank: Annotating predicate argument structure, 1994.

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

[14] Andrew McCallum and Wei Li. Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. In Marti Hearst and Mari Ostendorf, editors, *HLT-NAACL*, Edmonton, Alberta, Canada, 2003. Association for Computational Linguistics.

[15] Kamal Nigam, John Lafferty, and Andrew McCallum. Using maximum entropy for text classification. In *IJCAI'99 Workshop on Information Filtering*, 1999.

[16] David Pinto, Andrew McCallum, Xing Wei, and W. Bruce Croft. Table extraction using conditional random fields. In *Proceedings of the ACM SIGIR*, 2003.

[17] L.R. Rabiner. A tutorial on hidden markov models. In *Proc. of the IEEE*, volume 77, pages 257–286, 1989.

[18] Tobias Scheffer, Christian Decomain, and Stefan Wrobel. Active hidden markov models for information extraction. In *Advances in Intelligent Data Analysis, 4th International Conference, IDA 2001*, 2001.

[19] Fei Sha and Fernando Pereira. Shallow parsing with conditional random fields. In Marti Hearst and Mari Ostendorf, editors, *HLT-NAACL: Main Proceedings*, pages 213–220, Edmonton, Alberta, Canada, 2003. Association for Computational Linguistics.

[20] J. Stylos, B. A. Myers, and A. Faulring. Citrine:providing intelligent copy-and-paste. In *Proceedings of ACM Symposium on User Interface Software and Technology (UIST 2004)*, pages 185–188, 2005.

[21] B. Tasker, D. Klein, M. Collins, D. Koller, and C.Manning. Max-margin parsing. In *Empirical Methods in Natural Language Processing (EMNLP04)*, 2004.