A Fast and Accurate Vietnamese Word Segmenter

Dat Quoc Nguyen[1], **Dai Quoc Nguyen**[2], **Thanh Vu**[3], **Mark Dras**[4], **Mark Johnson**[4]
[1]The University of Melbourne, Australia; [2]Deakin University, Australia;
[3]Newcastle University, United Kingdom; [4]Macquarie University, Australia
dqnguyen@unimelb.edu.au, dai.nguyen@deakin.edu.au,
thanh.vu@newcastle.ac.uk, {mark.dras, mark.johnson}@mq.edu.au

Abstract

We propose a novel approach to Vietnamese word segmentation. Our approach is based on the Single Classification Ripple Down Rules methodology (Compton and Jansen, 1990), where rules are stored in an exception structure and new rules are only added to correct segmentation errors given by existing rules. Experimental results on the benchmark Vietnamese treebank show that our approach outperforms previous state-of-the-art approaches JVnSegmenter, vnTokenizer, DongDu and UETsegmenter in terms of both accuracy and performance speed. Our code is open-source and available at: `https://github.com/datquocnguyen/RDRsegmenter`.

Keywords: Vietnamese, Word segmentation, Single classification ripple down rules

1. Introduction

Word segmentation is referred to as an important first step for Vietnamese NLP tasks (Dien et al., 2001; Ha, 2003; Duc Cong et al., 2016). Unlike English, white space is a weak indicator of word boundaries in Vietnamese because when written, it is also used to separate syllables that constitute words. For example, a written text "thuế thu nhập cá nhân" (individual$_{\text{cá\_nhân}}$ income$_{\text{thu\_nhập}}$ tax$_{\text{thuế}}$) consisting of 5 syllables forms a two-word phrase "thuế_thu_nhập cá_nhân."[1] More specifically, about 85% of Vietnamese word types are composed of at least two syllables and 80%+ of syllable types are words by themselves (Thang et al., 2008; Le et al., 2008), thus creating challenges in Vietnamese word segmentation (Nguyen et al., 2012).

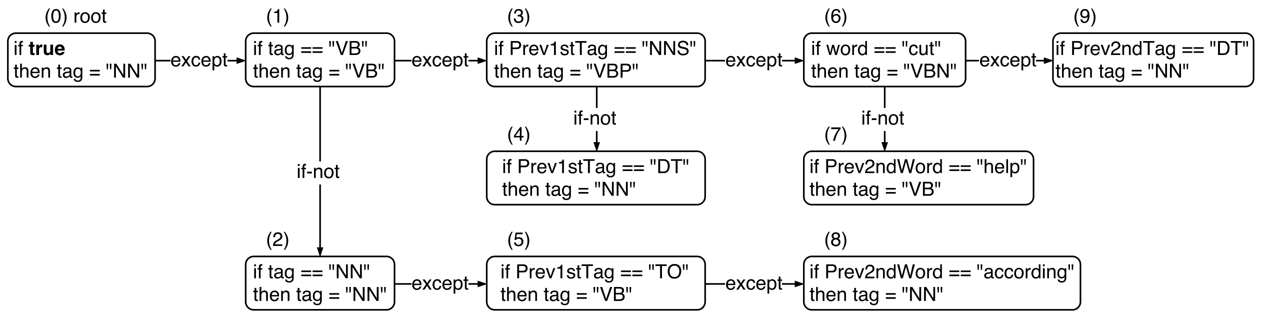
Many approaches are proposed for the Vietnamese word segmentation task. Le et al. (2008), Pham et al. (2009) and Tran et al. (2012) applied the maximum matching strategy (NanYuan and YanBin, 1991) to generate all possible segmentations for each input sentence; then to select the best segmentation, Le et al. (2008) and Tran et al. (2012) used n-gram language models while Pham et al. (2009) employed part-of-speech (POS) information from an external POS tagger. In addition, Nguyen et al. (2006), Dinh and Vu (2006) and Tran et al. (2010) considered this segmentation task as a sequence labeling task, using either a linear-chain CRF, SVM or MaxEnt model to assign each syllable a segmentation tag such as B (Begin of a word) or I (Inside of a word). Another promising approach is joint word segmentation and POS tagging (Takahashi and Yamamoto, 2016; Nguyen et al., 2017b), which assigns a combined segmentation and POS tag to each syllable. Furthermore, Luu and Kazuhide (2012), Liu and Lin (2014) and Nguyen and Le (2016) proposed methods based on pointwise prediction (Neubig and Mori, 2010), where a binary classifier is trained to identify whether or not there is a word boundary between two syllables.

In this paper, we propose a novel method to Vietnamese word segmentation. Our method automatically constructs a Single Classification Ripple Down Rules (*SCRDR*) tree (Compton and Jansen, 1990) to correct wrong segmentations given by a longest matching-based word segmenter. On the benchmark Vietnamese treebank (Nguyen et al., 2009), experimental results show that our method obtains better accuracy and performance speed than the previous state-of-the-art methods JVnSegmenter (Nguyen et al., 2006), vnTokenizer (Le et al., 2008), DongDu (Luu and Kazuhide, 2012) and UETsegmenter (Nguyen and Le, 2016).

2. SCRDR methodology

This section gives a brief introduction of the SCRDR methodology (Compton and Jansen, 1988; Compton and Jansen, 1990; Richards, 2009). A SCRDR tree is a binary tree with only two unique types of edges "except" and "if-not", where every node is associated with a *rule* in a form of "*if* **condition** *then* **conclusion**." To ensure that the tree always produces a conclusion, the rule at its root (default) node has a trivial condition which is always satisfied.

[1]In the traditional underscore-based representation in the Vietnamese word segmentation task (Nguyen et al., 2009), white space is only used to separate words while underscore is used to separate syllables inside a word.



(0) root
if true then tag = "NN"
except
(1)
if tag == "VB" then tag = "VB"
except
(3)
if Prev1stTag == "NNS" then tag = "VBP"
except
(6)
if word == "cut" then tag = "VBN"
except
(9)
if Prev2ndTag == "DT" then tag = "NN"
if-not
(4)
if Prev1stTag == "DT" then tag = "NN"
if-not
(7)
if Prev2ndWord == "help" then tag = "VB"
if-not
(2)
if tag == "NN" then tag = "NN"
except
(5)
if Prev1stTag == "TO" then tag = "VB"
except
(8)
if Prev2ndWord == "according" then tag = "NN"

Figure 1: An illustration of a SCRDR tree for POS tagging. This figure is adapted from Nguyen et al. (2016).

Each case to be evaluated starts at the root node and ripples down as follows: (i) If the case satisfies the condition of a current node's rule, the case is then passed on to the current node's "except" child if this "except" child exists. (ii) Otherwise, if the case does not satisfy the condition, it is then passed on to the current node's "if-not" child. So, the conclusion returned by the tree is the conclusion of the last satisfied rule in the evaluation path to a leaf node.

For example, Figure 1 illustrates a SCRDR tree for POS tagging. Let us consider a concrete case "*as/IN investors/NNS anticipate/VB a/DT recovery/NN*" where "*anticipate*" and "*VB*" is the current considered pair of word and its initial POS tag. Because this case satisfies the conditions of the rules at nodes (0), (1) and (3), it is passed on to node (6) using the "except" edge. The case does not satisfy the condition of the rule at node (6), thus it is passed on to node (7) using the "if-not" edge. Finally, the case does not satisfy the condition of the rule at the leaf node (7). So, the rule at node (3)—the last satisfied rule in the the evaluation path (0)-(1)-(3)-(6)-(7)—concludes "*VBP*" should be the POS tag of the word "*anticipate*" instead of the initial POS tag "*VB*."

To correct a wrong conclusion returned for a given case, a new node containing a new *exception* rule may be attached to the last node in the evaluation path. If the last node's rule is the last satisfied rule given the case, the new node is added as its child with the "except" edge; otherwise, the new node is attached with the "if-not" edge.

SCRDR has been successfully applied in NLP tasks for temporal relation extraction (Pham and Hoffmann, 2006), word lemmatization (Plisson et al., 2008), POS tagging (Xu and Hoffmann, 2010; Nguyen et al., 2011b; Nguyen et al., 2014; Nguyen et al., 2016), named entity recognition (Nguyen and Pham, 2012) and question answering (Nguyen et al., 2011a; Nguyen et al., 2013; Nguyen et al., 2017a). The works by Plisson et al. (2008), Nguyen et al. (2011b), Nguyen et al. (2014) and Nguyen et al. (2016) build the tree automatically, while others manually construct the tree.

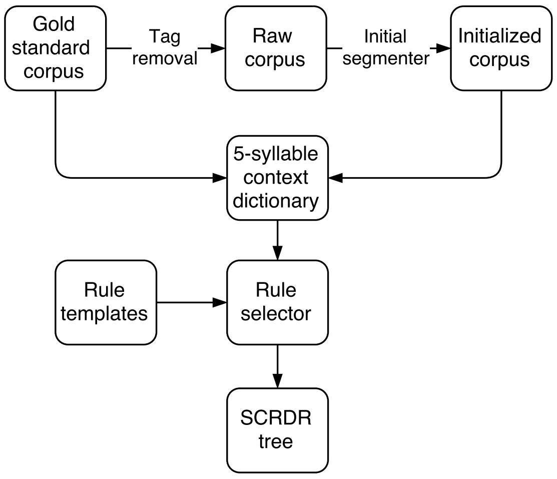


Figure 2: Diagram of our approach.

3. Our approach

This section describes our new error-driven approach to automatically construct a SCRDR tree to correct wrong segmentations produced by an initial word segmenter.

Following Nguyen et al. (2006) and Tran et al. (2010), we also formalize the word segmentation problem as a sequence labeling task. In particular, each syllable is labeled by either segmentation tag B (Begin of a word) or I (Inside of a word). As a result, our approach can be viewed as an extension to word segmentation of the automatic SCRDR approach for POS tagging (Nguyen et al., 2014; Nguyen et al., 2016). Our learning diagram is described in Figure 2.

We start with an underscore-based *gold standard training corpus* consisting of manually word-segmented sentences, e.g. "thuế_thu_nhập cá_nhân" (individual$_{\text{cá\_nhân}}$ income$_{\text{thu\_nhập}}$ tax$_{\text{thuế}}$) and transform this corpus into a BI-formed representation (e.g. "thuế/B thu/I nhập/I cá/B nhân/I"). We then extract syllables to construct the *raw corpus* (which does not have B and I segmentation tags, and would look like "thuế thu nhập cá nhân").

Tuple as key	Value	
(“”, “”, “”, “”, **thuế**, **B**, thu, B, nhập, I)	B	√
(“”, “”, thuế, B, **thu**, **B**, nhập, I, cá, B)	I	X
(thuế, B, thu, B, **nhập**, **I**, cá, B, nhân, I)	I	√
(thu, B, nhập, I, **cá**, **B**, nhân, I, “”, “”)	B	√
(nhập, I, cá, B, **nhân**, **I**, “”, “”, “”, “”)	I	√

Table 1: Examples of key-value pairs in the 5-syllable context dictionary $\mathcal{D}$ when comparing the BI-formed gold standard corpus “thuế/B thu/I nhập/I cá/B nhân/I” and the BI-formed initialized corpus “thuế/B thu/B nhập/I cá/B nhân/I.” Here, “” denotes an empty element in tuples. √ and X represent the correct and incorrect initial segmentations, respectively.

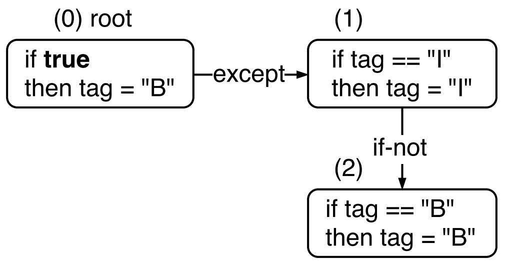


Figure 3: SCRDR tree initialization.

We apply an *initial segmenter* on the input raw corpus to get the output BI-formed *initialized corpus*. For example, given the input raw text “thuế thu nhập cá nhân”, the initial segmenter returns the output BI-formed initialized text “thuế/B thu/B nhập/I cá/B nhân/I.” The initial segmenter in our approach is based on the longest matching strategy (Poowarawan, 1986), using a Vietnamese lexicon from Le et al. (2008).

We then compare the BI-formed gold standard corpus and the BI-formed initialized corpus to generate a *5-syllable context dictionary* $\mathcal{D}$ where each key-value pair consists of a 5-syllable window tuple as key and a gold standard tag as value. Here, each tuple captures a 5-syllable window context of a current syllable and its initial segmentation tag B/I in a format of (Previous-2nd-syllable, Previous-2nd-tag, Previous-1st-syllable, Previous-1st-tag, syllable, tag, Next-1st-syllable, Next-1st-tag, Next-2nd-syllable, Next-2nd-tag) from the initialized corpus,[2] while the gold standard tag is the corresponding segmentation tag of the current syllable in the gold standard corpus. So, a wrong segmentation is when the initial segmentation tag is different from the gold standard tag, as shown in the second row in Table 1.

Based on the 5-syllable context dictionary $\mathcal{D}$, the *rule*

syllable	s_{-2}, s_{-1}, s_0, s_{+1}, s_{+2}
	(s_{-2}, s_0), (s_{-1}, s_0), (s_{-1}, s_{+1}), (s_0, s_{+1}) (s_0, s_{+2})
	(s_{-2}, s_{-1}, s_0), (s_{-1}, s_0, s_{+1}), (s_0, s_{+1}, s_{+2})
tag	t_{-2}, t_{-1}, t_0, t_{+1}, t_{+2}
	(t_{-2}, t_{-1}), (t_{-1}, t_{+1}), (t_{+1}, t_{+2})
syllable & tag	(t_{-1}, s_0), (s_0, t_{+1}), (t_{-1}, s_0, t_{+1}), (t_{-2}, t_{-1}, s_0) (s_0, t_{+1}, t_{+2})

Table 2: Short descriptions of our rule templates. “s” refers to syllable and “t” refers to B/I segmentation label while subscripts -2, -1, 0, 1, 2 denote indices. For example, (s_{-1}, s_{+1}) represents the rule template “IF Previous-1st-syllable == **tuple.Previous-1st-syllable** && Next-1st-syllable == **tuple.Next-1st-syllable** THEN tag = **gold-standard-tag**”, where elements in **bold** are replaced by concrete values from tuple and gold tag pairs in the 5-syllable context dictionary $\mathcal{D}$. Given (s_{-1}, s_{+1}) and the second row in Table 1, we have a concrete rule “IF Previous-1st-syllable == **thuế** && Next-1st-syllable == **nhập** THEN tag = **I**.”

selector selects the most suitable rules to construct the *SCRDR tree*. Concrete rules are generated based on *rule templates*. Table 2 presents short descriptions of the rule templates. The SCRDR tree is initialized with a default rule—the rule at the root node—and its two exception rules, as shown in Figure 3. Our learning process to automatically add new exception rules to the SCRDR tree is as follows:

- Let us consider a node N in the tree. We define a subset $\mathcal{T}_{\mathsf{N}}$ of the context dictionary $\mathcal{D}$ such that the rule at N is the last satisfied rule in the evaluation path for every tuple in $\mathcal{T}_{\mathsf{N}}$ but N returns a wrong segmentation tag. For example, given node (2) in Figure 3 and $\mathcal{D}$ in Table 1, $\mathcal{T}_{(2)}$ would contain a pair of the tuple (“”, “”, thuế, B, **thu**, **B**, nhập, I cá, B) and gold segmentation tag **I** from the second row in Table 1. A new node containing a new exception rule must be added to the current tree to correct the errors given by N.[3]

- The new exception rule is selected from all concrete rules, in which these concrete rules are generated by applying the rule templates to all tuples in $\mathcal{T}_{\mathsf{N}}$. The selected rule must satisfy following constraints: (1) If N is not one of the first three nodes in Figure 3, then the selected rule's condition must not be satisfied by every tuple for which N already returns a correct segmentation tag. (2) The selected rule is associated with the highest

[2]Syllables in each tuple are all converted into a lowercase form.

[3]See the second last paragraph in Section 2. for how to attach a new node to an existing SCRDR tree.

value of the subtraction $a - b$. Here a is the number of tuples in $\mathcal{T}_N$ in which each tuple not only satisfies the rule's condition but also gets a correct segmentation tag given by the rule's conclusion, while b is the number of tuples in $\mathcal{T}_N$ in which each tuple also satisfies the rule's condition but gets a wrong segmentation tag given by the rule's conclusion. (3) The value $a - b$ must be not smaller than a given threshold.

- This process is repeated until at any node it cannot select a new exception rule satisfying constraints above.

With the learned SCRDR tree, we perform word segmentation on unsegmented text as follows: The initial segmenter takes the input unsegmented text to generate a BI-formed initialized text. Next, by sliding a 5-syllable window from left to right, a tuple is generated for each syllable in the initialized text; then the learned SCRDR tree takes the input tuple to return a final segmentation tag to the corresponding syllable. Finally, the output of this labeling process is converted to the traditional underscore-based representation.

4. Experiments

4.1. Experimental setup

Following Nguyen and Le (2016), we conduct experiments and compare the performance of our approach—which we call RDRsegmenter—with published results of other state-of-the-art approaches on the benchmark Vietnamese treebank (Nguyen et al., 2009). The training set consists of 75k manually word-segmented sentences (about 23 words per sentence in average).[4] The test set consists of 2120 sentences (about 31 words per sentence) in 10 files from *800001.seg* to *800010.seg*.[5] We use F_1 score as the main evaluation metric to measure the performance of word segmentation.

Note that to determine the threshold in our RDRsegmenter, we sampled a development set of 5k sentences from the full training set and used the remaining 70k sentences for training. We found an optimal threshold value at 2 producing the highest F_1 score on the development set. Then we learned a SCRDR tree from the full training set with the optimal threshold, resulting in 1447 rules in total.

[4]The data, officially released in 2013, is provided for research or educational purpose by the national project VLSP on Vietnamese language and speech processing.

[5]The test set was originally released for evaluation in the POS tagging shared task at the VLSP 2013 workshop.

Approach	Precision	Recall	F_1
vnTokenizer	96.98	97.69	97.33
JVnSegmenter-Maxent	96.60	97.40	97.00
JVnSegmenter-CRFs	96.63	97.49	97.06
DongDu	96.35	97.46	96.90
UETsegmenter	**97.51**	98.23	97.87
Our **RDRsegmenter**	97.46	**98.35**	**97.90**

Table 3: Vietnamese word segmentation results (in %). The results of vnTokenizer, JVnSegmenter and DongDu are reported in Nguyen and Le (2016).

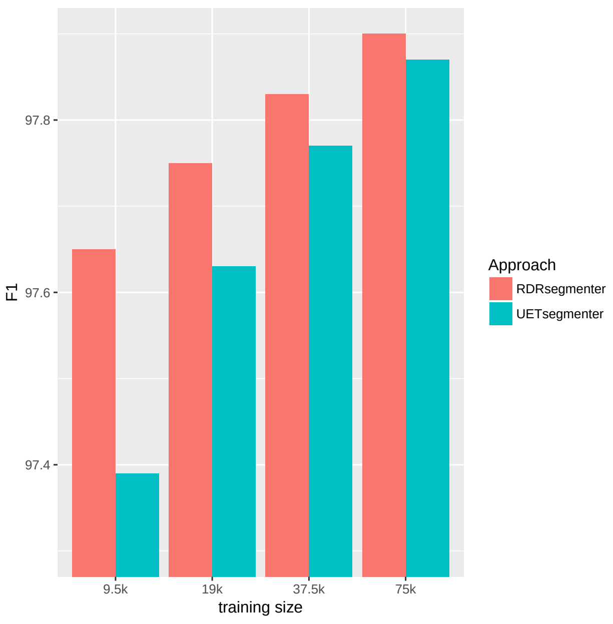


Figure 4: F_1 scores (in %) when varying the training size at 9.5k, 19k, 37.5k and full 75k sentences.

4.2. Main results

Table 3 compares the Vietnamese word segmentation results of our RDRsegmenter with results reported in prior work, using the same experimental setup.

Table 3 shows that RDRsegmenter obtains the highest F_1 score. In particular, RDRSegmenter obtains 0.5+% higher F_1 than vnTokenizer (Le et al., 2008) though both approaches use the same lexicon for initial segmentation. In terms of a sequence labeling task, RDRSegmenter outperforms JVnSegmenter (Nguyen et al., 2006) with 0.8+% improvement. Compared with the pointwise prediction approaches DongDu (Luu and Kazuhide, 2012) and UETsegmenter (Nguyen and Le, 2016), RDRsegmenter does significantly better than DongDu and somewhat better than UETsegmenter. In Figure 4, we show F_1 scores of RDRsegmenter and UETsegmenter at different training sizes, showing that RDRsegmenter clearly improves performance in a smaller dataset scenario.

It is worth noting that on a personal computer of Intel Core i7 2.2 GHz, our RDRsegmenter processes at a speed of 62k words per second in a single threaded implementation, which is 1.3 times faster than UETsegmenter.[6] In addition, Nguyen and Le (2016) showed that UETsegmenter is faster than vnTokenizer, JVnSegmenter and DongDu.[7] So RDRsegmenter is also faster than vnTokenizer, JVnSegmenter and DongDu.

5. Conclusion

In this paper, we have proposed a new error-driven method to automatically construct a Single Classification Ripple Down Rules tree for Vietnamese word segmentation. Experiments on the benchmark Vietnamese treebank show that our method obtains better accuracy and speed than previous approaches. Our code is available at: `https://github.com/datquocnguyen/RDRsegmenter`.
Note that excluding the language-specific initial segmenter, our method generally can be viewed as a language independent approach. Here, a Vietnamese syllable is analogous to a character in other languages such as Chinese and Japanese. So we will adapt our method to those languages in future work.

6. Acknowledgments

This research was partially supported by the Australian Government through the Australian Research Council's Discovery Projects funding scheme (project DP160102156). This research was also partially supported by NICTA, funded by the Australian Government through the Department of Communications and the Australian Research Council through the ICT Centre of Excellence Program. This research was done while the first author was at Macquarie University.

7. Bibliographical References

Compton, P. and Jansen, B. (1988). Knowledge in Context: A Strategy for Expert System Maintenance. In *Proceedings of the 2nd Australian Joint Artificial Intelligence Conference*, pages 292–306.

Compton, P. and Jansen, R. (1990). A Philosophical Basis for Knowledge Acquisition. *Knowledge Aquisition*, 2(3):241–257.

Dien, D., Kiem, H., and Toan, N. V. (2001). Vietnamese Word Segmentation. In *Proceedings of the Sixth Natural Language Processing Pacific Rim Symposium*, pages 749–756.

Dinh, D. and Vu, T. (2006). A Maximum Entropy Approach for Vietnamese Word Segmentation. In *Proceedings of the 2006 International Conference on Research, Innovation and Vision for the Future*, pages 248–253.

Duc Cong, S. N., Hung Ngo, Q., and Jiamthapthaksin, R. (2016). State-of-the-art Vietnamese word segmentation. In *Proceedings of the 2nd International Conference on Science in Information Technology*, pages 119–124.

Ha, L. A. (2003). A method for word segmentation in Vietnamese. In *Proceedings of Corpus Linguistics*.

Le, H. P., Nguyen, T. M. H., Roussanaly, A., and Ho, T. V. (2008). A hybrid approach to word segmentation of Vietnamese texts. In *Proceedings of the 2nd International Conference on Language and Automata Theory and Applications*, pages 240–249.

Liu, W. and Lin, L. (2014). Probabilistic Ensemble Learning for Vietnamese Word Segmentation. In *Proceedings of the 37th International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 931–934.

Luu, T. A. and Kazuhide, Y. (2012). Ứng dụng phương pháp Pointwise vào bài toán tách từ cho tiếng Việt. https://github.com/rockkhuya/DongDu.

NanYuan, L. and YanBin, Z. (1991). A Chinese word segmentation model and a Chinese word segmentation system PC-CWSS. *Journal of Chinese Language and Computing*, 1(1).

Neubig, G. and Mori, S. (2010). Word-based Partial Annotation for Efficient Corpus Construction. In *Proceedings of the Seventh International Conference on Language Resources and Evaluation*, pages 2723–2727.

Nguyen, T.-P. and Le, A.-C. (2016). A Hybrid Approach to Vietnamese Word Segmentation. In *Proceedings of the 2016 IEEE RIVF International Conference on Computing and Communication Technologies: Research, Innovation, and Vision for the Future*, pages 114–119.

Nguyen, D. B. and Pham, S. B. (2012). Ripple Down Rules for Vietnamese Named Entity Recognition. In *Proceedings of the 4th international conference on Computational Collective Intelligence: Technologies and Applications*, pages 354–363.

[6]We repeated the segmentation process on the test set 100 times, and then computed the averaged speed. Note that model loading time was not taken into account, in which RDRsegmenter took 50 miniseconds while UETsegmenter took 10 seconds.

[7]Evaluated on a computer of Intel Core i5-3337U 1.80GHz, Nguyen and Le (2016) showed that JVnSegmenter, vnTokenizer, DongDu and UETsegmenter obtained performance speeds at 1k, 5k, 17k and 33k words per second, respectively. All of them are implemented in Java except DongDu which is in C++. Our RDRsegmenter is also implemented in Java.

Nguyen, C.-T., Nguyen, T.-K., Phan, X.-H., Nguyen, L.-M., and Ha, Q.-T. (2006). Vietnamese Word Segmentation with CRFs and SVMs: An Investigation. In *Proceedings of the 20th Pacific Asia Conference on Language, Information and Computation*, pages 215–222.

Nguyen, P. T., Vu, X. L., Nguyen, T. M. H., Nguyen, V. H., and Le, H. P. (2009). Building a Large Syntactically-Annotated Corpus of Vietnamese. In *Proceedings of the Third Linguistic Annotation Workshop*, pages 182–185.

Nguyen, D. Q., Nguyen, D. Q., and Pham, S. B. (2011a). Systematic Knowledge Acquisition for Question Analysis. In *Proceedings of the 8th International Conference on Recent Advances in Natural Language Processing*, pages 406–412.

Nguyen, D. Q., Nguyen, D. Q., Pham, S. B., and Pham, D. D. (2011b). Ripple Down Rules for Part-of-Speech Tagging. In *Proceedings of the 12th International Conference on Intelligent Text Processing and Computational Linguistics - Volume Part I*, pages 190–201.

Nguyen, Q. T., Nguyen, N. L., and Miyao, Y. (2012). Comparing Different Criteria for Vietnamese Word Segmentation. In *Proceedings of the 3rd Workshop on South and Southeast Asian Natural Language Processing*, pages 53–68.

Nguyen, D. Q., Nguyen, D. Q., and Pham, S. B. (2013). KbQAS: A Knowledge-based QA System. In *Proceedings of the 12th International Semantic Web Conference (Posters & Demonstrations Track)*, pages 109–112.

Nguyen, D. Q., Nguyen, D. Q., Pham, D. D., and Pham, S. B. (2014). RDRPOSTagger: A Ripple Down Rules-based Part-Of-Speech Tagger. In *Proceedings of the Demonstrations at the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 17–20.

Nguyen, D. Q., Nguyen, D. Q., Pham, D. D., and Pham, S. B. (2016). A Robust Transformation-Based Learning Approach Using Ripple Down Rules for Part-of-Speech Tagging. *AI Communications*, 29(3):409–422.

Nguyen, D. Q., Nguyen, D. Q., and Pham, S. B. (2017a). Ripple Down Rules for Question Answering. *Semantic Web*, 8(4):511–532.

Nguyen, D. Q., Vu, T., Nguyen, D. Q., Dras, M., and Johnson, M. (2017b). From Word Segmentation to POS Tagging for Vietnamese. In *Proceedings of the Australasian Language Technology Association Workshop 2017*, pages 108–113.

Pham, S. B. and Hoffmann, A. (2006). Efficient Knowledge Acquisition for Extracting Temporal Relations. In *Proceedings of the 17th European Conference on Artificial Intelligence*, pages 521–525.

Pham, D. D., Tran, G. B., and Pham, S. B. (2009). A Hybrid Approach to Vietnamese Word Segmentation using Part of Speech tags. In *Proceedings of the 2009 International Conference on Knowledge and Systems Engineering*, pages 154–161.

Plisson, J., Lavrač, N., Mladenić, D., and Erjavec, T. (2008). Ripple Down Rule Learning for Automated Word Lemmatisation. *AI Communications*, 21(1):15–26.

Poowarawan, Y. (1986). Dictionary-based Thai Syllable Separation. In *Proceedings of the Ninth Electronics Engineering Conference*, pages 409–418.

Richards, D. (2009). Two Decades of Ripple Down Rules Research. *Knowledge Engineering Review*, 24(2):159–184.

Takahashi, K. and Yamamoto, K. (2016). Fundamental tools and resource are available for Vietnamese analysis. In *Proceedings of the 2016 International Conference on Asian Language Processing*, pages 246–249.

Thang, D. Q., Phuong, L. H., Huyen, N. T. M., Tu, N. C., Rossignol, M., and Luong, V. X. (2008). Word segmentation of Vietnamese texts : a comparison of approaches. In *Proceedings of the 6th International Conference on Language Resources and Evaluation*, pages 1933–1936.

Tran, T. O., Le, A. C., and Ha, Q. T. (2010). Improving Vietnamese Word Segmentation and POS Tagging using MEM with Various Kinds of Resources. *Journal of Natural Language Processing*, 17(3):41–60.

Tran, N. A., Dao, T. T., and Nguyen, P. T. (2012). An effective context-based method for Vietnamese-word segmentation. In *Proceedings of the 1st International Workshop on Vietnamese Language and Speech Processing*, pages 34–40.

Xu, H. and Hoffmann, A. (2010). RDRCE: Combining Machine Learning and Knowledge Acquisition. In *Proceedings of the 11th International Workshop on Knowledge Management and Acquisition for Smart Systems and Services*, pages 165–179.