Integrating High Precision Rules with Statistical Sequence Classifiers for Accuracy and Speed

Wenhui Liao, Marc Light, and Sriharsha Veeramachaneni
Research and Development,Thomson Reuters
610 Opperman Drive, Eagan MN 55123

Abstract

Integrating rules and statistical systems is a challenge often faced by natural language processing system builders. A common subclass is integrating high precision rules with a Markov statistical sequence classifier. In this paper we suggest that using such rules to constrain the sequence classifier decoder results in superior accuracy and efficiency. In a case study of a named entity tagging system, we provide evidence that this method of combination does prove efficient than other methods. The accuracy was the same.

1 Introduction

Sequence classification lies at the core of several natural language processing applications, such as named entity extraction, Asian language segmentation, Germanic language noun decompounding, and event identification. Statistical models with a Markov dependency have been successful employed to perform these tasks, e.g., hidden Markov models (HMMs)(Rabiner, 1989) and conditional random fields (CRFs)(Lafferty et al., 2001). These statistical systems employ a Viterbi (Forney, 1973) decoder at runtime to efficiently calculate the most likely label sequence based on the observed sequence and model. Statistical machine translation systems make use of similar decoders.

In many situations it is beneficial, and sometimes required, for these systems to respect constraints from high precision rules. And thus when building working sequence labeling systems, researchers/software engineers are often faced with the task of combining these two approaches. In this paper we argue for a particular method of combining statistical models with Markov dependencies and high precision rules. We outline a number of ways to do this and then argue that guiding the decoder of the statistical system has many advantages over other methods of combination.

But first, does the problem of combining multiple approaches really happen? In our experience the need arises in the following way: a statistical approach with a Markov component is chosen because it has the best precision/recall characteristics and has reasonable speed. However, a number of rules arise for varied reasons. For example, the customer provides domain knowledge not present in the training data or a particular output characteristic is more important that accuracy. Consider the following fictitious but plausible situation: A named entity tagging system is built using a CRF. The customer then provides a number of company names that cannot be missed, i.e., false negatives for these companies are catastrophic but false positives can be tolerated. In addition, it is known that, unlike in the training data, the runtime data will have a company name immediately before every ticker symbol. The question facing the builder of the system is how to combine the CRF with rules based on the must-find company list and the company-name-before-every-ticker-symbol fact.

Similar situations arise for the other sequence tagging situations mentioned above and for machine translation. We suspect that even for non-language applications, such as gene sequence labeling, similar situations arise.

Proceedings of the NAACL HLT Workshop on Software Engineering, Testing, and Quality Assurance for Natural Language Processing, pages 74–77,
Boulder, Colorado, June 2009.

In the next section we will discuss a number of methods for combining statistical systems and high precision rules and argue for guiding the decoder of the statistical model. Then in section 3, we describe an implementation of the approach and give evidence that the speed benefits are substantial.

2 Methods for Combining a Markov Statistical System and High Precision Rules

One method of combination is to encode high precision rules as features and then train a new model that includes these features. One advantage is that the system stays a straightforward statistical system. In addition, the rules are fully integrated into the system allowing the statistical model weigh the rules against other evidence. However, the model may not give the rules high weight if training data does not bear out their high precision or if the rule trigger does not occur often enough in the training data. Thus, despite a "rule" feature being on, the system may not "follow" the rule in its result labeling. Also, addition or modification of a rule would require a retraining of the model for optimal accuracy. The retraining process may be costly and/or may not be possible in the operational environment.

Another method is to run both the statistical system and the rules and then merge the resulting labels giving preference to the labels resulting from the high precision rules. The benefits are that the rules are always followed. However, the statistical system does not have the information needed to give an optimal solution based on the results of the high precision rules. In other words, the results will be inconsistent from the view of the statistical system; *i.e.*, if it had know what the rules were going to say, then it would have calculated the remaining part of the label sequence differently. In addition, the decoder considers part of the label sequence search space that is only going to be ruled out, pun intended, later.

Now for the preferred method: run the rules first, then use their output to guide the decoder for the statistical model. The benefits of this method are that the rules are followed, the statistical system is informed of constraints imposed by the rules and thus the statistical system calculates optimal paths given these constraints. In addition, the decoder considers only those label sequences consistent with these constraints, resulting in a smaller search space. Thus, we would expect this method to produce both a more accurate and a faster implementation.

Consider Figure 1 which shows a lattice that represents all the labeling sequences for the input ... *Microsoft on Monday announced a* ... The possible labels are O (out), P (person), C (company), L (location) . Assume *Microsoft* is in a list of must-find companies and that *on* and *Monday* are part of a rule that makes them NOT names in this context. The bold points are constraints from the high-precision rules. In other words, only sequences that include these bold points need to be considered.

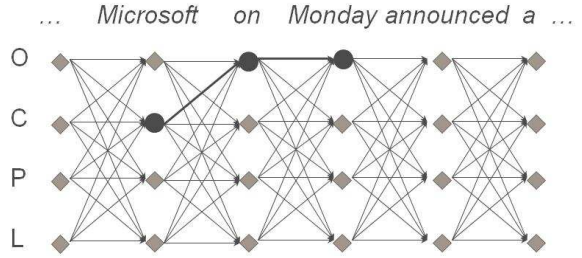

Figure 1: Guiding decoding with high-precision rules

Figure 1 also illustrates how the constraints reduce the search space. Without constraints, the search space includes $4^6 = 4096$ sequences, while with constraints, it includes only $4^3 = 64$.

It should also be noted that we do not claim to have invented the idea of constraining the decoder. For example, in the context of active learning, where a human corrects some of the errors made by a CRF sequence classifier, (Culota et al., 2006) proposed a constrained Viterbi algorithm that finds the path with maximum probability that passes through the labels assigned by the human. They showed that constraining the path to respect the human labeling considerably improves the accuracy on the remaining tokens in the sequence. Our contribution is noticing that constraining the decoder is a good way to integrate rule output.

3 A Case Study: Named Entity Recognition

In this section, we flesh out the discussion of named entity (NE) tagging started above. Since the entity type of a word is determined mostly by the context of the word, NE tagging is often posed as a sequence

classification problem and solved by Markov statistical systems.

3.1 A Named Entity Recognition System

The system described here starts with a CRF which was chosen because it allows for the use of numerous and arbitrary features of the input sequence and it can be efficiently trained and decoded. We used the Mallet toolkit (McCallum, 2002) for training the CRF but implemented our own feature extraction and runtime system. We used standard features such as the current word, the word to the right/left, orthographic shape of the word, membership in word sets (e.g., common last names), features of neighboring words, etc.

The system was designed to run on news wire text and based on this data's characteristics, we designed a handful of high precision rules including:

Rule 1*: if a token is in a must-tag list, this token should be marked as Company no matter what the context is.*

Rule 2*: if a capitalized word is followed by certain company suffix such as* ***Ltd, Inc, Corp****, etc., label both as Company.*

Rule 3*: if a token sequence is in a company list and the length of the sequence is larger than 3, label them as Company.*

Rule 4*: if a token does not include any uppercase letters, is not pure number, and is not in an exceptions list, label it as not part of a name. (The exceptions list includes around 70 words that are not capitalized but still could be an NE, such as* ***al, at, in, -****, etc.)*

Rule 5*: if a token does not satisfy rule 4 but its neighboring tokens satisfy rule 4, then if this token is a time related word, label it as not part of a name. (Example time tokens are* ***January*** *and* ***Monday****.)*

The first three rules aim to find company names and the last two to find tokens that are *not* part of a name.

These rules are integrated into the system as described in section 2: we apply the rules to the input token sequence and then use the resulting labels, if any, to constrain the Viterbi decoder for the CRF.

A further optimization of the system is based on the following observation: features need not be calculated for tokens that have already received labels from the rules. (An exception to this is when features are copied to a neighbor, e.g., the token to my left is a number.) Thus, we do not calculate many features of rule-labeled tokens. Note that feature extraction can often be a major portion of the computational cost of sequence labeling systems (see Table 1(b))

3.2 Evidence of Computational Savings Resulting from Our Proposed Method of Integration

We compare the results when high-precision rules are integrated into CRF for name entity extraction (company, person, and location) in terms of both accuracy and speed for different corpora. Three corpora are used, CoNLL (CoNLL 2003 English shared task official test set), MUC (Message Understanding Conference), and TF (includes around 1000 news articles from Thomson Financial).

Table 1(a) shows the results for each corpora respectively. The baseline method does not use any high-precision rules, the Post-corr uses the high-precision rules to correct the labeling from the CRF, and Constr-viti uses the rules to constrain the label sequences considered by the Viterbi decoder. In general, Constr-viti achieves slightly better precision and recall.

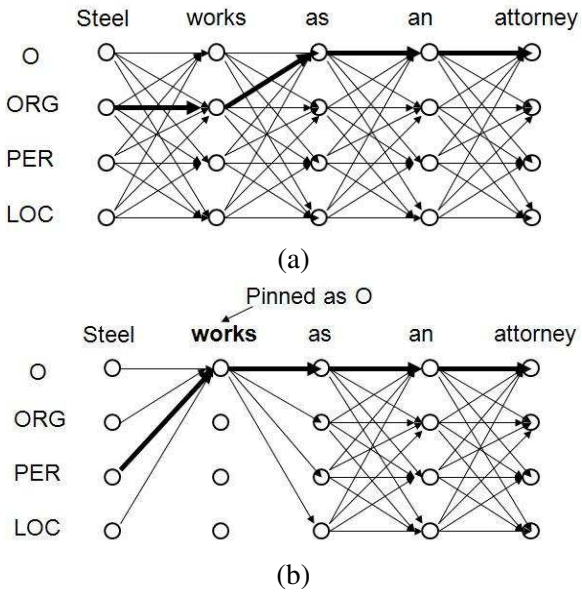

Figure 2: (b) A test example : (a) without constraints; (b) with constraints

To better understand how our strategy could improve the accuracy, we did some analysis on the

Table 1: Experiment Results

Database	Methods	Precision	Recall	F1
CoNLL	Baseline	84.38	83.02	83.69
	Post-corr	85.87	84.86	85.36
	Constr-viti	**85.98**	**85.55**	**85.76**
TF	Baseline	**88.39**	82.42	85.30
	Post-corr	87.69	88.30	87.99
	Constr-viti	88.02	**88.54**	**88.28**
MUC	Baseline	**92.22**	88.72	**90.43**
	Post-Corr	91.28	88.87	90.06
	Constr-viti	90.86	**89.37**	90.11

(a)Precision and Recall

Methods	Rules	Features	Viterbi	Overall
Baseline	0	0.78	0.22	1
Post-corr	0.08	0.78	0.22	1.08
Constr-vite	0.08	0.35	0.13	**0.56**
Baseline	0	0.85	0.15	1
Post-Corr	0.14	0.85	0.15	1.14
Constr-vite	0.14	0.38	0.1	**0.62**
Baseline	0	0.79	0.21	1
Post-corr	0.12	0.79	0.21	1.12
Constr-vite	0.12	0.36	0.12	**0.60**

(b)Time Efficiency

testing data. In one example as shown in Figure 2, *Steel works as an attorney*, without high-precision rules, *Steel works* is tagged as a company since it is in our company list. Post-correction changes the label of *works* to O, but it is unable to fix *Steel*. With our strategy, since *works* is pinned as O in the Vertibi algorithm, *Steel* is tagged as Per. Thus, compared to post-correction, the advantage of constraining Viterbi is that it is able to affect the whole path where the token is, instead a token itself. However, the improvements were not significant in our case study. We have not done an error analysis. We can only speculate that the high precision rules do not have perfect precision and thus create a number of errors that the statistical model would not have made on its own.

We also measured how much the constrained Viterbi method improves efficiency. We divide the computational time to three parts: time in applying rules, time in feature extraction, and time in Viterbi computation. Table 1(b) lists the time efficiency. Instead using specific time unit (e.g. second), we use ratio instead by assuming the overall time for the baseline method is 1. As shown in the table, for the three data sets, the overall time of our method is 0.56, 0.62, and 0.60 of the time of the baseline algorithm respectively. The post-correction method is the most expensive one because of the extra time spending in rules. Overall, the constrained Viterbi method is substantially faster than the Baseline and Post-corr methods in addition to being more accurate.

4 Conclusions

The contribution of this paper is the repurposing of the idea of constraining a decoder: we constrain the decoder as a way to integrate high precision rules with a statistical sequence classifier. In a case study of named entity tagging, we show that this method of combination does in fact increase efficiency more than competing methods without any lose of accuracy. We believe analogous situations exist for other sequence classifying tasks such as Asian language segmentation, Germanic language noun decompounding, and event identification.

References

Aron Culota, Trausti Kristjansson, Andrew McCallum, and Paul Viola. 2006. Corrective feedback and persistent learning for information extraction. *Artificial Intelligence Journal*, 170:1101–1122.

G. D. Forney. 1973. The viterbi algorithm. *Proceedings of the IEEE*, 61(3):268–278.

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

A.K. McCallum. 2002. Mallet: A machine learning for language toolkit. http://mallet.cs.umass.edu.

Lawrence R. Rabiner. 1989. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, pages 257–286.