Rule Extraction from A Trained Conditional Random Field Model

Bhuban Mohan Seth
Computer Science and Enginnering
IIT Bombay
Powai, Mumbai 400076
bmseth@cse.iitb.ac.in

Dr. Pushpak Bhattacharyya
Computer Science and Enginnering
IIT Bombay
Powai, Mumbai 400076
pb@cse.iitb.ac.in

Abstract

Conditional Random Field (CRF) has proven to be highly successful for sequence labeling problems like part of speech tagging, segmentation etc. However, the model acts like a *black box*, providing no insight into what is learned. We propose a system for rule extraction from CRF to assist comprehensibility of the model. Experiments on POS tagging and chunking problem in English are performed as case studies. We test the quality of the extracted rule base by implementing a majority voting rule based tagger, which shows maximum precision of 93.9% for POS tagging and 77% for chunking. The obtained rules conform to our linguistic knowledge of English. We also give quantitative comparison of our approach with PART decision list and C4.5 decision tree learner. Comprehensibility of statistical models is our guiding principle.

1 Introduction

Conditional random field (CRF) (Lafferty et al., 2001) has proven to be highly successful in different sequence labeling problems in natural language processing like parts of speech tagging, segmentation etc. However, CRF does not provide any explanation of its operation and acts like a *black-box*. The opaqueness of CRF models can be remedied through the use of rule extraction, whose primary objective is to induce rules that mimic the behaviour of the CRF model as closely as possible. Martens (Martens, et al , 2008) mentioned the rationale behind rule extraction from any non-linear black-box model as to understand the classifications made by the model and gain some insight into the workings of the model and suggested fidelity as the corresponding quantitative measure.

Our work presented here, aims to extract the knowledge learned by a CRF model and represent them through human comprehensible symbolic *if-then* rules. This will clearly reveal what the model has learned from the training corpus and shed light on the classifications made by the model. Specifically, it will be then possible to detect the incorrect rules learned by the model and apply the corresponding correct rules in a preprocessing stage to improve the overal performance. To the best of our knowledge, there has been no previous work on rule extraction from CRF.

The roadmap of the paper is as follows. Literature on rule extraction is presented in section 2. Section 3 introduces the sequence labeling problem along with an introduction to CRF. We present our approach in section 4. This is followed by experimental results in section 5. Section 6 concludes the paper with pointers to future work.

2 Related Works

There has been a lot of work on rule extraction from different predictive models to improve the comprehensibility of the model. Huysmans (Huysmans et al., 1998) discussed how different rule extraction algorithms can be classified and evaluated. They also discussed some of the important rule extraction al-

[1]Proceedings of ICON-2011: 9th International Conference on Natural Language Processing, Macmillan Publishers, India. Also accessible from http://ltrc.iiit.ac.in/proceedings/ICON-2011

gorithms in detail. One of the major works in rule mining in the field of natural language processing is Brills tagger (Brill, 1995). This work is an example of rule mining, where the input is the tagged data and a set of transformation rule templates and the output is an ordered list of transformation rules. The idea of transformation based error driven learning is used. The output is an ordered list of transformation rules which is not as comprehensible as a set of unordered rules.

Another related work is rule extraction from artificial neural network (ANN) by Towell and Shavlik (Towell, G., G., & Shavlik, J., W., 1993). Here rules are found, based on the link weights of an artificial neural network. The main idea behind the rule extraction technique is to consider each node of the trained ANN and find a constraint on the value of the inputs of the node such that the node is guaranteed to have activation value one and then represent this constraint in terms of symbolic rules. This way, an unordered list of symbolic rules is obtained to represent the knowledge learned by the trained ANN. Finding the complete set of all possible constraints is a tedious job and computationally costly. Hence, the framework of rules is set to M of N rules. An M of N rule is of the following form

> If any M of the given N antecedents are true)
> Then output is true

Towell and Shavlik (Towell, G., G., & Shavlik, J., W., 1993) also described how the above rule framework allows efficient reduction of the problem size by clustering and elimination technique on the inputs of a node, based on their link weights.

Decision Trees are widely used in predictive modeling. Quinlan's C4.5 (Quinlan, J., R., 1986) is one of the most popular algorithms for the induction of decision trees. Quinlan (Quinlan, J., R., 1987) discussed techniques to obtain an unordered list of rules from a trained decision tree.

Decision list is another approach of rule mining. A decision list (Eibe et al. , 1998) gives an ordered list of rules. One of the most effective implementations of decision list is PART, available in Weka[1] package.

CRF basically projects the classification problem into a high dimensional feature space and hence, one way to make the model interpretable is to obtain a sparse solution vector. In this regard Tibshirani (Tibshirani, 1996) introduced lasso to build a sparse model where the irrelevant features will get weights exactly equal to zero. While, this approach is suitable for the problem of relevant feature selection, it fails to bring human comprehensibility to the CRF model. We will discuss this issue while discussing our approach in sub-section 4.2 and in sub-section 5.1.

3 Conditional Random Field

A sequence labeling problem is to find the correct label sequence $Y = y_1y_2y_3...y_n$ for the given observation sequence $X = x_1x_2x_3...x_n$. Here each label y_i belongs to a finite label alphabet Ω. For example, X may range over all possible sentences in English and Y ranges over parts of speech tags of those sentences, where Ω is the set of all possible part-of-speech tags in English.

A CRF (Lafferty et al., 2001) is a discriminative model, and models the conditional probability of the label sequence given the observation sequence, as follows.

$$p(Y/X) = \frac{1}{Z(X)} \exp(\sum_{c \in C} \sum_{i=1}^{F} \lambda_i f_i(X_c, Y_c)) \quad (1)$$

$$Z(X) = \sum_{Y} \exp(\sum_{c \in C} \sum_{i=1}^{F} \lambda_i f_i(X_c, Y_c)) \quad (2)$$

The index i ranges over a large set of F real valued feature functions. Each feature function f_i is preferably taken to be a binary function which takes the value 1 if the feature is present, 0 otherwise. C is the set of all cliques in the graph of X and Y. For every clique $c \in C, X_c = X \cap c$ and $Y_c = Y \cap c$. λ_i is the weight of the feature function f_i. Although, a general CRF allows arbitrary graphical structure on Y, we will focus on a more constrained class of CRF, called linear chain CRF, where the graph structure of Y is a linear chain.

The feature functions in CRF can be defined either on a single label (state feature function) or on an edge (transition feature function) of the graph of Y. An example of a state feature function for En-

[1] http://www.cs.waikato.ac.nz/ml/weka/

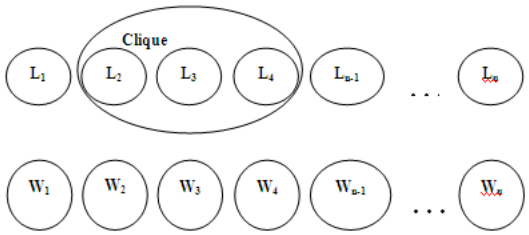

Figure 1: Window of length *three* around current observation L_3

glish POS tagging is as follows:

$$f(x,y)=\langle \begin{array}{ll} 1 & \text{if } x \text{ is capitalized and } y \text{ is} \\ & \text{proper noun} \\ 0 & \text{otherwise} \end{array} \rangle$$

An example of transition feature function for English POS tagging is as follows:

$$f(x,y_i,y_{i-1})=\langle \begin{array}{ll} 1 & \text{if } x \text{ ends with } \textit{'ing'},\ y_{i-1} \\ & \text{is } \textit{Auxiliary Verb} \text{ and } y_i \text{ is} \\ & \text{a proper noun} \\ 0 & \text{otherwise} \end{array} \rangle$$

The set of feature functions is usually pre-decided by using a set of feature templates derived from the domain knowledge of the problem. However, McCallum (McCallum, A., 2003) discussed an efficient method of automatic feature induction for CRF. It is to be noted that during training, the weight values of these feature functions are obtained from the training data.

4 Our Approach to Rule Extraction from CRF

Given a sentence, we consider a window of length $three$ ($L = 3$) around the current observation (figure 1). That means our window contains the current, the previous and the next observation.

Mathematically, we want to find rules of the following form,

If
Clique C around current observation W_i shows a property P_c
Then,
W_i has label L_i

The property of a clique is a set of features in that clique. For example, the rule below:

If
The previous word is *Noun*, the current word has suffix *'s'* and the next word is *noun*
Then,
The current word has POS tag *Verb*

We generalize this framework by considering a number of predefined attributes over observations. We call the suffix of the word, the word itself and the label of the word as the attributes of the word.

Let there be *A* number of attributes of each word. Ratnaparkhi (Ratnaparkhi, 1996) listed typical attributes of word for POS tagging. Now assume the current label depends only on the attributes of the observations in the window of length L around the current observation. We define the framework of rules we are interested, in as follows:

If $AttrSet = valueset$
Then $Y_{curr} = y$
where,
$AttrSet \subset \bigcup_{i \in Winw} \{X_{i:1}, X_{i:2}, ..., X_{i:A}, Y_i\}$

$X_{i:j}$ is the j^{th} attribute of the observation in i relative position to current observation.
Y_i is the label of observation in i relative position, other than current observation.
Winw is the window of Length L around the current observation
$valueset$ is one possible grounding of $AttrSet$, Y_{curr} is the current label and y is a possible label

Note, we have formalized the notion of the property (P_c) of a clique to be any subset of all attributes of all observations in the window of length L around the current observation.

We define any feature function to be of the following form.

$f(AttrSet = valueset, Y_{curr} = y)$
where,
$AttrSet \subset \bigcup_{i \in Winw} \{x_{i:1}, x_{i:2}, ..., x_{i:A}, Y_i\}$

4.1 Analysis of the Model

Close analysis of the Improved Iterative Scaling (IIS) (Berger, A. L., 1997) training and the Viterbi inference methods yields the observations described in the following subsections.

Key Observations: One of the training procedures for CRF is IIS algorithm , Whereas L-BFGS, a state-of-the-art quasi-Newton method for large optimization problems, offers a fast training procedure for CRF, but IIS algorithm is useful to analyze the CRF model parameters. Hence, we discuss IIS algorithm to the extent, it helps in establishing our approach to rule extraction.

The main idea behind the IIS algorithm is as follows. Initially, it assigns arbitrary values to model parameters. Then, it iteratively increments or decrements the model parameter values such that the log likelihood function continually increases in each iteration. This process goes on until all the model parameter values converge.

Improved Iterative Scaling Algorithm:

Initialize all model parameters (λ_i) to arbitrary values
Repeat until convergence,
For each model parameter(λ_i),
Do,
$\lambda_i = \lambda_i + \delta_i$
Where, $\delta_i = \frac{1}{S} log \frac{\widetilde{E}(f_i)}{E(f_i)}$
S is a constant
$E(f_i)$ is expected no. of occurrence of f_i according to the model with current value of λ_i
$\widetilde{E}(f_i)$ is average no. of occurrence of feature f_i in the corpus

Hence, at convergence of IIS algorithm, the increment or decrement value for the weights of each feature function will be zero; in other words, $E(f_i) = \widetilde{E}(f_i)\ \forall f_i$

Also, at every iteration,

$\delta_i > 0$ iff $\widetilde{E}(f_i) > E(f_i)$
$\delta_i < 0$ iff $\widetilde{E}(f_i) < E(f_i)$

During IIS training, among a set of conflicting feature functions, the ones which occur more frequently than others will get positive λ, while the ones which occur less frequent will get negative λ. Now, higher this margin of difference higher will be the increments. High positive λ means the feature is frequently occurring in the corpus and hence, indicating a correct rule.

4.2 Rule Extraction Procedure:

From the above discussion it is evident that if a feature function acquires a negative λ value, then it means the joint occurrence of the arguments of the feature function is not expected. One such example is

f_1 (current word = the & current tag= $Noun$)

λ value of this feature function will be negative since the word '*the*' is not expected to have POS tag *Noun*. Hence, the frequency of occurrence of this feature function will be very low. Comparatively, its conflicting feature function, say,

f_2 (current word = the & current tag= $Determiner$)

will occur much more frequently and will acquire a high positive λ value.

4.2.1 Shortcomings of looking at the highest weight approach

In the light of the above discussion, it may seem that just by looking at the feature functions with highest λ values, one would be able to interpret the behavior of the model. Actually, individual λ values can mislead. Let us consider, the two feature functions below.

f_1(*last two letters of current word=ed,POS tag of current word=VVN(verb in past particle)*)

f_2(*last two letters of current word=ed,POS tag of current word=VVD(verb in past tense)*)

We trained a CRF on 50,000 POS tagged words from BNC corpus and observed that λ values of the above two feature functions are 2.66 and 2.67 respectively and they appear in the list of top 10λ values of the trained model. Inspection of the training corpus showed that, both the above feature functions occur frequently in the corpus which justifies the high λ values, but clearly, neither of these feature functions indicates a correct rule. In general, *looking at the highest weights approach* fails when multiple conflicting feature functions occur frequently in the corpus and acquire high λ values. This also explains why finding a set of most important feature func-

tions by the techniques such as lasso or automatic feature induction will often mislead us. We formalize this notion in below:

Consider the following general feature function. $f(AttrSet = valueset, Y_{curr} = y)$

Now, if this feature function has a positive λ value then the following rule is definitely a candidate rule to be considered as a correct rule.

If $AttrSet = valueset$ then, $Y_{curr} = y$

But the ambiguity results when its conflicting feature functions have positive λ values.

$f(AttrSet = valueset, Y_{curr} = y')$ where $y' \neq y$

In such cases we need to discriminate between these feature functions (and hence the corresponding rules), where the same grounding of same attribute set is present in each one of them, but each indicates a different label for current observation.

4.2.2 Our Heuristic Based Approach

We propose a simple heuristic measure for this discrimination and we can call this the *confidence score* for a rule:

$$\text{Conf. Score(if } \textit{AttrSet=valueset} \text{ then } Y_{cur} = y) = \frac{\lambda of f(AttrSet = valueset, Y_{cur} = y)}{\sum_{y':\lambda>0} \lambda of f(AttrSet = valueset, Y_{cur} = y)} \quad (3)$$

The above confidence score is defined on all rules only under the following condition:

λ of f(*AttrSet=valueset*, $Y_{curr} = y) >= 0$

The intuition is very clear. We take the quotient of the λ of the current feature function f with the sum of the λ of all conflicting feature function of f (including f itself).

So, if there is no conflicting feature function then the measure will come out to be 1 which is justifiable.

If there are two conflicting feature functions and one of them has very high λ value and the other has small λ value then the first feature function will have high confidence score while the other will have low confidence Score.

In case all the conflicting feature functions have comparable λ value (means they are all equally likely and hence not a single rule is acceptable), then each of them will have very low confidence heuristic score.

Now, a special case arises when a particular feature function acquires a tiny positive λ value, but the corresponding rule gets a unit confidence heuristic score just because none of its conflicting feature function got positive λ value. This situation arises when all the feature functions in a conflicting set occur evenly and one of them occurs a little bit more frequently. Clearly, the confidence score of the rule is misleading.

We experimentally used a heuristic to alleviate this problem. Instead of taking all feature functions with positive λ values, we have considered all the feature function whose λ value is greater than some **minimum threshold** (α) and we changed the condition of the confidence score heuristic to the following.

λ of f(*AttrSet=valueset*, $Y_{curr} = y) >= \alpha$

Experimental results with α value of 0.1 shows that this heuristic reduces the size of the produced rule base. It also eliminates the less informative subsumed rules. We show the results in the experiment section.

As can be speculated, putting threshold on the confidence score will be the parameter for quality of the extracted rules. We call this **Confidence Threshold**(θ).

We ensure the generality of the rules by checking whether the number of occurrence of the pattern (feature function) in the training corpus is greater than a minimum threshold. This criterion eliminates many rules which try to account for the noise in the data.

4.3 Rule Extraction Algorithm:

- Extract λ values for all feature functions from trained CRF model.
- Eliminate all feature functions having negative λ values.
- For each feature function of form

  $f(AttrSet = valueset, Y_{curr} = y)$
  Calculate the confidence score (conf. score) of the corresponding rule
  If AttrSet = valueset then, $Y_{curr} = y$

  If
  Conf. Score>Conf. Threshold(θ)
  Then,
  Insert the rule in Set of Extracted Rules

5 Experimental Results

To test the performance of our rule extraction algorithm from CRF, we have chosen POS tagging and chunking problem in English.

We have used British National Corpus (BNC)[2] for POS tagging problem and CoNLL 2000 data[3] for chunking problem throughout our experiments. We have used open source CRF tool called Pocket_CRF[4], Weka J48 decision tree learner and Weka PART decision list in our experiments.

In our experiments, we have measured the quality of the extracted rule base by creating a *majority voting rule based tagger* for both the problem. We have chosen the precision, recall and average rule size of this rule based tagger as a measure of the quality, scope and comprehensibility of the extracted rule base respectively. These are defined as follows.

[2]http://www.natcorp.ox.ac.uk/

[3]http://www.cnts.ua.ac.be/conll2000/

[4]http://sourceforge.net/projects/pocket-crf-1/

Inverse Reg. Level(C)	Sparsity(%)	Accuracy(%)
100	2.05	93.68
50	2.13	93.77
10	2.31	93.69
1	10.27	93.99
0.1	33.74	93.76
0.01	51.52	91.57
0.001	54.59	84.23

Table 1: Effect of Varying L1 Regularization on Model Sparsity and Accuracy on English Chunking Problem

$$\text{Precision} = \frac{\text{\# observation correctly tagged}}{\text{\# observation tagged}} \quad (4)$$

$$\text{Support} = \frac{\text{\# observation correctly tagged}}{\text{Total \# observation}} \quad (5)$$

$$\text{Average Rule Size of RuleBase} = \frac{\sum_{rule} (\text{\# Predicates in rule body})}{\text{\# rules in RuleBase}} \quad (6)$$

5.1 Experiment on Regularization

As stated in section 2 and sub-section 4.2, building a L1 regularized sparse CRF model is assumed to bring interpretability to the model. To investigate this, we trained a CRF on CoNLL 2000 chunking dataset and varied the level of L1 regularization (pocket_crf allows the parameter C which is basically the inverse of the weight to regularization term). We report in table 1 the model sparsity (number of zero model parameters divided by total number of model parameters) and accuracy on validation data set. The drastic fall in accuracy forbids us to increase level of regularization further. At the highest level of regularization, the model contains millions of non-zero weights, thereby, fails to make the model interpretable. However, we find regularization level of 1.0 does make the model a bit sparse and improves the accuracy by eliminating over-fitting problem. Hence, hereafter, we fix the regularization level to 1 whenever we train a CRF.

#Rules	θ	P(%)	R(%)	F(%)
6864	0.95	93.10	59.87	72.87
9327	0.90	93.94	62.09	74.76
9462	0.85	92.74	62.30	74.53
9838	0.80	91.01	65.17	75.95
9959	0.75	85.92	64.95	73.98

Table 2: Effect of confidence threshold on performance of rule extraction algorithm on English parts of speech tagging problem

5.2 Measures of Rules and Rule Base

To measure the *"interestingness"* and *"goodness"* of individual rules, we have used *lift*, *confidence* and *conviction* as the metrics (Bayardo, et al , 1999). These are defined as follows.

$$\text{Support}(A)=\frac{|A|}{|Dataset|} \quad [7]$$

$$\text{Confidence}(A \rightarrow B)=\frac{Support(A \bigcup B)}{Support(A)} \quad [8]$$

$$\text{Lift}(A \rightarrow B)=\frac{Support(A \bigcup B)}{Support(A)*Support(B)} \quad [9]$$

$$\text{Conviction}(A \rightarrow B)=\frac{1-Support(B)}{1-Confidence(A \rightarrow B)} \quad [10]$$

Note that, our confidence score measure is different from the measure of *confidence*. The former is a heuristic defined over λ values, whereas, the latter is a count based metric based on gold data.

5.3 Experiment on Confidence Threshold (θ)

Confidence heuristic threshold (θ) is the parameter which controls the quality and scope of the rule base. Hence, we have experimented with different values of confidence heuristic threshold θ.

5.3.1 POS Tagging Experiment

We trained a CRF model on 50,000 POS tagged data and tested the extracted rule base on 12,500 test data. We kept minimum lambda threshold α to be $zero$. The trained CRF model showed 86.7% accuracy on test data. The experimental results are shown in table 2. We report 0.9 as the optimal value of θ for POS tagging problem.

Due to large size of the extracted rule base, it is as incomprehensible as the trained CRF model. Therefore, we focus on analysis of the extracted rules, which reveals the following points.

#Rules	θ	P(%)	R(%)	F(%)
1598	0.95	83.47	46.70	59.89
1602	0.90	83.67	47.27	60.41
1620	0.85	82.92	47.87	60.70
1647	0.80	82.40	48.69	61.21

Table 3: Effect of minimum lambda threshold on performance of rule extraction algorithm on English POS tagging problem

- Half of the extracted rules are never used in the inference procedure. The reason is that these rules are highly specific rules.
- A significant number of rules are subsumptions of some general rules already obtained. Thus, these subsumed rules are redundant and less informative. Hence, we need to eliminate these subsumed rules. We explain this with examples below.

Rule *G*:
If current word is *'.'* then current POS tag is *PUN (punctuation)*

Rule *S*:
If current word is *'.'* and previous word is *come* then current POS tag is *PUN (punctuation)*

Here, rule *G* subsumes rule *S* and rule *S* is redundant where rule *G* is already present. Hence we need to eliminate S.

We experimented with minimum lambda threshold (α) value of 0.1 keeping all other conditions of experiment unchanged and found that number of extracted rules is drastically reduced. The corresponding experimental result is shown in table 3.

Analysis of the extracted rules showed that both the number of specific rules and number of subsumed rules have been significantly reduced. This partially solves the two problems we encounter in previous experiment. Hence, here after in all experiments we have used confidence threshold value of 0.9 and minimum lambda threshold value of 0.1.

In table 5, we give instances of some of the (a)*high quality*, (b)*medium quality* and (C)*low quality* rules obtained at confidence heuristic threshold

#Rules	θ	P(%)	R(%)	F(%)	Fidelity(%)
1573	0.95	77.28	50.62	61.17	74.61
1698	0.90	74.60	54.24	62.81	71.88
1818	0.85	72.22	56.38	63.33	69.45
1937	0.80	71.95	58.74	64.68	69.21
2048	0.75	70.03	60.68	65.02	67.35

Table 4: Effect of confidence threshold on performance of rule extraction algorithm on English chunking problem

Approach	#Rules	P(%)	R(%)	Av. Rule Size
C4.5	66047	82.7	82.7	2.32
PART	429	78.6	78.6	1.09
Rule from CRF	1432	82.82	62.4	1.62

Table 6: Comparison of rule extraction algorithm from CRF with C4.5 decision tree and PART decision list for English POS tagging problem

value of 0.9 and minimum lambda threshold value of 0.1. We tested this rules on gold data and report the count based confidence value, lift value and conviction of each of these rules.

5.3.2 Chunking Experiment

Similarly, we experimented on English chunking problem. We trained a CRF model on 50,000 chunk tagged words and tested the extracted rule base on 40,000 test data. We kept minimum lambda threshold α to be $zero$. The trained CRF model showed 93.99% accuracy on test data. The experimental results are shown in table 4.

The size of the extracted rule base is not very large whereas, the low recall of the rule base indicates its limited scope. Hence, we kept the value of minimum lambda threshold as $zero$.

5.3.3 Comparison with C4.5 Decision Tree and PART Decision List

We compare our approach with the C4.5 decision tree and Weka PART decision list.

We have used 15,000 tagged words from BNC corpus as the training data and have tested on 5000 word. We set pruning confidence factor of C4.5 as low as 0.1 to allow maximum pruning. The results are shown in table 6.

Approach	#Rules	P(%)	R(%)
C4.5	5015	94.80	94.80
PART	471	96.42	96.42
Rule from CRF	1698	74.60	54.24

Table 7: Comparison of rule extraction algorithm from CRF with C4.5 decision tree and PART decision list for English chunking problem

We analyzed the results and observed the following points:

- C4.5 decision tree shows comparatively high precision and recall both but the extremely large size of the extracted rule base makes it incomprehensible. Also, C4.5 fails to extract rules on features of the words.
- PART decision list succeeds to retrieve a smaller rule base with low average rule size. The rules are based on features of the word and seems linguitically interesting. But, the generated rule base is an ordered list of rules. Hence, as we go down the ordered list, the actual number of antecedents of the rules increases rapidly which makes individual rules cumulatively complicated and hard to comprehend.
- The rule base extracted from CRF is both small in size and contains rules on features of the word. Compared to PART decision list, even though our approach retrieves rule base of greater size and lower recall, the unordered nature of the rule base makes it comprehensible and hence, acceptable. Each rule is independent of the other rules, hence, can be separately understood.

Similarly, we carried out experiments on English chunking problem. As specified already, we have used CoNLL 2000 chunking dataset. The results are shown in table 7.

6 Conclusions and Future Works

This paper has presented a simple heuristic based method of rule extraction from a trained CRF. An Empirical threshold based technique is discussed to eliminate highly specific rules and subsumed rules.

Quality	Rule Example	Confidence(%)	Lift	Conviction
High	If current word ends with *'ons'* then current POS tag is *NN2(plural noun)* (for example, *stallions*)	99.61	15.01	245.53
High	If current word ends with *'ical'* then current POS tag is *AJ0(adjective)* (for example, *magical*)	98.96	12.98	88.98
High	If current word ends with *'nal'* then current POS tag is *AJ0(adjective)* (for *vocational*, *additional*)	92.59	12.15	12.47
Medium	If current word starts with '.' then current POS tag is *PUN(punctuation)*	100	9.35	Undefined
Low	If current word starts with *'dur'* then current POS tag is *PRP(preposition)* (exception, *duration, durable*)	96.29	9.85	24.35
Low	If current word starts with *'pain'* then current POS tag is *NN1(noun)* (exception, *painted, painful*)	56.81	3.26	1.91

Table 5: Examples of high, medium and low quality rules obtained for parts of speech tagging problem in English

This reduces the size of the extracted rule base and thus, increases its comprehensibility. Experiments on English POS tagging substantiates this fact. The extracted rule base is found to be more comprehensible than that the rule base obtained by C4.5 decision tree and PART decision list algorithm. The main advantage of our method of rule extraction is that the extracted rule base is unordered and feature rich.

Our method has focused on rule extraction from individual feature function independently. Whereas, it is considered that the combined effect of all feature functions decides the most probable label sequence. Hence, an interesting question for future research is whether we can extract rules from multiple feature functions together. Also, it would be interesting to compare rules from different graphical models.

Throughout our work, we tried to analyze linguistic richness of the extracted rules. An interesting question in the field of rule extraction for natural language processing tusk would be to find a measure of linguistic richness of a rule.

References

Berger A. L. 1997. The Improved Iterative Scaling algorithm: A Gentle Introduction. Unpublished Manuscript

Brill E. 1995. Transformation-based error-driven learning and natural language processing: a case study in part-of-speech tagging. Journal of Computational Linguistics, 543–565.

Frank E., & Witten H. Ian 1998. Generating Accurate Rule Sets Without Global Optimization. *Proceedings of the Fifteenth International Conference on Machine Learning*, 144–151.

Huysmans J., Baesens B., & Vanthienen 1998. Using Rules to Improve Comprehensibility of Predictive Models.

Lafferty J., McCallum A., & Pereira F. 2001. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. *Proceedings of the Eighteenth International Conference on Machine Learning*, 282–289. Morgan kaufmann Publishers Inc.

McCallum A. 2003. Efficiently Inducing Features of Conditional Random Fields. *Uncertainty in Artificial Intelligence-UAI*, 403–410.

Quinlan J. R. 1987. Generating Production Rules from Decision Trees. *Proceedings of the 10th International Joint Conference on Artificial Intelligence*, 304–307. Morgan kaufmann Publishers Inc.

Quinlan J. R. 1986. Induction of Decision Trees. *Journal of Machine Learning*.

Towell G. G., & Shavlik, J., W. 1993. Extracting Refined Rules from Knowledge based Neural Networks. *Machine Learning*, 71–101.

Bayardo Jr., & Agrawal Rakesh 1999. Mining the Most Interesting Rules. *Proceedings of the Fifth ACL SIGKDD International Conference on Knowledge Discovery and Data Mining*, 145–154.

Tibshirani R. 1996. Regression Shrinkage and Selection via the Lasso. *Journal of the Royal Statistical Society. Series B, Vol. 58*, 267–288.

Ratnaparkhi A. 1996. A Maximum Entropy Model for Part-of-Speech Tagging. *Proceedings of the Conference on Empirical Models in Natural Language Processing*, 133–142.

Martens D., Huysmans J., Setiono R., Vanthienen J. & Baesens B. 2008. Rule Extraction from Support Vector Machines: An Overview of Issues and Application in Credit Scoring. *In the Book of Rule Extraction from Support Vector Machines*, 33–63.