A Hybrid Approach to Word Segmentation of Vietnamese Texts

Hong Phuong Le, Thi Minh Huyen Nguyen, Azim Roussanaly, Tuong Vinh Ho

A Hybrid Approach to Word Segmentation of Vietnamese Texts

Lê Hồng Phương[1], Nguyễn Thị Minh Huyền[2], Azim Roussanaly[1], and Hồ Tường Vinh[3]

[1] LORIA, Nancy, France
[2] Vietnam National University, Hanoi, Vietnam
[3] IFI, Hanoi, Vietnam

Abstract We present in this article a hybrid approach to automatically tokenize Vietnamese text. The approach combines both finite-state automata technique, regular expression parsing and the maximal-matching strategy which is augmented by statistical methods to resolve ambiguities of segmentation. The Vietnamese lexicon in use is compactly represented by a minimal finite-state automaton. A text to be tokenized is first parsed into lexical phrases and other patterns using pre-defined regular expressions. The automaton is then deployed to build linear graphs corresponding to the phrases to be segmented. The application of a maximal-matching strategy on a graph results in all candidate segmentations of a phrase. It is the responsibility of an ambiguity resolver, which uses a smoothed bigram language model, to choose the most probable segmentation of the phrase. The hybrid approach is implemented to create *vnTokenizer*, a highly accurate tokenizer for Vietnamese texts.

1 Introduction

As many occidental languages, Vietnamese is an alphabetic script. Alphabetic scripts usually separate words by blanks and a tokenizer which simply replaces blanks with word boundaries and cuts off punctuation marks, parentheses and quotation marks at both ends of a word, is already quite accurate [5]. However, unlike other languages, in Vietnamese blanks are not only used to separate words, but they are also used to separate syllables that make up words. Furthermore, many of Vietnamese syllables are words by themselves, but can also be part of multi-syllable words whose syllables are separated by blanks between them. In general, the Vietnamese language creates words of complex meaning by combining syllables that most of the time also possess a meaning when considered individually. This linguistic mechanism makes Vietnamese close to that of syllabic scripts,

like Chinese. That creates problems for all natural language processing tasks, complicating the identification of what constitutes a word in an input text.

Many methods for word segmentation have been proposed. These methods can be roughly classified as either dictionary-based or statistical methods, while many state-of-the-art systems use hybrid approaches [6].

We present in this paper an efficient hybrid approach for the segmentation of Vietnamese text. The approach combines both finite-state automata technique, regular expression parsing and the maximal-matching method which is augmented by statistical methods to deal with ambiguities of segmentation. The rest of the paper is organized as follows. The next section gives the construction of a minimal finite-state automaton that encodes the Vietnamese lexicon. Sect. 3 discusses the application of this automaton and the hybrid approach for word segmentation of Vietnamese texts. The developed tokenizer for Vietnamese and its experimental results are shown in Sect. 4. Finally, we conclude the paper with some discussions in Sect. 5.

2 Lexicon Representation

In this section, we first briefly describe the Vietnamese lexicon and then introduce the construction of a minimal deterministic, acyclic finite-state automaton that accepts it.

2.1 Vietnamese Lexicon

The Vietnamese lexicon edited by the Vietnam Lexicography Center (Vietlex[4]) contains $40,181$ words, which are widely used in contemporary spoken language, newspapers and literature. These words are made up of $7,729$ syllables. It is noted that Vietnamese is an inflexionless language, this means that every word has exactly one form.

There are some interesting statistics about lengths of words measured in syllables as shown in Table 1. Firstly, there are about 81.55% of syllables which are words by themselves, they are called single words; 15.69% of words are single ones. Secondly, there are 70.72%

[4] *http://www.vietlex.com/*

of compound words which are composed of two syllables. Finally, there are 13, 59% of compounds which are composed of at least three syllables; only 1, 04% of compounds having more than four syllables.

Table 1. Lengths of words measured in syllables

Length	#	%
1	6,303	15.69
2	28,416	70.72
3	2,259	5.62
4	2,784	6.93
≥ 5	419	1.04
Total	40,181	100

The high frequency of two-syllable compounds suggests us a simple but efficient method to resolve ambiguities of segmentation. The next paragraph presents the representation of the lexicon.

2.2 Lexicon Representation

Minimal deterministic finite state automata (MDFA) have been known to be the best representation of a lexicon. They are not only compact but also give the optimal access time to data [1]. The Vietnamese lexicon is represented by an MDFA.

We implement an algorithm developed by J. Daciuk *et al.* [2] that incrementally builds a minimal automaton in a single phase by adding new strings one by one and minimizing the resulting automaton on-the-fly.

The minimal automaton that accepts the Vietnamese lexicon contains 42,672 states in which 5,112 states are final ones. It has 76,249 transitions; the maximum number of outgoing transitions from a state is 85, and the maximum number of incoming transitions to a state is 4,615. The automaton operates in optimal time in the sense that the time to recognize a word corresponds to the time required to follow a single path in the deterministic finite-state machine, and the length of the path is the length of the word measured in characters.

3 Vietnamese Word Segmentation

We present in this section an application of the lexicon automaton for the word segmentation of Vietnamese texts. We first give the specification of segmentation task.

3.1 Segmentation Specification

We have developed a set of segmentation rules based on the principles discussed in the document of the ISO/TC 37/SC 4 work group on word segmentation (2006) [3]. Notably, the segmentation of a corpus follows the following rules:

1. *Compounds:* word compounds are considered as words if their meaning is not compound from their sub parts, or if their usage frequency justifies it.
2. *Derivation:* when a bound morpheme is attached to a word, the result is considered as a word. The reduplication of a word (common phenomenon in Vietnamese) also gives a lexical unit.
3. *Multiword expressions:* expressions such as "because of" are considered as lexical units.
4. *Proper names:* name of people and locations are considered as lexical units.
5. *Regular patterns:* numbers, times and dates are recognized as lexical units.

3.2 Word Segmentation

An input text for segmentation is first analyzed by a regular expression recognizer for detection of regular patterns such as proper names, common abbreviations, numbers, dates, times, email addresses, URLs, punctuations, *etc.* The recognition of arbitrary compounds, derivation, and multiword expressions is committed to a regular expression that extracts phrases of the text.

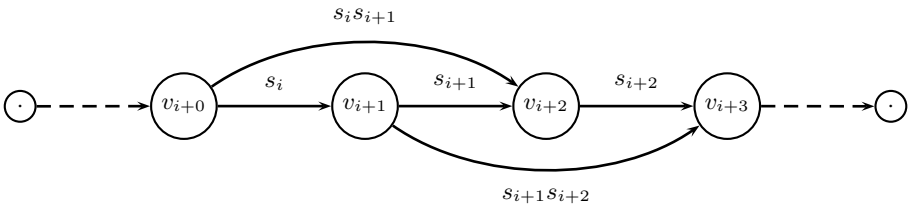
The regular recognizer analyzes the text using a greedy strategy in that all patterns are scanned and the longest matched pattern is taken out. If a pattern is a phrase, that is a sequence of syllables and spaces, it is passed to a segmenter for detection of word composition. In general, a phrase usually has several different word compositions;

nevertheless, there is typically one correct composition which the segmenter need to determine.

A simple segmenter could be implemented by the maximal matching strategy which selects the segmentation that contains the fewest words [8]. In this method, the segmenter determines the longest syllable sequence which starts at the current position and is listed in the lexicon. It takes the recognized pattern, moves the position pointer behind the pattern, and starts to scan the next one. Although this method works quite well since long words are more likely to be correct than short words. However, this is a too greedy method which sometimes leads to wrong segmentation because of a large number of overlapping candidate words in Vietnamese. Therefore, we need to list all possible segmentations and design a strategy to select the most probable correct segmentation from them.

A phrase can be formalized as a sequence of blank-separated syllables $s_1 s_2 \cdots s_n$. We ignore for the moment the possibility of seeing a new syllable or a new word in this sequence. Due to the fact that, as we showed in the previous section, most of Vietnamese compound words are composed of two syllables, the most frequent case of ambiguities involves three consecutive syllables $s_i s_{i+1} s_{i+2}$ in which both of the two segmentations $(s_i s_{i+1})(s_{i+2})$ and $(s_i)(s_{i+1} s_{i+2})$ may be correct, depending on context. This type of ambiguity is called *overlap ambiguity*, and the string $s_i s_{i+1} s_{i+2}$ is called an overlap ambiguity string.

Figure 1. Graph representation of a phrase



The phrase is represented by a linearly directed graph $G = (V, E)$, $V = \{v_0, v_1, \ldots, v_n, v_{n+1}\}$, as shown in Fig. 1. Vertices v_0 and v_{n+1} are respectively the start and the end vertex; n vertices $v_1, v_2, \ldots, v_n$ are aligned to n syllables of the phrase. There is an arc (v_i, v_j) if the consecutive syllables $s_{i+1}, s_{i+2}, \ldots, s_j$ compose a word,

for all $i < j$. If we denote $\texttt{accept}(\mathcal{A}, s)$ the fact that the lexicon automaton $\mathcal{A}$ accepts the string s, the formal construction of the graph for a phrase is shown in Algorithm 1. We can then propose all segmentations of the phrase by listing all shortest paths on the graph from the start vertex to the end vertex.

Algorithm 1 Construction of the graph for a phrase $s_1 s_2 \dots s_n$

```
1: V ← ∅;
2: for i = 0 to n + 1 do
3:     V ← V ∪ {v_i};
4: end for
5: for i = 0 to n do
6:     for j = i to n do
7:         if (accept(A_W, s_i ··· s_j)) then
8:             E ← E ∪ {(v_i, v_{j+1})};
9:         end if
10:    end for
11: end for
12: return G = (V, E);
```

As illustrated in Fig. 1, each overlap ambiguity string results in an ambiguity group, therefore, if a graph has k ambiguity groups, there are 2^k segmentations of the underlying phrase[5]. For example, the ambiguity group in Fig. 1 gives two segmentations $(s_i s_{i+1}) s_{i+2}$ and $s_i (s_{i+1} s_{i+2})$.

We discuss in the next subsection the ambiguity resolver which we develop to choose the most probable segmentation of a phrase in the case it has overlap ambiguities.

3.3 Resolution of Ambiguities

The ambiguity resolver uses a bigram language model which is augmented by the linear interpolation smoothing technique.

In n-gram language modeling, the probability of a string $P(s)$ is expressed as the product of the probabilities of the words that compose the string, with each word probability conditional on the

[5] If these ambiguity groups do not overlap each other.

identity of the last $n-1$ words, *i.e.*, if $s = w_1 \cdots w_m$ we have

$$P(s) = \prod_{i=1}^{m} P(w_i|w_1^{i-1}) \approx \prod_{i=1}^{m} P(w_i|w_{i-n+1}^{i-1}), \tag{1}$$

where w_i^j denotes the words $w_i \cdots w_j$. Typically, n is taken to be two or three, corresponding to a bigram or trigram model, respectively.[6]

In the case of a bigram model $n = 2$, to estimate the probabilities $P(w_i|w_{i-1})$ in (1), we can use training data, and take the maximum likelihood (ML) estimate for $P(w_i|w_{i-1})$ as follows

$$P_{ML}(w_i|w_{i-1}) = \frac{P(w_{i-1}w_i)}{P(w_{i-1})} = \frac{c(w_{i-1}w_i)/N}{c(w_{i-1})/N} = \frac{c(w_{i-1}w_i)}{c(w_{i-1})},$$

where $c(\alpha)$ denotes the number of times the string α occurs and N is the total number of words in the training data.

The maximum likelihood estimate is a poor one when the amount of training data is small compared to the size of the model being built, as is generally the case in language modeling. A zero bigram probability can lead to errors of the modeling. Therefore, a variety of smoothing techniques have been developed to adjust the maximum likelihood estimate in order to produce more accurate probabilities. Not only do smoothing methods generally prevent zero probabilities, but they also attempt to improve the accuracy of the model as a whole. Whenever a probability is estimated from few counts, smoothing has the potential to significantly improve estimation [7].

We adopt the linear interpolation technique to smooth the model. This is a simple yet effective smoothing technique which is widely used in the domain of language modeling [4]. In this method, the bigram model is interpolated with a unigram model $P_{ML}(w_i) = c(w_i)/N$, a model that reflects how often each word occurs in the training data. We take our estimate $\widehat{P}(w_i|w_{i-1})$ to be

$$\widehat{P}(w_i|w_{i-1}) = \lambda_1 P_{ML}(w_i|w_{i-1}) + \lambda_2 P_{ML}(w_i), \tag{2}$$

where $\lambda_1 + \lambda_2 = 1$ and $\lambda_1, \lambda_2 \geq 0$.

[6] To make the term $P(w_i|w_{i-n-1}^{i-1})$ meaningful for $i < n$, one can pad the beginning of the string with a distinguished token. We assume there are $n-1$ such distinguished tokens preceding each phrase.

The objective of smoothing techniques is to improve the performance of a language model, therefore the estimation of λ values in (2) is related to the evaluation of the language model. The most common metric for evaluating a language model is the probability that the model assigns to test data, or more conveniently, the derivative measured of entropy. For a smoothed bigram model that has probabilities $p(w_i|w_{i-1})$, we can calculate the probability of a sentence $P(s)$ using (1). For a test set T composed of n sentences $s_1, s_2, \ldots, s_n$, we can calculate the probability $P(T)$ of the test set as the product of the probabilities of all sentences in the set $P(T) = \prod_{i=1}^{n} P(s_i)$. The entropy $H_p(T)$ of the model on data T is defined by

$$H_p(T) = \frac{-\log_2 P(T)}{N_T} = -\frac{1}{N_T} \sum_{i=1}^{n} \log_2 P(s_i), \tag{3}$$

where N_T is the length of the text T measured in words. The entropy is inversely related to the average probability a model assigns to sentences in the test data, and it is generally assumed that lower entropy correlates with better performance in applications.

Starting from a part of the training set which is called the "validation" data, we define $C(w_{i-1}, w_i)$ to be the number of times the bigram (w_{i-1}, w_i) is seen in the validation set. We need to choose λ_1, λ_2 to maximize

$$L(\lambda_1, \lambda_2) = \sum_{w_{i-1}, w_i} C(w_{i-1}, w_i) \log_2 \widehat{P}(w_i|w_{i-1}) \tag{4}$$

such that $\lambda_1 + \lambda_2 = 1$, and $\lambda_1, \lambda_2 \geq 0$.

The λ_1 and λ_2 values can be estimated by an iterative process given in Algorithm 2. Once all the parameters of the bigram model have been estimated, the smoothed probabilities of bigrams can be easily computed by (2). These results are used by the resolver to choose the most probable segmentation of a phrase, say, s, by comparing probabilities $P(s)$ which is estimated using (1). The segmentation with the greatest probability will be chosen.

We present in the next section the experimental setup and obtained results.

Algorithm 2 Estimation of values λ

$\lambda_1 \leftarrow 0.5, \lambda_2 \leftarrow 0.5$;
$\epsilon \leftarrow 0.01$;
repeat
  $\widehat{\lambda}_1 \leftarrow \lambda_1, \quad \widehat{\lambda}_2 \leftarrow \lambda_2$;
  $c_1 \leftarrow \sum_{w_{i-1},w_i} \frac{C(w_{i-1},w_i)\lambda_1 P_{ML}(w_i|w_{i-1})}{\lambda_1 P_{ML}(w_i|w_{i-1})+\lambda_2 P_{ML}(w_i)}$;
  $c_2 \leftarrow \sum_{w_{i-1},w_i} \frac{C(w_{i-1},w_i)\lambda_2 P_{ML}(w_i)}{\lambda_1 P_{ML}(w_i|w_{i-1})+\lambda_2 P_{ML}(w_i)}$;
  $\lambda_1 \leftarrow \frac{c_1}{c_1+c_2}, \quad \lambda_2 \leftarrow 1-\widehat{\lambda}_1$;
  $\widehat{\epsilon} \leftarrow \sqrt{(\widehat{\lambda}_1-\lambda_1)^2+(\widehat{\lambda}_2-\lambda_2)^2}$;
until $(\widehat{\epsilon} \leq \epsilon)$;
return λ_1, λ_2;

4 Experiments

We present in this section the experimental setup and give a report on results of experiments with the hybrid approach presented in the previous sections. We also describe briefly vnTokenizer, an automatic software for segmentation of Vietnamese texts.

4.1 Corpus Constitution

The corpus upon which we evaluate the performance of the tokenizer is a collection of 1264 articles from the "Politics – Society" section of the Vietnamese newspaper *Tuổi trẻ* (The Youth), for a total of 507, 358 words that have been manually spell-checked and segmented by linguists from the Vietnam Lexicography Center. Although there can be multiple plausible segmentations of a given Vietnamese sentence, only a single correct segmentation of each sentence is kept. We assume a single correct segmentation of a sentence for two reasons. The first one is of its simplicity. The second one is due to the fact that we are not currently aware of any effective way of using multiple segmentations in typical applications concerning Vietnamese processing.

We perform a 10-fold cross validation on the test corpus. In each experiment, we take 90% of the gold test set ($\approx$ 456, 600 lexical units) as training set, and 10% as test set. We present in the next paragraph the training and results of the model.

4.2 Results

In an experiment, the bigram language model is trained on a training set. An estimation of parameters λs in the Algorithm 2 is given in Table 2. With a given error $\epsilon = 0.03$, the estimated parameters converge after four iterations.

Table 2. Estimation of lambda values

Step	λ_1	λ_2	ϵ
0	0.500	0.500	1.000
1	0.853	0.147	0.499
2	0.952	0.048	0.139
3	0.981	0.019	0.041
4	0.991	0.009	0.015

The above experimental results reveal a fact that the smoothing technique basing on the linear interpolation adjusts well bigram and unigram probabilities, it thus improves the estimation and the accuracy of the model as a whole. Table 3 presents the values of precisions, recalls and F-measures of the system on two versions with or without ambiguity resolution. Precision is computed as the count of common tokens over tokens of the automatically segmented files, recall as the count of common tokens over tokens of the manually segmented files, and F-measure is computed as usual from these two values.

Table 3. Precision, recall and F-measure of the system

Precision	Recall	F-measure
0.948	0.960	0.954
0.950	0.963	0.956

The system has good recall ratios, about 96%. However, the use of the resolver for resolution of ambiguities only slightly improves the overall accuracy. This can be explained by the fact that the bigram model exploits a small amount of training data compared to the size of the universal language model. It is hopeful that the resolver may improve further the accuracy if it is trained on larger corpora.

4.3 vnTokenizer

We have developed a software tool named *vnTokenzier* that implements the presented approach for automatic word segmentation of Vietnamese texts. The tool is written in Java and bundled as an Eclipse plug-in and it has already been integrated into *vnToolkit*, an Eclipse Rich Client[7] application which is intended to be a general framework integrating tools for processing of Vietnamese text. *vnTokenizer* plug-in, *vnToolkit* and related resources, include the lexicon and test corpus are freely available for download[8]. They are distributed under the GNU General Public License[9].

5 Conclusion

We have presented an efficient hybrid approach to word segmentation of Vietnamese texts that gives a relatively high accuracy. The approach has been implemented to produce *vnTokenizer*, an automatic tokenizer for Vietnamese texts.

By analyzing results of experiments, we found two types of ambiguity strings in word segmentation of Vietnamese texts: (1) *overlap ambiguity strings* and (2) *combination ambiguity strings*. A sequence of syllables $s_1 s_2 \ldots s_n$ is called a combination ambiguity string if it is a compound word by itself and there exists its sub sequences which are also words by themselves in some context. For instance, the word *bà ba* (a kind of pajamas) may be segmented into two words *bà* and *ba* (the third wife), and there exists contexts under which this segmentation is both syntactically and semantically correct. Being augmented with a bigram model, our tokenizer is able to resolve effectively overlap ambiguity strings, but combination ambiguity strings have not been discovered. There is a delicate reason, it is that combination ambiguities require a judgment of the syntactic and semantic sense of the segmentation – a task where an agreement cannot be reached easily among different human annotators. Furthermore, we observe that the relative frequency of combination ambiguity strings in Vietnamese is small. In a few ambiguity cases involving bigrams,

[7] *http://www.eclipse.org/rcp/*
[8] *http://www.loria.fr/∼lehong/projects.php*
[9] *http://www.gnu.org/copyleft/gpl.html*

we believe that a trigram model resolver would work better. These questions would be of interest for further research to improve the accuracy of the tokenizer.

Finally, we found that the majority of errors of segmentation are due to the presence in the texts of compounds absent from the lexicon. Unknown compounds are a much greater source of segmenting errors than segmentation ambiguities. Future efforts should therefore be geared in priority towards the automatic detection of new compounds, which can be performed by means either statistical in a large corpus or rule-based using linguistic knowledge about word composition.

Acknowledgements

The work reported in this article would not have been possible without the enthusiastic collaboration of all the linguists at the Vietnam Lexicography Center. We thank them for their help in data preparation.

References

1. Denis Maurel, Electronic Dictionaries and Acyclic Finite-State Automata: A State of The Art, *Grammars and Automata for String Processing*, 2003.
2. Jan Daciuk, Stoyan Mihov, Bruce W. Watson and Richard E. Watson, Incremental Construction of Minimal Acyclic Finite-State Automata, *Computational Linguistics*, Vol. 26, No. 1, 2000.
3. ISO/TC 37/SC 4 AWI N309, Language Resource Management - Word Segmentation of Written Texts for Mono-lingual and Multi-lingual Information Processing - Part I: General Principles and Methods. Technical Report, ISO, 2006.
4. Frederick Jelinke and Robert L. Mercer, Interpolated estimation of Markov source parameters from sparse data, *Proceedings of the Workshop on Pattern Recognition in Practice*, The Netherlands, 1980.
5. Helmut Schmid, Tokenizing. In: Anke Lüdeling and Merja Kytö, editors: Corpus Linguistics. An International Handbook. Mouton de Gruyter, Berlin, 2007.
6. Jianfeng Gao et al, Chinese Word Segmentation and Named Entity Recognition: A Pragmatic Approach, *Computational Linguistics*, 2006.
7. Stanley F. Chen and Joshua Goodman, An Empirical Study of Smoothing Techniques for Language Modeling, *Proceedings of the 34th Annual Meeting of the ACL*, 1996.
8. Wong P. and Chan C., Chinese Word Segmentation based on Maximum Matching and Word Binding Force, *Proceedings of the 16th Conference on Computational Linguistics*, Copenhagen, DK, 1996.