RDRPOSTagger: A Ripple Down Rules-based Part-Of-Speech Tagger

Dat Quoc Nguyen[1] and **Dai Quoc Nguyen**[1] and **Dang Duc Pham**[2] and **Son Bao Pham**[1]
[1] Faculty of Information Technology
University of Engineering and Technology
Vietnam National University, Hanoi
{datnq, dainq, sonpb}@vnu.edu.vn
[2] L3S Research Center, Germany
pham@L3S.de

Abstract

This paper describes our robust, easy-to-use and language independent toolkit namely RDRPOSTagger which employs an error-driven approach to automatically construct a Single Classification Ripple Down Rules tree of transformation rules for POS tagging task. During the demonstration session, we will run the tagger on data sets in 15 different languages.

1 Introduction

As one of the most important tasks in Natural Language Processing, Part-of-speech (POS) tagging is to assign a tag representing its lexical category to each word in a text. Recently, POS taggers employing machine learning techniques are still mainstream toolkits obtaining state-of-the-art performances[1]. However, most of them are time-consuming in learning process and require a powerful computer for possibly training machine learning models.

Turning to rule-based approaches, the most well-known method is proposed by Brill (1995). He proposed an approach to automatically learn transformation rules for the POS tagging problem. In the Brill's tagger, a new selected rule is learned on a context that is generated by all previous rules, where a following rule will modify the outputs of all the preceding rules. Hence, this procedure returns a difficulty to control the interactions among a large number of rules.

Our RDRPOSTagger is presented to overcome the problems mentioned above. The RDRPOSTagger exploits a failure-driven approach to automatically restructure transformation rules in the form of a Single Classification Ripple Down Rules (SCRDR) tree (Richards, 2009). It accepts interactions between rules, but a rule only changes the outputs of some 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 tree returns an incorrect classification. A specific description of our new RDRPOSTagger approach is detailed in (Nguyen et al., 2011).

Packaged in a 0.6MB zip file, implementations in Python and Java can be found at the tagger's website *http://rdrpostagger.sourceforge.net/*. The following items exhibit properties of the tagger:

• The RDRPOSTagger is easy to configure and train. There are only two threshold parameters utilized to learn the rule-based model. Besides, the tagger is very simple to use with standard input and output, having clear usage and instructions available on its website.

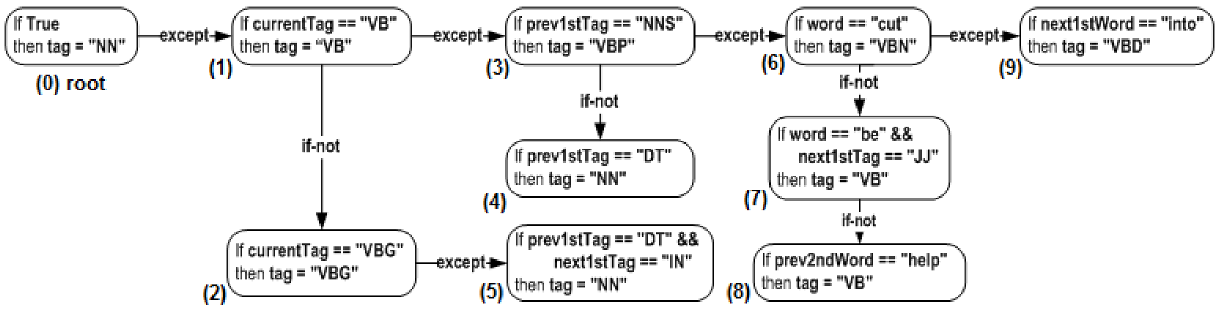
• The RDRPOSTagger is language independent. This POS tagging toolkit has been successfully applied to English and Vietnamese. To train the toolkit for other languages, users just provide a lexicon of words and the most frequent associated tags. Moreover, it can be easily combined with existing POS taggers to reach an even better result.

• The RDRPOSTagger obtains very competitive accuracies. On Penn WSJ Treebank corpus (Marcus et al., 1993), taking WSJ sections 0-18 as the training set, the tagger achieves a competitive performance compared to other state-of-the-art English POS taggers on the test set of WSJ sections 22-24. For Vietnamese, it outperforms all previous machine learning-based POS tagging systems to obtain an up-to-date highest result on the Vietnamese Treebank corpus (Nguyen et al., 2009).

• The RDRPOSTagger is fast. For instance in English, the time[2] taken to train the tagger on the WSJ sections 0-18 is **40** minutes. The tagging speed on the test set of the WSJ sections 22-24 is **2800** words/second accounted for the latest implementation in Python whilst it is **92k** words/second

[1] http://aclweb.org/aclwiki/index.php?title=POS_Tagging_(State_of_the_art)

[2] Training and tagging times are computed on a Windows-7 OS computer of Core 2Duo 2.4GHz & 3GB of memory.



If True then tag = "NN"
(0) root
except
If currentTag == "VB" then tag = "VB"
(1)
if-not
If currentTag == "VBG" then tag = "VBG"
(2)
If prev1stTag == "NNS" then tag = "VBP"
(3)
If prev1stTag == "DT" then tag = "NN"
(4)
If prev1stTag == "DT" && next1stTag == "IN" then tag = "NN"
(5)
If word == "cut" then tag = "VBN"
(6)
If word == "be" && next1stTag == "JJ" then tag = "VB"
(7)
If prev2ndWord == "help" then tag = "VB"
(8)
If next1stWord == "into" then tag = "VBD"
(9)

Figure 1: A part of our SCRDR tree for English POS tagging.

for the implementation in Java.

2 SCRDR methodology

A SCRDR tree (Richards, 2009) is a binary tree with two distinct types of edges. These edges are typically called *except* and *if-not* edges. 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 is referred to as the *conclusion*.

Cases in SCRDR are evaluated by passing a 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 exception child of N using the *except* link if it exists. Otherwise, the case is passed on to the N's *if-not* child. The conclusion given by this process is the conclusion from the last node in the SCRDR tree which *fired* (satisfied by the case). 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 node containing a new rule (i.e. a new exception rule) is added to an SCRDR tree when the evaluation process returns the *wrong* conclusion. The new node is attached to the last node in the evaluation path of the given case with the *except* link if the last node is the *fired* one. Otherwise, it is attached with the *if-not* link.

For example with the SCRDR tree in the figure 1, given a case *"as/IN investors/NNS anticipate/VB a/DT recovery/NN"* where *"anticipate/VB"* is the current word and tag pair, the case satisfies the conditions of the rules at nodes (0), (1) and (3), it then is passed to the node (6) (utilizing except links). As the case does not satisfy the condition of the rule at node (6), it will be transferred to node (7) using if-not link. Since the case does not fulfill the conditions of the rules at nodes (7) and (8), we have the evaluation path (0)-(1)-(3)-(6)-(7)-(8) with fired node (3). Therefore, the tag for *"anticipate"* is concluded as "VBP".

Rule (1) - the rule at node (1) - is the exception rule[3] of the default rule (0). As node (2) is the if-not child node of the node (1), the associated rule (2) is also an exception rule of the rule (0). Similarly, both rules (3) and (4) are exception rules of the rule (1) whereas all rules (6), (7) and (8) are exception rules of the rule (3), and so on. Thus, the exception structure of the SCRDR tree extends to 4 levels: rules (1) and (2) at layer 1, rules (3), (4) and (5) at layer 2, rules (6), (7) and (8) at layer 3, and rule (9) at layer 4.

3 The RDRPOSTagger toolkit

The toolkit consists of four main components: Utility, Initial-tagger, SCRDR-learner and SCRDR-tagger.

3.1 The Utility

The major functions of this component are to evaluate tagging performances (displaying accuracy results), and to create a lexicon of words and the most frequent associated tags as well as to extract *Raw corpus* from an input golden training corpus.

3.2 The Initial-tagger

The initial-tagger developed in the RDRPOSTagger toolkit is based on the lexicon which is generated in the use of the Utility component to assign a tag for each word. To deal with unknown words, the initial-tagger utilizes several regular expressions or heuristics for English and Vietnamese whereas the most frequent tag in the training corpus is exploited to label unknown-words when adapting to other languages.

3.3 The SCRDR-learner

The SCRDR-learner component uses a failure-driven method to automatically build a SCRDR tree of transformation rules. Figure 3 describes the learning process of the learner.

[3]The default rule is the unique rule which is not an exception rule of any other rule. Every rule in layer n is an exception rule of a rule in layer $n - 1$.

#12: *if* next1stTag == **"object.next1stTag"** *then* tag = **"correctTag"**
#14: *if* prev1stTag == **"object.prev1stTag"** *then* tag = **"correctTag"**
#18: *if* word == **"object.word"** && next1stTag == **"object.next1stTag"** *then* tag = **"correctTag"**

Figure 2: Rule template examples.

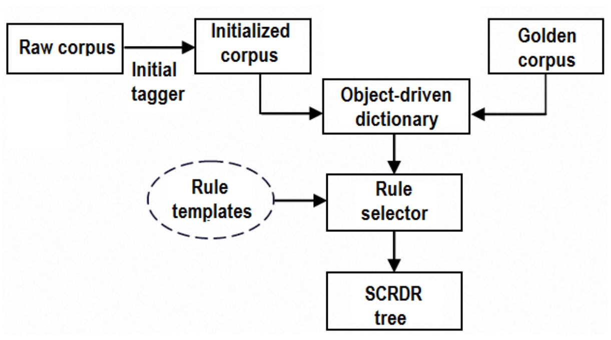


Figure 3: The diagram of the learning process of the learner.

The *Initialized corpus* is returned by performing the Initial-tagger on the Raw corpus. By comparing the initialized one with the *Golden corpus*, an *Object-driven dictionary* of pairs (***O**bject, correctTag*) is produced in which *Object* captures the 5-word window context covering the current word and its tag in following format (*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*) from the initialized corpus, and the *correctTag* is the corresponding tag of the current word in the golden corpus.

There are 27 *Rule templates* applied for *Rule selector* which is to select the most suitable rules to build the *SCRDR tree*. Examples of the rule templates are shown in figure 2 where elements in bold will be replaced by concrete values from *Object*s in the object-driven dictionary to create concrete rules. The SCRDR tree of rules is initialized by building the default rule and all exception rules of the default one in form of *if currentTag = "**TAG**" then tag = "**TAG**"* at the layer-1 exception structure, for example rules (1) and (2) in the figure 1, and the like. The learning approach to construct new exception rules to the tree is as follows:

• At a node-F in the SCRDR tree, let SO be the set of Objects from the object-driven dictionary, which those Objects are fired at the node-F but their initialized tags are incorrect (the *currentTag* is not the *correctTag* associated). It means that node-F gives wrong conclusions to all Objects in the SO set.

• In order to select a new exception rule of the rule at node-F from all concrete rules which are generated for all Objects in the SO set, the selected rule have to satisfy constraints: (i) The rule must be unsatisfied by cases for which node-F has already given correct conclusions. This constraint does not apply to node-F at layer-1 exception structure. (ii) The rule must associate to a highest score value of subtracting B from A in comparison to other ones, where A and B are the numbers of the SO's Objects which are correctly and incorrectly concluded by the rule respectively. (iii) And the highest value is not smaller than a given threshold.

The SCRDR-learner applies two threshold parameters: first threshold is to choose exception rules at the layer-2 exception structure (e.g rules (3), (4) and (5) in figure 1), and second threshold is to select rules for higher exception layers.

• The process to add new exception rules is repeated until there is no rule satisfying the constraints above. At each iteration, a new rule is added to the current SCRDR tree to correct error conclusions made by the tree.

3.4 The SCRDR-tagger

The SCRDR-tagger component is to perform the POS tagging on a raw text corpus where each line is a sequence of words separated by white space characters. The component labels the text corpus by using the Initial-tagger. It slides due to a left-to-right direction on a 5-word window context to generate a corresponding Object for each initially tagged word. The Object is then classified by the learned SCRDR tree model to produce final conclusion tag of the word as illustrated in the example in the section 2.

4 Evaluation

The RDRPOSTagger has already been successfully applied to English and Vietnamese corpora.

4.1 Results for English

Experiments for English employed the Penn WSJ Treebank corpus to exploit the WSJ sections 0-18 (38219 sentences) for training, the WSJ sections 19-21 (5527 sentences) for validation and the WSJ sections 22-24 (5462 sentences) for test.

Using a lexicon created in the use of the training set, the Initial-tagger obtains an accuracy of 93.51% on the test set. By varying the thresholds on the validation set, we have found the most suitable values[4] of 3 and 2 to be used for evaluating the RDRPOSTagger on the test set. Those thresholds return a SCRDR tree model of 2319 rules in a 4-level exception structure. The training time and tagging speed for those thresholds are mentioned in the introduction section. On the same test set, the RDRPOSTagger achieves a performance at 96.49% against 96.46% accounted for the state-of-the-art POS tagger TnT (Brants, 2000).

For another experiment, only in training process: 1-time occurrence words in training set are initially tagged as out-of-dictionary words. With a learned tree model of 2418 rules, the tagger reaches an accuracy of 96.51% on the test set.

Retraining the tagger utilizing another initial tagger[5] developed in the Brill's tagger (Brill, 1995) instead of the lexicon-based initial one, the RDRPOSTagger gains an accuracy result of 96.57% which is slightly higher than the performance at 96.53% of the Brill's.

4.2 Results for Vietnamese

In the first Evaluation Campaign[6] on Vietnamese Language Processing, the POS tagging track provided a golden training corpus of 28k sentences (631k words) collected from two sources of the national VLSP project and the Vietnam Lexicography Center, and a raw test corpus of 2100 sentences (66k words). The training process returned a SCRDR tree of 2896 rules[7]. Obtaining a highest performance on the test set, the RDRPOSTagger surpassed all other participating systems.

We also carry out POS tagging experiments on the golden corpus of 28k sentences and on the Vietnamese Treebank of 10k sentences (Nguyen et al., 2009) according to 5-fold cross-validation scheme[8]. The average accuracy results are presented in the table 1. Achieving an accuracy of 92.59% on the Vietnamese Treebank, the RDRPOSTagger outperforms previous Maximum Entropy Model, Conditional Random Field and Support Vector Machine-based POS tagging systems (Tran et al., 2009) on the same evaluation scheme.

Table 1: Accuracy results for Vietnamese

Corpus	Initial-tagger	RDRPOSTagger
28k	91.18%	93.42%
10k	90.59%	92.59%

5 Demonstration and Conclusion

In addition to English and Vietnamese, in the demonstration session, we will present promising experimental results and run the RDRPOSTagger for other languages including Bulgarian, Czech, Danish, Dutch, French, German, Hindi, Italian, Lao, Portuguese, Spanish, Swedish and Thai. We will also let the audiences to contribute their own data sets for retraining and testing the tagger.

In this paper, we describe the rule-based POS tagging toolkit RDRPOSTagger to automatically construct transformation rules in form of the SCRDR exception structure. We believe that our robust, easy-to-use and language-independent toolkit RDRPOSTagger can be useful for NLP/CL-related tasks.

[4]The thresholds 3 and 2 are reused for all other experiments in English and Vietnamese.

[5]The initial tagger gets a result of 93.58% on the test set.

[6]http://uet.vnu.edu.vn/rivf2013/campaign.html

[7]It took 100 minutes to construct the tree leading to tagging speeds of 1100 words/second and 45k words/second for the implementations in Python and Java, respectively, on the computer of Core 2Duo 2.4GHz & 3GB of memory.

[8]In each cross-validation run, one fold is selected as test set, 4 remaining folds are merged as training set. The initial tagger exploits a lexicon generated from the training set. In training process, 1-time occurrence words are initially labeled as out-of-lexicon words.

References

Thorsten Brants. 2000. TnT: a statistical part-of-speech tagger. In *Proc. of 6th Applied Natural Language Processing Conference*, pages 224–231.

Eric Brill. 1995. Transformation-based error-driven learning and natural language processing: a case study in part-of-speech tagging. *Comput. Linguist.*, 21(4):543–565.

Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of English: the penn treebank. *Comput. Linguist.*, 19(2):313–330.

Phuong Thai Nguyen, Xuan Luong Vu, Thi Minh Huyen Nguyen, Van Hiep Nguyen, and Hong Phuong Le. 2009. Building a Large Syntactically-Annotated Corpus of Vietnamese. In *Proc. of LAW III workshop*, pages 182–185.

Dat Quoc Nguyen, Dai Quoc Nguyen, Son Bao Pham, and Dang Duc Pham. 2011. Ripple Down Rules for Part-of-Speech Tagging. In *Proc. of 12th CICLing - Volume Part I*, pages 190–201.

Debbie Richards. 2009. Two decades of ripple down rules research. *Knowledge Engineering Review*, 24(2):159–184.

Oanh Thi Tran, Cuong Anh Le, Thuy Quang Ha, and Quynh Hoang Le. 2009. An experimental study on vietnamese pos tagging. *Proc. of the 2009 International Conference on Asian Language Processing*, pages 23–27.