Ripple Down Rules for Part-of-Speech Tagging

Dat Quoc Nguyen[1], Dai Quoc Nguyen[1],
Son Bao Pham[1,2], and Dang Duc Pham[1]

[1] Human Machine Interaction Laboratory,
Faculty of Information Technology,
University of Engineering and Technology,
Vietnam National University, Hanoi
{datnq,dainq,sonpb,dangpd}@vnu.edu.vn
[2] Information Technology Institute,
Vietnam National University, Hanoi

Abstract. This paper presents a new approach to learn a rule based system for the task of part of speech tagging. Our approach is based on an incremental knowledge acquisition methodology where rules are stored in an exception-structure and new rules are only added to correct errors of existing rules; thus allowing systematic control of interaction between rules. Experimental results of our approach on English show that we achieve in the best accuracy published to date: 97.095% on the Penn Treebank corpus. We also obtain the best performance for Vietnamese VietTreeBank corpus.

1 Introduction

Part-of-speech (POS) tagging is one of the most important tasks in Natural Language Processing, which assigns a tag representing its lexical category to each word in a text. After the text is tagged or annotated, it can be used in many applications such as: machine translation, information retrieval etc. A number of approaches for this task have been proposed that achieved state-of-the-art results including: Hidden Markov Model-based approaches [1], Maximum Entropy Model-based approaches [2] [3] [4], Support Vector Machine algorithm-based approaches [5], Perceptron learning algorithms [2][6]. All of these approaches are complex statistics-based approaches while the obtained results are progressing to the limit. The combination utilizing the advantages of simple rule-based systems [7] can surpass the limit. However, it is difficult to control the interaction among a large number of rules.

Brill [7] proposed a method to automatically learn transformation rules for the POS tagging problem. In Brill's learning, the selected rule with the highest score is learned on the context that is generated by all preceding rules. In additions, there are interactions between rules with only front-back order, which means an applied back rule will change the results of all the front rules in the whole text. Hepple [8] presented an approach with two assumptions for disabling interactions between rules to reduce the training time while sacrificing a small fall of accuracy.

Ngai and Florian [9] presented a method to impressively reduce the training time by recalculating the score of transformation rules while keeping the accuracy.

In this paper, we propose a failure-driven approach to automatically restructure transformation rules in the form of a Single Classification Ripple Down Rules (SCRDR) tree [10][11][12]. Our approach allows interactions between rules but a rule only changes the results of selected previous rules in a controlled context. All rules are structured in a SCRDR tree, which allows a new exception rule to be added when the system returns an incorrect classification. Moreover, our system can be easily combined with existing part of speech tagger to obtain an even better result. For Vietnamese, we obtained the highest accuracy at present time on VietTreebank corpus [13]. In addition, our approach obtains promising results in term of the training time in comparison with Brill's learning.

The rest of paper is organized as follows: in section 2, we provide some related works including Brill's learning, SCRDR tree, among others and describe our approach in section 3. We describe our experiments in section 4 and discussion in section 5. The conclusion and future works will be presented in section 6.

2 Related Works

2.1 Transformation-Based Learning

The well-known transformation-based error-driven learning method had been introduced by Brill [7] for POS tagging problem and this method has been used in many natural language processing tasks, for example: text chunking, parsing, named entity recognition. The key idea of the method is to compare the golden-corpus that was correctly tagged and the current-corpus created through an initial tagger, and then automatically generate rules to correct errors based on predefined templates. For example, corresponding with a template *"transfer tag of current word from **A** to **B** if the next word is **W**"* is some rules like as: *"transfer tag of current word from **JJ** to **NN** if the next word is **of**"* or *"transfer tag of current word from **VBD** to **VBN** if the next word is **by**"*...

Transformation-based learning algorithm runs in multiple iterations as follows:

- **Input:** Raw-corpus that contains the entire raw text without tags extracted from the golden-corpus that contains manually tagged *word/tag* pairs.
- **Step 1:** Annotated-corpus is generated using an initial tagger where its input is the raw-corpus.
- **Step 2:** Comparing the annotated-corpus and the golden-corpus to determine tag errors in the annotated-corpus. From these errors, all templates are used for creating potential rules.
- **Step 3:** Each rule will be applied to a copy of annotated-corpus. The score of a rule is computed by subtracting of number of additional errors from number of correctly changed tags. The rule with the best score is selected.
- **Step 4:** Update the annotated-corpus by applying selected rule.

- **Step 5:** Stop if the best score is smaller than a predefined threshold T, else repeat step 2.
- **Output:** Front-back ordered list of transformation rules.

The training process of Brill's tagger includes two phases:

- The first-phase is used to assign the most likely tag for unknown words. Initially, the most likely tag for unknown words starting with a capital letter is **NNP** and otherwise it is **NN**. In this phase, the lexical transformation rules are used to predict the most likely tag for unknown words. The transformation templates in this phase depend on character(s), prefix, suffix of a word and only the preceding/following word. For example, *"change the most likely tag of an unknown-word to* $\boldsymbol{Y}$ *if the word has suffix* $\boldsymbol{x}$*,* $|x| <= 4$*"*, *"change the most likely tag of an unknown-word to* $\boldsymbol{Y}$ *if the last (1, 2, 3, 4) characters of the word are* $\boldsymbol{x}$*"* or *"change the most likely tag of an unknown-word to* $\boldsymbol{Y}$ *if the word* $\boldsymbol{x}$ *ever appears immediately to the left/right of the word"*
- The second phase uses transformation-based error-driven learning for producing contextual transformation rules. Each word is assigned a tag by the initial tagger: known-words were annotated with the highest frequency tag using the lexicon extracted from corpus that was used for learning lexical transformation rules, and unknown-words were assigned with default tags **NNP** or **NN** and subsequently the ordered lexical transformation rules were applied.

To tag raw texts, the known-words are assigned by the highest frequency tag using lexicon extracted from the training corpus and unknown-words are assigned with default tags **NNP** or **NN** and then the ordered lexical transformation rules are applied to these unknown-words. Finally, the ordered contextual transformation rules will be applied to all words. In the tagging process, a word can be tagged multiple times. At each the iteration during the training phase, all possible rules will be generated and each rule's score is computed based on the entire corpus. Therefore, training phase in Brill's learning takes a significant amount of time.

The transformation-based learning of Brill allows interactions between learnt rules. A new rule can change the result of any previous rules.

Hepple [8] presented a method to impressively improve about 950 times [9] at the training time while there was a small fall in the precision by using two assumptions: independence and commitment, which disables any interaction between learned rules. The commitment assumption assumes that a tag was changed at most once by a rule in the whole training period. And the independence assumption imposes that if a rule changes a tag, it will not change the context relevant to the firing of a future rule. Ngai and Florian [9] proposed an approach called as Fast TBL to significantly reduce about 340 times at the training time on corpus of about 1 million words while achieved the same accuracy as a standard transformation-based tagger. The central idea of this approach is to save the number of corrected-tags and the number of additional errors for each rule for recalculation when applying a newly selected rule to the current corpus.

Another drawback of Brill's learning is that it is not able to estimate probabilities of class memberships. An approach presented in [14] shows how to convert transformation-based rules list to decision trees for resolving this problem.

2.2 Single Classification Ripple Down Rules

Ripple Down Rules (RDR) [10][15][12] were developed to allow users incrementally add rules to an existing rule-based system whiles systematically controlling interactions between rules and ensuring consistency among existing rules.

Suppose the system's classification produced by some rule ***R*** is deemed incorrect by the expert. As the justification for the decision that the classification is incorrect, the expert creates a new rule ***Re*** which acts as an *exception* to the rule ***R***. The justification would refer to attributes of the case, such as patient data in the medical domain, or a linguistic pattern matching the case in the natural language domain[15].

The new rule ***Re*** will only be applied to cases for which the provided conditions in ***Re*** are true and for which rule ***R*** would produce the classification, if rule ***Re*** had not been entered. In other words, in order for ***Re*** to be applied to a case as an exception rule to ***R***, rule ***R*** has to be satisfied as well. A sequence of nested exception rules, of any depth, may occur. Whenever a new exception rule is added, a difference to the previous rule has to be identified by the expert. This is a natural activity for the expert when justifying his/her decision to colleagues or apprentices. The case which triggered the addition of an exception rule is stored along with the new rule. This case, called the *cornerstone case* of the rule ***R***, is retrieved when an exception to ***R*** needs to be entered. The cornerstone case is intended to assist the expert in coming up with a justification, since a valid justification must point at differences between the cornerstone case and the case at hand for which ***R*** does not perform satisfactorily. A number of RDR-based systems also store with every rule all cases for which the rule has given a correct conclusion. These systems effectively store all *seen cases.* This enables the consistency test to be checked against not only the cornerstone cases but all previously seen cases.

A SCRDR tree [10][12] is a finite binary tree with two distinct types of edges. These edges are typically called *except* and *if not* (or *false*) edges as shown in figure 1. Associated with each node in a tree is a rule. A rule has the form: *if* α *then* β where α is called the *condition* and β the *conclusion.*

An SCRDR tree is evaluated for a case by passing the case to the root of the tree. At any node in the tree, if the condition of a node N's rule is satisfied by the case, the case is passed on to the *except child* of N if it exists. Otherwise, the case is passed on to N's *if not child* if it exists. The conclusion given by this process is the conclusion from the last node in the SCRDR tree which *fired.* To ensure that a conclusion is always given, the root node typically contains a trivial condition which is always satisfied. This node is called the *default node.*

A new rule is added to an SCRDR tree when the evaluation process returns a wrong conclusion using the *fired rule* ***R***. A new node containing the new rule is attached to the last node evaluated in the tree provided the new rule is consistent

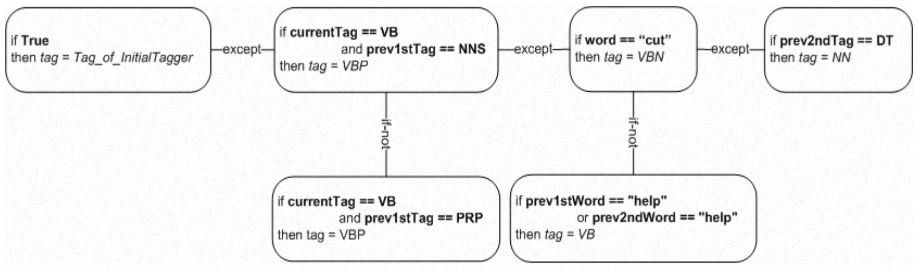
if True
then tag = Tag_of_InitialTagger
except
if currentTag == VB and prev1stTag == NNS
then tag = VBP
except
if word == "cut"
then tag = VBN
except
if prev2ndTag == DT
then tag = NN
if-not
if currentTag == VB and prev1stTag == PRP
then tag = VBP
if-not
if prev1stWord == "help" or prev2ndWord == "help"
then tag = VB

Fig. 1. A part of SCRDR tree for POS tagging

with the existing knowledge base. This is done by making sure cases that have been previously classified correctly by the rule ***R*** do not match the new rule. If the node has no exception link, the new node is attached using an *exception link*, otherwise an *if not link* is used.

2.3 Vietnamese POS Tagging Problems

Dinh and Hoang [16] proposed an approach for Vietnamese POS tagging problem that gave accuracy of 87% by building an English-Vietnamese bilingual corpus which contains approximately 5 million words. They tagged the English corpus using transformation-based learning of Brill [7] and convert POS-annotation tags from English side to Vietnamese using existing word-alignment tools.

Three available tools using machine learning methods: Conditional Random Fields [17], Maximum Entropy Model, Support Vector Machine were used to combine with morpheme-based approach in [18] for Vietnamese POS tagging, that achieved the highest averaged accuracy using the 5-fold cross-validation of 91.64% on Vietnamese Treebank corpus [13].

3 Our Approach

In this section, we describe a transformation-based failure-driven approach to automatically build a single classification Ripple Down Rule (SCRDR) tree for POS tagging problem. Figure 2 describes the learning model used in our approach.

The *Raw corpus* is annotated by using an *Initial tagger* to create the *Annotated corpus*. By comparing the annotated corpus with the *Golden corpus*, an *Object-driven dictionary* is generated based on the *Object Template* which captures the context containing the current word and its tag, (1^{st}, 2^{nd}, 3^{rd}) previous and next words and (1^{st}, 2^{nd}, 3^{rd}) previous and next tags in following format (*previous* 3^{rd} *word, previous* 3^{rd} *tag, previous* 2^{nd} *word, previous* 2^{nd} *tag, previous* 1^{st} *word, previous* 1^{st} *tag, word, currentTag, next* 1^{st} *word, next* 1^{st} *tag, next* 2^{nd} *word, next* 2^{nd} *tag, next* 3^{rd} *word, next* 3^{rd} *tag*) in the annotated corpus.

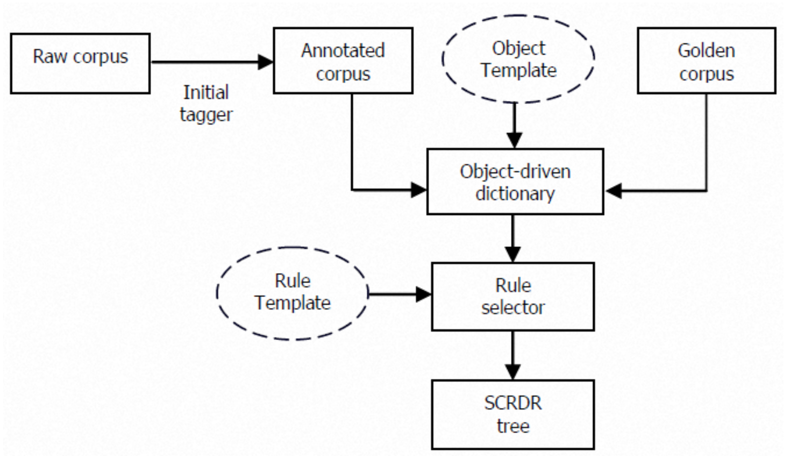

Fig. 2. The diagram describing our approach

An object-driven dictionary is a set of the format (***O****bject, correct Tag*) in which *Object* captures the context of the current word in the annotated corpus and *correct Tag* is the corresponding tag in the golden corpus.

Rule1: *if* {word == **"object.word"**} *then* tag = **"correctTag"**
Rule2: *if* {next1stTag == **"object.next1stTag"**} *then* tag = **"correctTag"**
Rule3: *if* {prev1stTag == **"object.prev1stTag"**} *then* tag = **"correctTag"**

Fig. 3. Some rule examples

From the object template, rule templates are created based on the templates of Brill's tagger for *Rule selector*. Examples of rule templates are shown in figure 3 where elements in bold will be replaced by concrete values for creating concrete rules.

Training algorithm:

– **Step 1:** Load raw corpus and assign initial tags using an initial tagger.
– **Step 2:** Create Object-driven dictionary by comparing output of the initial tagger and the golden corpus.
– **Step 3:** Build the default node representing the initial tagger.
– **Step 4:** At a node-FR in SCRDR tree, let SE be the set of elements from the object-driven dictionary that fired at the node-FR but theirs tags are incorrect i.e. node-FR gives wrong conclusions for elements in SE.

  To select a new exception rule, a list of all concrete rules is generated based on rule-templates from all elements in SE and unsatisfied cornerstone case of node-FR. The rule with the highest value by subtracting B from A would be selected where A is the number of elements in SE that is correctly modified by the rule and B is the number of elements in SE that is incorrectly changed by the rule.

The newly selected rule is added to the SCRDR tree where the cornerstone case is the case in SE that is correctly modified by the selected rule.

This step process is repeated until the score for the selected rule is under a given threshold. At each iteration, a new exception rule is added to correct an error made by the existing rule-based system.

To illustrate how the new exception rules are added, lets consider the following rule (a node in the SCRDR tree)

if currentTag == **"vb"** and prev1stTag == **"nns"** *then* tag = **"vbp"**

cc: *('the', 'dt', 'latest', 'jjs', 'results',* ***'nns'****, 'appear',* ***'vb'****, 'in', 'in', 'today', 'nn', ''s', 'pos')*

Suppose we have a case that this rule fires but returns a wrong conclusion i.e. incorrect tag. The following rule can be added as an exception rule of the rule in the SCRDR tree with the cornerstone case (cc) being the case that was misclassified originally:

if word == **"cut"** *then* tag = **"vbn"**

cc: *('keeping', 'vbg', 'their', 'prp$', 'people',* ***'nns'****,* ***'cut'****,* ***'vb'****, 'off', 'rp', 'from', 'in', 'the', 'dt')*

To take a further example, suppose we have a new case that the above newly added rule fires but the conclusion is incorrect. The following exception is added to correct the mistake:

if prev2ndTag == **"dt"** *then* tag = **"nn"**

cc: *('to', 'to', 'the',* ***'dt'****, 'capital-gains',* ***'nns'****,* ***'cut'****,* ***'vb'****, ',', ',', 'which', 'wdt', 'has', 'vbz')*

Tagging process:

– Raw texts are tagged by initial tagger to create the annotated texts.
– Make objects to capture context surrounding current *word/tag* in annotated texts.
– Each object is classified by SCRDR tree for generating output tag.

In our method, we use two thresholds: one for finding rules for nodes at the depth of 1 and the other is used for nodes at higher levels. One reason is that the default node has no cornerstone case.

4 Experiment

We apply our approach to both English and Vietnamese part of speech tagging tasks.

4.1 Results for English

Following [2] [3] [4] [5] [6], we split the Penn Wall Street Journal Treebank [19] into training, development and test sets as shown in table 1 for our experiments for English. We retrained Brill's tagger on training data at default threshold of 2 resulting in 1595 rules with the time cost for learning contextual transformation rules of 2700 minutes.

Table 1. Data Set

DataSet	Sections	Sentences	Tokens
Training	0-18	38,219	912,344
Develop	19-21	5,527	131,768
Test	22-24	5,462	129,654

For our method, RDR tree was built on whole training data using Brill's retrained initial tagger as the initial tagger that achieved the baseline accuracies of 93.67% and 93.58% on development data and test data respectively.

Brill's retrained tagger achieved an accuracy of 96.57% on development data while the result of our taggers on development data is shown in table 2. The accuracy is comparable while our method improves up to 33 times in training time.

Table 2. Pos tagging in accuracy of development data of our approach

Threshold	Number of rules	Accuracy (%)	Training time (minutes)
(50, 20)	133	95.76	14
(10, 10)	393	96.21	30
(5, 5)	830	96.42	48
(3, 2)	**2517**	**96.55**	**82**
(1, 1)	18310	96.35	512

Table 3 shows the performance for our method using the best threshold and Brill's tagger on test data.

Table 3. Pos tagging accuracy on test data

Method	Accuracy (%)
Brill	96.530
Our approach	96.548

Table 4 shows the accuracy of our method depending on the depth of the RDR tree.

Table 4. Accuracy and speed tagging in our method on test data on PenIV 2.66GHZ of CPU, 1G of RAM

Depth	Number of rules	Accuracy (%)	Speed tagging (number of words second)
<= 1	1433	96.372	161
<= 2	2467	96.540	160
<= 3	2517	96.548	160

Table 5. Accuracy of our method with different initial taggers on test data

Initial Tagger (IT)	Accuracy of IT (%)	Accuracy of IT and RDR tree(%)	Number of rules in RDR tree
Brill's tagger	96.53	96.68	322
Tagger of Tsuruoka and Tsujii	96.987	97.095	130

Table 5 shows the returned results when we used Brill's retrained tagger and tagger of Tsuruoka and Tsujii [4] that was trained on same WSJ 0-18 training data at default parameters as initial taggers for building an RDR tree in our approach. It can be seen that our approach can be used to improve performance of existing approaches by adding more exception rules.

4.2 Results for Vietnamese

We ran experiments for Vietnamese on the same corpus as in [18] on Vietnamese Treebank corpus [13]. This corpus contains approximately 10000 sentences with a tag set of 17 labels. We randomly divide the corpus into five folds; giving one fold size of around 44±1K words. Each time, four folds were merged as the training set and the remaining fold is selected as the test set. Final result is the averaged results of five runs using the best threshold found for the English experiment.

Table 6 shows the result of our approach, which we use an open dictionary assigning the most frequent tag in whole training set for a word as the initial tagger. For this open dictionary assumption, when a word (in test set) not in the dictionary, it would be tagged as Np if the first character is an upper letter or N if otherwise by default. With the open dictionary assumption, the accuracy of the initial tagger is 90.38%.

Table 6. Five-fold cross validation accuracy of our method in open dictionary assumption

Method	Final accuracy (%)
Open dict	92.24

From table 6, it can be seen that our method achieves a higher accuracy than accuracy of 91.64% of the method described in [18].

We also trained our method using the closed dictionary assumption where the initial tagger assigns a word with the most frequent tag that extracted from whole training and test sets. In addition, we retrained Brill's tagger for Vietnamese using the closed dictionary assumption without the process of learning lexical transformation rules. Both of methods obtained an accuracy of 92.81% at the initial state.

Table 7 shows the results for our approach and Brill's tagger in closed dictionary assumption.

Table 7. Five-fold cross validation accuracy in closed dictionary assumption

Method	Final accuracy (%)
Brill's tagger	94.72
Our method	94.61

It can be seen that our approach is comparable to that of Brill's in this corpus. Due to the small size of this corpus, our approach can utilize experts to add rules instead of learning rule from the corpus.

5 Discussion

For Brill's approach and Hepple's approach, a new rule is selected based on the accuracy of context, so the output of initial tagger is always changed when the new rule applies. In our approach, objects always are static so that new rules are selected based on the original state of the output of the initial tagger. Therefore our approach can be easily combined with existing state of the art taggers to improve their performance as demonstrated when we improve the existing result of Tsuruoka and Tsujii [4].

Another important point is that our approach is very suitable to use experts to add new exception rules given a concrete case at hand that is misclassified by the system. This is especially important for under-resourced languages where obtaining a large annotated corpus is difficult.

6 Conclusion

In this paper, we propose a failure-driven approach to automatically restructure transformation rules in the form of a Single Classification Ripple Down Rules tree. Our approach allows controlled interactions between rules where a rule only changes the results of a limited number of other rules. On the Penn Treebank, our approach achieves the best performance published to date of 97.095%. For Vietnamese, our approach achieves an accuracy of 92.24% for open-dictionary assumption and an accuracy of 94.61% for close-dictionary assumption. This is also the best result to date to the best of our knowledge.

In the future we will involve experts to manually add more exception rules to the current rule based system to even improve the performance of the system further.

Another avenue is to use Ngai and Florian's method [9] to improve the training time and extend the lexical transformation rule learning of Brill for Vietnamese.

Acknowledgment

This work is partly supported by the research project No. QG.10.39 granted by Vietnam National University, Hanoi and IBM Faculty Award 2009 for the third author.

References

1. Brants, T.: Tnt – a statistical part-of-speech tagger. In: Proc. ANLP, pp. 224–231 (2000)
2. Collins, M.: Discriminative training methods for hidden markov models: theory and experiments with perceptron algorithms. In: Proc. EMNLP, pp. 1–8 (2002)
3. Toutanova, K., Klein, D., Manning, C.D., Singer, Y.: Feature-rich part-of-speech tagging with a cyclic dependency network. In: Proc. NAACL-HLT, pp. 173–180 (2003)
4. Tsuruoka, Y., Tsujii, J.: Bidirectional inference with the easiest-first strategy for tagging sequence data. In: Proc. HLT-EMNLP, pp. 467–474 (2005)
5. Giménez, J., Màrquez, L.: Svmtool: A general pos tagger generator based on support vector machines. In: Proc. LREC, pp. 43–46 (2004)
6. Shen, L., Satta, G., Joshi, A.: Guided learning for bidirectional sequence classification. In: Proc. ACL, pp. 760–767 (2007)
7. Brill, E.: Transformation-based error-driven learning and natural language processing: a case study in part-of-speech tagging. Computational Linguistics 21(4), 543–565 (1995)
8. Hepple, M.: Independence and commitment: assumptions for rapid training and execution of rule-based pos taggers. In: Proc. ACL, pp. 278–277 (2000)
9. Ngai, G., Florian, R.: Transformation-based learning in the fast lane. In: Proc. NAACL, pp. 1–8 (2001)
10. Compton, P., Jansen, R.: Knowledge in context: a strategy for expert system maintenance. In: Proc. AI 1988, pp. 292–306 (1988)
11. Compton, P., Jansen, R.: A philosophical basis for knowledge acquisition. Knowl. Acquis. 2(3), 241–257 (1990)
12. Richards, D.: Two decades of ripple down rules research. Knowl. Eng. Rev. 24(2), 159–184 (2009)
13. Nguyen, P.T., Vu, X.L., Nguyen, T.M.H., Nguyen, V.H., Le, H.P.: Building a large syntactically-annotated corpus of vietnamese. In: Proc. LAW, pp. 182–185 (2009)
14. Florian, R., Henderson, J.C., Ngai, G.: Coaxing confidences from an old friend: probabilistic classifications from transformation rule lists. In: Proc. EMNLP, pp. 26–34 (2000)
15. Pham, S.B., Hoffmann, A.: Efficient knowledge acquisition for extracting temporal relations. In: Proc. ECAI, pp. 521–525 (2006)

16. Dien, D., Kiem, H.: Pos-tagger for english-vietnamese bilingual corpus. In: Proc. HLT-NAACL WBT, pp. 88–95 (2003)
17. Lafferty, J.D., McCallum, A., Pereira, F.C.N.: Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In: Proc. ICML, pp. 282–289 (2001)
18. Tran, O.T., Le, C.A., Ha, T.Q., Le, Q.H.: An experimental study on vietnamese pos tagging. In: Proc. IALP, pp. 23–27 (2009)
19. Marcus, M.P., Marcinkiewicz, M.A., Santorini, B.: Building a large annotated corpus of english: the penn treebank. Computational Linguistics 19(2), 313–330 (1993)