Abstract

In recent years, social networks have become very popular. It is easy for users to share their data using online social networks. Since data on social networks is idiomatic, irregular, brief, and includes acronyms and spelling errors, dealing with such data is more challenging than that of news or formal texts. With the huge volume of posts each day, effective extraction and processing of these data will bring great benefit to information extraction applications.

This thesis proposes a method to normalize Vietnamese informal text in social networks. This method has the ability to identify and normalize informal text based on the structure of Vietnamese words, Vietnamese syllable rules, and a trigram model. After normalization, the data will be processed by a named entity recognition (NER) model to identify and classify the named entities in these data. In our NER model, we use six different types of features to recognize named entities categorized in three predefined classes: Person (PER), Location (LOC), and Organization (ORG).

When viewing social network data, we found that the size of these data are very large and increase daily. This raises the challenge of how to decrease this size. To deal with this challenge, in this thesis, we propose three methods to compress text files, especially in Vietnamese text. The first method is a syllable-based method relying on the structure of Vietnamese morphosyllables, consonants, syllables and vowels. The second method is trigram-based Vietnamese text compression based on a trigram dictionary. The last method is based on an n-gram slide window, in which we use five dictionaries for unigrams, bigrams, trigrams, four-grams and five-grams. This method achieves a promising compression ratio of around 90% and can be used for any size of text file.

Keywords: text normalization, named entity recognition, text compression.
Contents

1 Introduction 7
  1.1 Motivation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 7
  1.2 Thesis objective and scope . . . . . . . . . . . . . . . . . . . . . . . 9

2 Vietnamese text compression 9
  2.1 A syllable-based method for Vietnamese (SBV) text compression . . . . 9
    2.1.1 Dictionary . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 9
    2.1.2 Morphosyllable rules . . . . . . . . . . . . . . . . . . . . . . . . 11
    2.1.3 SBV text compression . . . . . . . . . . . . . . . . . . . . . . . 12
    2.1.4 SBV text decompression . . . . . . . . . . . . . . . . . . . . . . 14
    2.1.5 Compression ratio . . . . . . . . . . . . . . . . . . . . . . . . . 15
    2.1.6 Experiments . . . . . . . . . . . . . . . . . . . . . . . . . . . . 15
  2.2 Trigram-based Vietnamese text compression . . . . . . . . . . . . . . . 18
    2.2.1 Dictionary . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 18
    2.2.2 TGV text compression . . . . . . . . . . . . . . . . . . . . . . . 19
    2.2.3 TGV text decompression . . . . . . . . . . . . . . . . . . . . . . 20
    2.2.4 Experiments . . . . . . . . . . . . . . . . . . . . . . . . . . . . 21
  2.3 N-gram based text compression . . . . . . . . . . . . . . . . . . . . . . 23
    2.3.1 Dictionaries . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 24
    2.3.2 N-gram based text compression . . . . . . . . . . . . . . . . . . 25
    2.3.3 N-gram based text decompression . . . . . . . . . . . . . . . . . 28
    2.3.4 Experiments . . . . . . . . . . . . . . . . . . . . . . . . . . . . 29

3 Normalization of Vietnamese informal text 33
  3.1 Preprocessing . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 34
    3.1.1 Clean up noisy symbols . . . . . . . . . . . . . . . . . . . . . . . 34
    3.1.2 Clean up repeated characters . . . . . . . . . . . . . . . . . . . . 34
  3.2 Spelling errors detection . . . . . . . . . . . . . . . . . . . . . . . . . . 34
    3.2.1 Typing error . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 34
    3.2.2 Spelling errors . . . . . . . . . . . . . . . . . . . . . . . . . . . 35
  3.3 Error correction . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 35
    3.3.1 Similarity of two morphosyllables . . . . . . . . . . . . . . . . . 35
    3.3.2 Similarity of two sentences . . . . . . . . . . . . . . . . . . . . . 36
  3.4 Experiments . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 37

4 Named entity recognition in Vietnamese informal text 39
  4.1 Normalization . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 39
  4.2 Capitalization classifier . . . . . . . . . . . . . . . . . . . . . . . . . . . 39
  4.3 Word segmentation and part of speech (POS) tagging . . . . . . . . . . . 40
    4.3.1 Word segmentation . . . . . . . . . . . . . . . . . . . . . . . . . . 40
    4.3.2 POS tagging . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 40
4.4 Extraction of features ............................................ 40
4.5 NER training set ...................................................... 41
4.6 Experiments .......................................................... 43

5 Conclusion ................................................................. 44
5.1 Contributions to text compression ................................. 44
5.2 Contribution to Vietnamese informal text normalization .......... 45
5.3 Contributions to NER in Vietnamese informal text .............. 45

6 List of Student’s Publication ........................................... 46
### List of Figures

1. a syllable-based Vietnamese text compression model . . . . . . . . . . . . 10
2. Compression ratios for file size smaller than 15 KB . . . . . . . . . . . . 16
3. Compression ratios for file size larger than 15 KB . . . . . . . . . . . . 17
4. Trigram-based Vietnamese Text Compression . . . . . . . . . . . . . . 18
5. Comparison between the dictionary one and the dictionary two . . . . . 22
6. Comparison between current method and syllable-based method . . . . . 23
7. Ngram-based Vietnamese text compression model . . . . . . . . . . . . . 24
8. Comparison between the three cases . . . . . . . . . . . . . . . . . . . 30
9. Comparison of compression ratio of three methods . . . . . . . . . . . . . 31
10. Compression ratio of our method, WinRAR and WinZIP . . . . . . . . . 32
11. Normalization model for Vietnamese tweets . . . . . . . . . . . . . . . 33
12. NER model in Vietnamese tweets . . . . . . . . . . . . . . . . . . . . . 39
### List of Tables

1. Dictionaries structure .......................................................... 10
2. Consonants dictionary ........................................................... 11
3. Vietnamese vowels and their marks ........................................ 11
4. Experimental results for file size smaller than 15 KB ............... 16
5. Experimental results for file size larger than 15 KB ................. 17
6. Dictionaries ............................................................................... 19
7. Compression ratio of the dictionary one and the dictionary two .... 21
8. Compression ratio of current method and syllable-based method .... 22
9. n-grams dictionaries ............................................................... 24
10. number of encoded bytes for each n-gram of each dictionary ....... 27
11. value of three MSB and number of bytes ................................. 27
12. Compression ratio of three experience cases ............................ 30
13. Compression ratio of the current method with the two previous methods .............................................................. 31
14. Compression ratio of our method, WinRAR and WinZIP .......... 32
15. tweets with spelling errors and their normalization. .................. 37
16. The results using fDice with two data sets in the trigram model ... 38
17. The results use fDice and Dice with trigram language model ...... 38
18. The result of the process of preparation data for SVM format ....... 42
19. The result of feature vectors .................................................... 42
20. Experimental results of case 1 and case 2 ............................... 43
21. Comparison performance of our method with that of [Tran 2007] .. 43
22. Contributions and publications ............................................... 45
## Acronyms

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NER</td>
<td>Named Entity Recognition</td>
</tr>
<tr>
<td>PER</td>
<td>Person</td>
</tr>
<tr>
<td>LOC</td>
<td>Location</td>
</tr>
<tr>
<td>ORG</td>
<td>Organization</td>
</tr>
<tr>
<td>NLP</td>
<td>Natural Language Processing</td>
</tr>
<tr>
<td>CUR</td>
<td>Currency</td>
</tr>
<tr>
<td>SBV</td>
<td>A syllable-based method for Vietnamese</td>
</tr>
<tr>
<td>CR</td>
<td>Compression ratio</td>
</tr>
<tr>
<td>TGV</td>
<td>Trigram-based Vietnamese</td>
</tr>
<tr>
<td>MBS</td>
<td>Most significant bit</td>
</tr>
<tr>
<td>SIM</td>
<td>Similarity of two sentences</td>
</tr>
<tr>
<td>R</td>
<td>Recall</td>
</tr>
<tr>
<td>F1</td>
<td>balance F-measure</td>
</tr>
<tr>
<td>POS</td>
<td>Part of speech</td>
</tr>
<tr>
<td>ORT</td>
<td>Orthographic</td>
</tr>
<tr>
<td>SVM</td>
<td>Support vector machine</td>
</tr>
</tbody>
</table>
1 Introduction

1.1 Motivation

In recent decades, along with the development of computer science and technology, the internet was also developed. Today, the internet has become one of the most popular channels for storing and transferring human information. The invention of the World Wide Web (the “Web”) and its rapid development created a convenient opportunity for the distributing and sharing of information over internet. It led to the explosion of information in terms of quantity, quality, and subject. Two decades ago, the capacity of information was usually measured in MB or GB. However, in recent years, along with the appearance of big data theory, the common measurement units are now GB, TB, and PB. Almost all information on the web has been presented in natural language under the format of the HTML language. This language lacks the capability to express the semantics of concepts and objects presented on the web. Therefore, the majority of current information on the web is only suitable for humans to read and understand. From the objectives of effective mining of information resources from web, several applications to extract documents automatically were developed, such as information extraction systems, information retrieval systems, machine translations, text summarization, and question answering systems, etc. for computers to understand the semantics of sections of a text, instead of trying to understand the entire semantics of text. Some approaches have been proposed so that we can understand main entities and concepts appearing in the text based on source knowledge of entities and concepts in the real world.

Named entity recognition (NER) is a subtask of information extraction and one of the important parts of Natural Language Processing (NLP). NER is the task of detecting named entities in documents and categorizing them to predefined classes. Common classes of NER systems are person (PER), location (LOC), organization (ORG), date (DATE), time (TIME), currency (CUR), etc. For example, let us consider the following sentence:

*On April 13, 2016, Mr. Hien Nguyen Thanh attended a meeting with CSC corporation in Ton Duc Thang University*

In this sentence, an NER system would recognize and classify four named entities as follows:

- *April 13, 2016* is a date
- *Hien Nguyen Thanh* is a person
- *CSC corporation* is an organization
- *Ton Duc Thang University* is an organization

After the named entity has been recognized, it can be used for different important tasks. For example, it can be used for named entity linking and machine translation, such as we
have on an iPhone device app, which takes a photo of a dish name on a menu, recognizes this name as an entity of dish names and foods, maps it to a knowledge source about entities and concepts in the real world, such as Wikipedia\(^1\). The app then translates it to the user’s language. Normally, from the recognized entities, other mining systems can be built to mine new classes of knowledge and get a better result than the raw text.

Many approaches have been proposed for NER from the first conference, the 6th Message Understanding Conference (MUC-6) in 1995. In this conference, the NER task was first introduced and was subsequently discussed at the next conference, the Conference on Computational Natural Language Learning (CoNLL) in 2002 and 2003. Most of them focused on English, Spanish, Dutch, German, and Chinese according to the data set from conferences and the popularity of these languages. In the domain of Vietnamese, several approaches have been proposed and were presented detail in \(^2\). However, none apply to Vietnamese informal text.

In this dissertation, we propose a method to fill that gap. We started research on NER for Vietnamese informal texts in the middle of 2014, and specifically focused on Vietnamese tweets on Twitter. When studying Vietnamese tweets, we found that they contained many spelling errors, typing errors, which created a significant challenge for NER. To overcome this challenge, we studied the Vietnamese language, normalization techniques, and proposed a method to normalize Vietnamese tweets in [Nguyen 2015b]. After we normalized these tweets, we proposed a method to recognize name entities in [Nguyen 2015a].

According to statistics from 2011, the number of tweets was up to 140 million per day\(^2\). With such a huge number of tweets being posted every day, it raised a storage challenge. Regarding this challenge, in [Nguyen 2015b], we used a trigram language model, which size is rather large compared with other methods. Therefore, we want to save its storage too. When researching this challenge, we found that there have not been any text compression methods proposed for Vietnamese. After studying several methods for text compression, we proposed the first approach for Vietnamese text compression based on syllable and structure of Vietnamese in [Nguyen 2016b]. In this approach, the compression ratio is converged to around 73\%. It is still low when compared with other methods and especially, this method has a high compression ratio for a small text file. From this disadvantage, we continued researching and proposed a method based on trigram in [Nguyen 2016c], and the compression ratio of this method shows significant improvement when compared with the previous method. Its compression ratio is around 82\%, and it is still not the best. In the next approach, we propose a method based on the n-gram slide window and achieve an encouraging compression ratio of around 90\%. This is higher than the two previous methods and with other methods. One important significance of this method, however, is that it can apply to any size of text file.

\(^1\)http://www.wikipedia.org
\(^2\)https://blog.twitter.com/2011/numbers
1.2 Thesis objective and scope

The objectives of this thesis are briefly summarized as follows.

1. To suggest a method to compress Vietnamese text based on Vietnamese morphosyllable structure, such as syllables, consonants, vowels, and marks; Vietnamese dictionary of syllables with their marks; Vietnamese dictionary of consonants, vowels, etc.

2. To propose a method to compress Vietnamese text based on the trigram language model.

3. To propose a method to compress text based on n-gram sliding windows.

4. To propose a method to detect Vietnamese errors in informal text, especially focused on Vietnamese tweets on Twitter, and to normalize them based on a dictionary of Vietnamese morphosyllables, Vietnamese morphosyllable structures, and Vietnamese syllable rules in the combination with language model.

5. To propose an NER model to recognize named entities in Vietnamese informal text, especially focused on Vietnamese tweets on Twitter.

The rest of this doctoral thesis summary is organized as follows. Section 2 presents the Vietnamese text compression. Section 3 presents the normalize of Vietnamese informal text. Section 4 presents the Name Entity Recognition in Vietnamese informal text. Conclusion is presented in Section 5.

2 Vietnamese text compression

2.1 A syllable-based method for Vietnamese (SBV) text compression

Figure 1 describes our method model. In our model, we use dictionaries and morphosyllable rules for both two phases. We will describe more details about it in following subsections.

2.1.1 Dictionary

In method, we use several dictionaries for both compression and decompression phases. These dictionaries have been built based on the combination between the syllables and marks. Because the total number of syllables and consonants is less than 256, for each dictionary we use the maximum 8 bits to represent. Table 1 describes the structure and number of entries of each dictionary.
There may appear to be cases where there are multiple capital letters for all characters of morphosyllable or capital letters of the first character of a syllable, non-standard word, e.g., email-id, or web link. We will handle this case by case and discuss more details in the following subsections.
2.1.2 Morphosyllable rules

a. Identifying consonant and syllable

In this section, we propose a rule to split the consonant and the syllable with mark. We build a table of consonants, including 27 consonants, as in Table 2. To split the consonant and syllable, we search the value of each consonant in the consonants dictionary. If it is found, then we can split the morphosyllable to the consonant and syllable with mark based on the length of the consonant in the consonants dictionary.

<table>
<thead>
<tr>
<th>index</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>ngh</td>
<td>ng</td>
<td>gi</td>
<td>gh</td>
<td>kh</td>
<td>nh</td>
<td>ph</td>
<td>th</td>
<td>tr</td>
<td>k</td>
<td>g</td>
<td>h</td>
<td>l</td>
<td>m</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>index</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
<th>19</th>
<th>20</th>
<th>21</th>
<th>22</th>
<th>23</th>
<th>24</th>
<th>24</th>
<th>25</th>
<th>26</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>n</td>
<td>q</td>
<td>r</td>
<td>s</td>
<td>t</td>
<td>v</td>
<td>x</td>
<td>p</td>
<td>b</td>
<td>ch</td>
<td>c</td>
<td>d</td>
<td>d</td>
<td></td>
</tr>
</tbody>
</table>

Table 2: Consonants dictionary

b. Identifying mark

Vietnamese has six types of marks. Because we split a morphosyllable into a consonant and syllable with mark, we must know what the mark of this syllable is to map it with the mark to find the correct syllable with mark in the dictionary. To do that, we built a table of 12 vowels and six marks. Refer to Table 3.

<table>
<thead>
<tr>
<th>mark</th>
<th>index</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>acute accent</td>
<td>0</td>
<td>ã</td>
<td>â</td>
<td>ấ</td>
<td>ắ</td>
<td>ế</td>
<td>ê</td>
<td>ế</td>
<td>ế</td>
<td>ố</td>
<td>ớ</td>
<td>ố</td>
<td>Ứ</td>
</tr>
<tr>
<td>dot below</td>
<td>1</td>
<td>a</td>
<td>â</td>
<td>ấ</td>
<td>ắ</td>
<td>ē</td>
<td>ē</td>
<td>ē</td>
<td>ế</td>
<td>ơ</td>
<td>ơ</td>
<td>ố</td>
<td>ơ</td>
</tr>
<tr>
<td>none</td>
<td>2</td>
<td>a</td>
<td>â</td>
<td>ấ</td>
<td>ắ</td>
<td>ē</td>
<td>ơ</td>
<td>ơ</td>
<td>ơ</td>
<td>ă</td>
<td>ă</td>
<td>ă</td>
<td>ă</td>
</tr>
<tr>
<td>tilda accent</td>
<td>5</td>
<td>ā</td>
<td>ā</td>
<td>ā</td>
<td>ā</td>
<td>ē</td>
<td>ē</td>
<td>ē</td>
<td>ē</td>
<td>ơ</td>
<td>ơ</td>
<td>ơ</td>
<td>ơ</td>
</tr>
<tr>
<td>grave accent</td>
<td>6</td>
<td>ā</td>
<td>ā</td>
<td>ā</td>
<td>ā</td>
<td>ē</td>
<td>ē</td>
<td>ē</td>
<td>ē</td>
<td>ơ</td>
<td>ơ</td>
<td>ơ</td>
<td>ơ</td>
</tr>
<tr>
<td>hook above</td>
<td>7</td>
<td>ā</td>
<td>ā</td>
<td>ā</td>
<td>ā</td>
<td>ē</td>
<td>ē</td>
<td>ē</td>
<td>ē</td>
<td>ơ</td>
<td>ơ</td>
<td>ơ</td>
<td>ơ</td>
</tr>
</tbody>
</table>

Table 3: Vietnamese vowels and their marks
c. Identifying capital letter

To identify the capital letter in consonant and syllable, we use a capitals dictionary to store all the capital letters of single consonants and vowels with their marks, similarly to vowels in Table 3, but in capital form. To identify capital letters of consonants, step by step we search each single consonant of this consonant in the capitals dictionary. If it is found in the capitals dictionary, we record the position of this single consonant in the consonant. We use the same method to identify the capital letters of the syllable.

2.1.3 SBV text compression

a. Syllables parser

The syllables parser has been used to separate morphosyllables in the input sequences, splitting morphosyllables into consonants and syllables. It is also used to classify each syllable to the corresponding dictionary and detect the capitalization of characters in consonants and syllables.

We separate morphosyllables based on the spaces character. In this stage, we also classify the morphosyllable to a standard morphosyllable or non-standard morphosyllable based on the Vietnamese dictionary of standard morphosyllables from section 2.1.3. A morphosyllable will be classified as non-standard if it does not appear in this dictionary. Before classifying a morphosyllable as a standard or non-standard morphosyllable, we must convert all characters of morphosyllable to lowercase.

Parse for standard morphosyllables

With each standard morphosyllable received from the separating morphosyllable task, the syllables parser splits it into consonant and syllable, assigns the capital property for them, and classifies them to the corresponding dictionary. This task can be described as follows:

1. Splitting morphosyllables to consonant and syllable is based on the structure of Vietnamese morphosyllable and morphosyllable rules.

2. Adding a position attribute for uppercase characters of consonant and syllable

3. Classifying the syllable into the corresponding dictionary is based on identifying mark rules.

Parse for non-standard morphosyllables

With non-standard morphosyllables, we classify them to one of two classes as follows:

1. Special characters: if one of their characters appears in the special character dictionary.

2. Other: the character does not appear in the special character dictionary.
b. Compression unit

The compression unit uses the results from the syllables parser, detecting consonants and syllables in dictionaries to find their corresponding code. Based on the structure of the syllable, consonant, and property of the character, if it has a capital letter or not. We will use two or three bytes to encode a morphosyllable. The compression task can be summarized as follows:

**Two bytes encoding**

A morphosyllable will be encoded by two bytes in these following cases:

1. A morphosyllable does not have capital letter in it and mark of the syllable is different from a tilde.
2. A special character occurs in the special character dictionary.

The two bytes encoding has a structure like below:

\[ 0 \ B_6^0 \ B_5^0 \ B_4^1 \ B_3^1 \ B_2^1 \ B_1^0 \ B_0^0 \]

Where:

- \( B_6^0 \): two bytes encoding.
- \( B_5^0 \ B_4^1 \ B_3^1 \ B_2^1 \): Encode for the index of the consonant in consonants dictionary.
- \( B_1^0 \) \( B_2^0 \): Encode for the index of the syllable mark in the dictionary from Table 1.
- \( B_7^2 \ B_6^2 \ B_5^2 \ B_4^3 \ B_3^3 \ B_2^3 \ B_1^3 \ B_0^0 \): Encode for the position of the syllable in the dictionary. As presented above, \( B_2^0 \) is used in the case the mark is an acute accent, dot below, or none.
- In the case of a special character, we will set all bits of \( B_6^0 \ B_5^0 \ B_4^1 \ B_3^1 \ B_2^1 \ B_1^0 \) to \( 1 \), \( B_2^2 \ B_2^3 \ B_2^4 \ B_2^5 \ B_2^6 \ B_2^7 \ B_2^8 \ B_2^9 \) will present for the position of special character in its corresponding dictionary.

**Three bytes encoding**

A morphosyllable will be encoded with three bytes when it is a standard morphosyllable and it has at least one capital letter or the mark of the syllable is a tilde. The three bytes encoding has a structure like below:

\[ 1 \ B_6^0 \ B_5^0 \ B_4^1 \ B_3^1 \ B_2^1 \ B_1^0 \ B_0^0 \]

Where:

- \( B_6^0 \ B_5^0 \ B_4^1 \ B_3^1 \ B_2^1 \): Encode for the position of the consonant in the dictionary.
- \( B_1^0 \) \( B_2^0 \): Encode for the capital characters of the consonant. We use three bits because the number characters of a consonant are less than or equal to three.
• $B_6^5 B_5^4$: Encode for mark.
• $B_7^3 B_6^2 B_5^1 B_4^0$: Encode for the position of the syllable in the dictionary.
• $B_3^0 B_2^1 B_1^2 B_0^3$: Encode for the capital characters of the syllable. We use four bits because the number characters of a syllable are less than or equal to four.

Other cases: unknown number of bytes
In the case of a non-standard morphosyllable and it is not a special character, we will encode it with the entire non-standard morphosyllable. To distinguish from the two cases above, we add two more bytes. The first byte has a value of 255 to designate it is a special case of non-standard word. The value of the second byte is the length of a non-standard morphosyllable.

2.1.4 SBV text decompression
SBV text decompression is the inversion of the SBV text compression phase. The SBV text decompression phase is undergone in two steps and can be summarized as follows:

• Code reading unit: This unit is used to read output sequence from the compression result as their input sequence separates it byte by byte.

• Decompression unit:
This unit is used to decode morphosyllables one by one. To do that, it reads one byte from the output sequence of the code reading unit. It analyzes this byte and decides how many bytes will be read more based on the first bit of this byte. There are two cases for this situation.

  – If the first bit of the first byte is 0, the decompression unit will read one byte more from the output sequence of the code reading unit and decodes it. This task is the inversion of two bytes encoding.

  – If the first bit of the first byte is 1:
    * If all remaining bits of the first byte is not equal to 1, the decompression unit reads two bytes more from the output sequence of the code reading unit and decodes it. This task is the inversion of three bytes encoding.
    * If all bits of the first byte are 1: the decompression unit reads one byte more to decide how many bytes will be read more based on the value of this byte. This task is the inversion of the special case of non-morphosyllable encoding.

After finishing the decoding for one morphosyllable, it will read the next byte, repeat the decompression task to decode another morphosyllable until it has read all the way to the last byte.
2.1.5 Compression ratio

Compression ratio is used to measure the efficiency of compression method, the higher the compression ratio the higher quality of compression method. Normally, we use unicode encoding to present Vietnamese text. Every character in Unicode is stored in two bytes. The compression ratio can be calculated by equation 1.

\[ CR = \left(1 - \frac{\text{compressed\_file\_size}}{\text{original\_file\_size}}\right) \times 100 \]  

(1)

Where:

- \( \text{original\_file\_size} = \) (total number of characters in original file) x 2
- \( \text{compressed\_file\_size} = \) total number of bytes in compressed file

2.1.6 Experiments

We conduct experiments to evaluate our method. We present our experiment results in Table 4 and Table 5. We also compress these input files using WinRAR version 5.21\(^3\) (the software combines LZSS [Storer 1982] and Prediction by Partial Matching [Cleary 1984]) and WinZIP version 19.5\(^4\) (the software combines LZ77 [Ziv 1978] and Huffman coding) to have a comparison. Table 4 shows the results of our method in 10 cases with different sizes and content of input files. The size of text files that we use in Table 4 is smaller than 15 KB. According to the results of Table 4 and Figure 2, our compression ratio is better than WinRAR and WinZIP. In these cases, our compression ratio is around 73%. In Table 4 and 5, Figure 2 and 3, we have some abbreviations and meanings as follows: OFS: original file size, CFS: compressed file size, CR: compression ratio.

In Table 5, we show that the result of our method in 10 cases with the size of text files is larger than 15 KB. According to the results, our compression ratio is lower than WinRAR and WinZIP.

\(^3\)http://www.rarlab.com/download.htm
Figure 2: Compression ratios for file size smaller than 15 KB

Table 4: Experimental results for file size smaller than 15 KB

<table>
<thead>
<tr>
<th>No</th>
<th>OFS (Byte)</th>
<th>CFSM (Byte)</th>
<th>CRM (Byte)</th>
<th>CFSR (Byte)</th>
<th>CRR (Byte)</th>
<th>CFSZ (Byte)</th>
<th>CRZ</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>488</td>
<td>134</td>
<td>72.55%</td>
<td>353</td>
<td>27.67%</td>
<td>409</td>
<td>16.19%</td>
</tr>
<tr>
<td>2</td>
<td>708</td>
<td>186</td>
<td>73.73%</td>
<td>421</td>
<td>40.54%</td>
<td>484</td>
<td>31.25%</td>
</tr>
<tr>
<td>3</td>
<td>1,292</td>
<td>335</td>
<td>74.08%</td>
<td>666</td>
<td>48.45%</td>
<td>722</td>
<td>44.12%</td>
</tr>
<tr>
<td>4</td>
<td>1,898</td>
<td>511</td>
<td>73.08%</td>
<td>829</td>
<td>56.33%</td>
<td>886</td>
<td>53.32%</td>
</tr>
<tr>
<td>5</td>
<td>2,698</td>
<td>708</td>
<td>73.76%</td>
<td>1,008</td>
<td>62.64%</td>
<td>1,061</td>
<td>60.68%</td>
</tr>
<tr>
<td>6</td>
<td>5,520</td>
<td>1,445</td>
<td>73.83%</td>
<td>1,816</td>
<td>67.10%</td>
<td>1,870</td>
<td>66.13%</td>
</tr>
<tr>
<td>7</td>
<td>7,422</td>
<td>2,012</td>
<td>72.90%</td>
<td>2,323</td>
<td>68.71%</td>
<td>2,385</td>
<td>67.87%</td>
</tr>
<tr>
<td>8</td>
<td>11,472</td>
<td>3,020</td>
<td>73.68%</td>
<td>3,273</td>
<td>71.47%</td>
<td>3,316</td>
<td>71.10%</td>
</tr>
<tr>
<td>9</td>
<td>13,920</td>
<td>3,676</td>
<td>73.60%</td>
<td>3,802</td>
<td>72.69%</td>
<td>3,836</td>
<td>72.45%</td>
</tr>
<tr>
<td>10</td>
<td>14,858</td>
<td>4,045</td>
<td>72.78%</td>
<td>4,124</td>
<td>72.25%</td>
<td>4,189</td>
<td>71.81%</td>
</tr>
<tr>
<td>No</td>
<td>ORIGINAL FILE SIZE (Byte)</td>
<td>COMPRESSION RATIO OF THREE METHODS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>----</td>
<td>---------------------------</td>
<td>------------------------------------</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>CR of our method</td>
<td>CR of WinRAR</td>
<td>CR of WinZIP</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>16,294</td>
<td>72.79%</td>
<td>73.08%</td>
<td>72.78%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>20,704</td>
<td>73.47%</td>
<td>74.11%</td>
<td>74.18%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>25,470</td>
<td>73.29%</td>
<td>74.58%</td>
<td>74.76%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>32,290</td>
<td>73.48%</td>
<td>76.36%</td>
<td>76.57%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>41,650</td>
<td>73.14%</td>
<td>76.89%</td>
<td>77.28%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>53,374</td>
<td>73.92%</td>
<td>78.04%</td>
<td>78.40%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>66,484</td>
<td>72.94%</td>
<td>77.25%</td>
<td>77.56%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>78,888</td>
<td>73.42%</td>
<td>78.13%</td>
<td>78.30%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>102,014</td>
<td>73.30%</td>
<td>78.55%</td>
<td>78.60%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>151,984</td>
<td>73.32%</td>
<td>79.04%</td>
<td>78.95%</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 5: Experimental results for file size larger than 15 KB

Figure 3: Compression ratios for file size larger than 15 KB
According to all results of our experiments, this method achieves a higher compression ratio when the file size is smaller than 15 KB. Especially, when the file size is smaller than 2.5 KB, the compression ratio of our method is more efficient than WinRAR and WinZIP. In these cases, in comparison with WinRAR and WinZIP, the compression ratio of our method is higher than 10%. Therefore, this method can apply efficiency to compress for Vietnamese short text such as SMS messages and text messages on social networks.

2.2 Trigram-based Vietnamese text compression

Although the compression ratio of the syllable-based method is very high, it converges to a ratio around 73%. Because of the structure of Vietnamese morphosyllables, it is very hard to improve this ratio if we still use this method. Therefore, in this section, we propose a new method for Vietnamese text compression called Trigram-based Vietnamese (TGV) text compression. Figure 4 describes TGV text compression model. In this model, we use a trigrams dictionary for both compression and decompression.

![Diagram of Trigram-based Vietnamese Text Compression](image)

Figure 4: Trigram-based Vietnamese Text Compression

2.2.1 Dictionary

In our method, we build two trigrams dictionaries with different sizes to evaluate the effects of dictionary size with our method. Each dictionary has two columns, one contains trigrams and one contains the index for these trigrams. These dictionaries were built based on a text corpus collected from open access databases. The size of the text corpus for the dictionary one is around 800 MB and for dictionary two is around 2.5 GB. We use SRILM
to generate the trigram data for these dictionaries. The trigrams data after using SRILM for
dictionary one is around 761 MB with more than 40,514,000 trigrams and for dictionary
two is around 1,586 MB with more than 84,000,000 trigrams. To reduce the search time
in dictionaries, we arranged them according to the alphabet. Table 6 describes the size and
number of trigrams of each dictionary.

<table>
<thead>
<tr>
<th>Dictionary</th>
<th>Number of trigrams</th>
<th>Size (MB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>one</td>
<td>40,514,822</td>
<td>761</td>
</tr>
<tr>
<td>two</td>
<td>84,003,322</td>
<td>1,586</td>
</tr>
</tbody>
</table>

Table 6: Dictionaries

2.2.2 TGV text compression

According to Figure 4, the compression phase has two main parts, the first part is a trigrams
parser and the second is a compression unit. In the following subsections, we will explain
them more in detail.

a. Trigrams parser

The trigrams parser is used to read the source text file, separate it into sentences based on
newline and splits all sentence text into trigrams. In the case of the last trigram, maybe it
just has a unigram or bigram. Therefore, we must assign an attribute to it to distinguish the
trigram from the unigram and bigram.

b. Compression unit

The compression unit uses the results from the trigram parser, and detects each trigram in
the dictionary to find the corresponding index for standard trigrams. If a trigram occurs in
dictionary, we encode it using four bytes, otherwise we encode it with exactly the number
of characters that it has. The compression task can be summarized as follows.

Encoding for trigrams in the dictionary

Compression unit searches the trigram in the trigrams dictionary to get the index of tri-
grams if found, otherwise return 0. When a trigram occurs in the dictionary, we use four
bytes to encode it. To distinguish with trigrams that do not occur in the dictionary and
bigram and unigram, the compression unit sets the most significant bit of the first byte to
zero. So, the four bytes encoding has the following structure:

0 B⁶ B⁵ B⁴ B³ B² B¹ B⁰ B⁶ B⁵ B⁴ B³ B² B¹ B⁰ B⁶ B⁵ B⁴ B³ B² B¹ B⁰ B⁶ B⁵ B⁴ B³ B² B¹ B⁰ B⁶ B⁵ B⁴ B³ B² B¹ B⁰ B⁶ B⁵ B⁴ B³ B² B¹ B⁰ B⁶ B⁵ B⁴ B³ B² B¹ B⁰

Where:
• The most significant bit of the first byte is 0: Encode for a trigram which occurs in the dictionary.

• $B_6 B_5 B_4 B_3 B_2 B_1 B_0 B_7 B_6 B_5 B_4 B_3 B_2 B_1 B_0 B_7 B_6 B_5 B_4 B_3 B_2 B_1 B_0$:
  - Encode for the index of the trigram in the dictionary.

Encoding for trigrams do not occur in the dictionary and for other cases

When a trigram does not occur in the dictionary and for other cases, e.g., unigram and bigram, the compression unit encodes it using exactly number of characters that it has. In this case, it sets the most significant bit of the first byte to one. The next seven bits of this byte will present the number of bytes of this trigram and other cases in Unicode encoding because the Vietnamese language was presented in Unicode encoding. So, the encoding structure of this case was described as follows.

$1 B_0 B_5 B_4 B_3 B_2 B_1 B_0 B_7 B_6 B_5 B_4 B_3 B_2 B_1 B_0$.

Where:

• The most significant bit of the first byte is 1: Encode for a trigram which does not occur in the dictionary and for other cases.

• $B_6 B_5 B_4 B_3 B_2 B_1 B_0$:
  - Number of bytes of this trigram or other cases in Unicode encoding.

• $B_7 B_6 B_5 B_4 B_3 B_2 B_1 B_0$:
  - Encoded bytes of trigram characters that do not occur in the dictionary and other cases. For our testing data, we use the Vietnamese language and normally it is presented by Unicode encoding. In the encoding stream, we use Unicode encoding. So, the value of $i$ is the number of bytes that Unicode encoding uses to encode this trigram or other cases.

• We set all values of $B_6 B_5 B_4 B_3 B_2 B_1 B_0$ to 1 to encode for a newline. Therefore, to encode for a newline we just use one byte.

2.2.3 TGV text decompression

TGV text compression is the inversion of TGV compression phase. The TGV text decompression process has undergone in two steps and can be summarized as follows.

• Code reading unit: this unit reads the encoding sequence from TGV text compression, and separates it byte by byte.

• Decompression unit: this unit uses the output sequence from the code reading unit as its input. It decodes one by one trigram or other cases.
2.2.4 Experiments

We conduct experiments to evaluate our method, using a data set that is a randomized collection from Vietnamese text and online newspapers. The data set includes 10 files completely different in size and content.

In order to evaluate the effects of trigrams in the dictionary (the size of dictionary), we conducted two experiments with dictionary one and dictionary two according to Table 6. We show the results of the two experiments in Table 7. Compression ratio was calculated according to Equation 1. According the Table 7, we found that the compression ratio from dictionary two is higher than the compression ratio from dictionary one. If we have a dictionary of all trigrams, the compression ratio will be better.

In Table 7 and Table 7, Figure 5 and 6, we have some abbreviations and meanings as follows: OFS: original file size, CFS: compressed file size, CR: compression ratio, D1: dictionary one, D2: Dictionary two, SB: Syllable-based method.

<table>
<thead>
<tr>
<th>No.</th>
<th>OFS Byte</th>
<th>CFS-D2 Byte</th>
<th>CR of D2</th>
<th>CFS-D1 Byte</th>
<th>CR of D1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1,166</td>
<td>185</td>
<td>84.13%</td>
<td>305</td>
<td>73.84%</td>
</tr>
<tr>
<td>2</td>
<td>2,240</td>
<td>359</td>
<td>83.97%</td>
<td>719</td>
<td>67.90%</td>
</tr>
<tr>
<td>3</td>
<td>6,628</td>
<td>1,710</td>
<td>74.20%</td>
<td>2,404</td>
<td>63.73%</td>
</tr>
<tr>
<td>4</td>
<td>12,224</td>
<td>2,057</td>
<td>83.17%</td>
<td>3,321</td>
<td>72.83%</td>
</tr>
<tr>
<td>5</td>
<td>22,692</td>
<td>3,702</td>
<td>83.69%</td>
<td>7,469</td>
<td>67.09%</td>
</tr>
<tr>
<td>6</td>
<td>49,428</td>
<td>7,870</td>
<td>84.08%</td>
<td>15,872</td>
<td>67.89%</td>
</tr>
<tr>
<td>7</td>
<td>96,994</td>
<td>17,723</td>
<td>81.73%</td>
<td>27,161</td>
<td>72.00%</td>
</tr>
<tr>
<td>8</td>
<td>156,516</td>
<td>27,434</td>
<td>82.47%</td>
<td>41,228</td>
<td>73.66%</td>
</tr>
<tr>
<td>9</td>
<td>269,000</td>
<td>49,902</td>
<td>81.45%</td>
<td>70,105</td>
<td>73.94%</td>
</tr>
<tr>
<td>10</td>
<td>489,530</td>
<td>92,739</td>
<td>81.06%</td>
<td>135,639</td>
<td>72.29%</td>
</tr>
</tbody>
</table>

Table 7: Compression ratio of the dictionary one and the dictionary two

In Figure 5, the compression ratio when we use the dictionary two is higher than the compression ratio of the dictionary one.
Figure 5: Comparison between the dictionary one and the dictionary two

<table>
<thead>
<tr>
<th>No.</th>
<th>OFS</th>
<th>CFS</th>
<th>CR</th>
<th>CFS-SB</th>
<th>CR-SB</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1,166</td>
<td>185</td>
<td>84.13%</td>
<td>345</td>
<td>70.41%</td>
</tr>
<tr>
<td>2</td>
<td>2,240</td>
<td>359</td>
<td>83.97%</td>
<td>599</td>
<td>73.26%</td>
</tr>
<tr>
<td>3</td>
<td>6,628</td>
<td>1,710</td>
<td>74.20%</td>
<td>1,803</td>
<td>72.80%</td>
</tr>
<tr>
<td>4</td>
<td>12,224</td>
<td>2,057</td>
<td>83.17%</td>
<td>3,495</td>
<td>71.41%</td>
</tr>
<tr>
<td>5</td>
<td>22,692</td>
<td>3,702</td>
<td>83.69%</td>
<td>6,418</td>
<td>71.72%</td>
</tr>
<tr>
<td>6</td>
<td>49,428</td>
<td>7,870</td>
<td>84.08%</td>
<td>13,881</td>
<td>71.92%</td>
</tr>
<tr>
<td>7</td>
<td>96,994</td>
<td>17,723</td>
<td>81.73%</td>
<td>26,772</td>
<td>72.40%</td>
</tr>
<tr>
<td>8</td>
<td>156,516</td>
<td>27,434</td>
<td>82.47%</td>
<td>43,701</td>
<td>72.08%</td>
</tr>
<tr>
<td>9</td>
<td>269,000</td>
<td>49,902</td>
<td>81.45%</td>
<td>74,504</td>
<td>72.30%</td>
</tr>
<tr>
<td>10</td>
<td>489,530</td>
<td>92,739</td>
<td>81.06%</td>
<td>139,985</td>
<td>76.25%</td>
</tr>
</tbody>
</table>

Table 8: Compression ratio of current method and syllable-based method
In order to evaluate our improvement method with the previous method, we compressed the input files using the syllable-based method in the previous section. In Table 8 and Figure 6, we show the result of our method in 10 cases above in comparison with the syllable-based method. Our improvement method achieves a higher compression ratio than the syllable-based method.

![Comparison between current method and syllable-based method](image)

Figure 6: Comparison between current method and syllable-based method

### 2.3 N-gram based text compression

In the previous sections we proposed Vietnamese text compression methods that are syllable-based and trigram-based. With the syllable-based method, it bases on the structure of the syllable and the structure of the Vietnamese morphosyllable and converges to a compression ratio around 73%. In the trigram-based method, the compression ratio is better than that of the syllable-based and normally it converges to around 82%. However, in this method, it just focuses on the trigram, and does not care about unigram, bigram, four-gram and five-gram. In this section, we present a method for Vietnamese text compression using n-gram dictionaries. This model has two main modules. The first module is used for text compression and the second module performs decompression. Figure 7 describes n-gram-based text compression model. In this model, we use n-gram dictionaries for both compression and decompression. We will describe the model in detail in the following subsections.
2.3.1 Dictionaries

Since we focus on Vietnamese, we build five different Vietnamese dictionaries of unigram, bigram, trigram, four-gram and five-gram corresponding to the number of grams compressed. Table 9 shows these dictionaries with their number of n-grams and size. These dictionaries have been built based on a text corpus collected from the Internet. The size of the text corpus is around 2.5 GB. We use SRILM\(^5\) to generate n-grams for these dictionaries. To increase the speed of searching in these dictionaries, we arranged them according to the alphabet. Table 9 describes the size and number of n-grams in each dictionary.

<table>
<thead>
<tr>
<th>n-gram dictionary</th>
<th>Number of n-grams</th>
<th>Size (MB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>7,353</td>
<td>0.05</td>
</tr>
<tr>
<td>2</td>
<td>20,498,455</td>
<td>474</td>
</tr>
<tr>
<td>3</td>
<td>84,003,322</td>
<td>1,586</td>
</tr>
<tr>
<td>4</td>
<td>169,916,000</td>
<td>4,155</td>
</tr>
<tr>
<td>5</td>
<td>225,203,959</td>
<td>6,800</td>
</tr>
</tbody>
</table>

Table 9: n-grams dictionaries

\(^5\)http://www.speech.sri.com/projects/srilm/
2.3.2 N-gram based text compression

As presented in Figure 7, the compression module takes a source text as an input, and then pass the text through two sub-modules, i.e., N-grams parser and Compression unit, to compress it. In following subsections, we explain in details.

Algorithm 1: Pseudo-code of the compression phase

```plaintext
input : The source text file
output: The encoded stream

inputstring = read source text file
1 count = number of grams in the inputstring
2 while count ≥ 5 do
3     st5 = get first five grams of the inputstring
4     index = find(st5, five_gram_dict)
5     if index ≥ 0 then
6         force_four_gram_compression(st4)
7         outputstring += compress(index, 5)
8         delete first five grams of the inputstring
9         count -= 5
10     end
11 else
12     st4 += get first gram of the inputstring
13     delete first gram of the inputstring
14     count -= 1
15     if number of grams of st4 = 4 then
16         four_gram_compression(st4)
17     end
18 end
19 if count > 0 then
20     four_gram_compression(inputstring)
21 end
```

a. N-grams parser

N-gram parser has been used to read a source text file, splits it to sentences based on newline and reads the number of grams in the combination with the result of the compression unit. In n-gram parser, we use five kinds of n-gram to store for unigram, bigram, trigram, four-gram and five-gram. Based on the result of the compression unit, the n-gram parser decides how many grams will be read next. Algorithm 1 shows the pseudo-code
of this phase. If five-gram was found in the five-gram dictionary, i.e., index > 0, the 
force_four_gram_compression function would be called to encode all previous n-grams 
(unigram, bigram, trigram and four-gram), then the compress function would be called to 
code this five-gram. Next, the n-gram parser reads next five grams in the input string. 
Otherwise, it would split one leftmost gram of five-gram for four-gram and read one gram 
more from the input string for five-gram. When the number of grams of four-gram was 4, 
it calls the four_gram_compression function.

Algorithm 2: Pseudo-code of the four_gram_compression

```plaintext
input : The four-gram string, in this case is st4
output : The encoded stream

1. index = find(st4, four_gram_dict)
2. if index ≥ 0 then
3.   force_trigram_compression(st3)
4.   outputstring += compress(index, 4)
5.   delete content of st4
6. end
7. else
8.   st3 += first gram of st4
9.   delete first gram of st4
10. if number of grams of st3 = 3 then
11.   trigram_compression(st3)
12. end
13. end
```

Algorithm 3: Pseudo-code of the force_four_gram_compression

```plaintext
input : The four-gram string, in this case is st4
output : The encoded stream

1. while number of grams of st4 > 0 do
2.   st3 += first gram of st4
3.   delete first gram of st4
4.   if number of grams of st3 = 3 then
5.     trigram_compression(st3)
6.   end
7. end
8. force_trigram_compression(st3)
```

Algorithm 2 shows the pseudo-code of the four_gram_compression function. This
function is used to compress four-gram if it occurs in four-gram dictionary. Otherwise, it splits one leftmost gram of the four-gram variable for the trigram variable.

**b. Compression unit**

The compression unit uses the result from the n-gram parser to decide how many grams will be compressed and what kind of n-gram dictionaries should be used. Based on the number of n-grams in each dictionary, we will construct the number of bytes to encode for each n-gram corresponding to the dictionary. Table 10 describes the number of bytes used to encode for each n-gram of each dictionary.

<table>
<thead>
<tr>
<th>N-gram dictionary</th>
<th>Number of n-grams</th>
<th>Number of bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>7,353</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>20,498,455</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>84,003,322</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>169,916,000</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>225,203,959</td>
<td>4</td>
</tr>
</tbody>
</table>

Table 10: number of encoded bytes for each n-gram of each dictionary

To classify the dictionary that was used to encode each n-gram and the other cases, we use three most significant bits (MSB) of the first byte of each encoded byte. Table 11 describes the value of these bits corresponding to each dictionary.

<table>
<thead>
<tr>
<th>N-gram dictionary</th>
<th>Value of three MSB</th>
<th>Number of bytes is read more</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0 0 1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>0 1 0</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>0 1 1</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>1 0 0</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>1 0 1</td>
<td>3</td>
</tr>
<tr>
<td>newline</td>
<td>1 1 0</td>
<td>0</td>
</tr>
<tr>
<td>others</td>
<td>1 1 1</td>
<td>value of five bits after three first bits of current byte</td>
</tr>
</tbody>
</table>

Table 11: value of three MSB and number of bytes
2.3.3 N-gram based text decompression

As seen in Figure 7, the decompression module takes an compressed text as an input, and then pass the text through two sub-modules, i.e., Code reading unit and Decompression unit, to decompress it. We explain in detail in following subsections.

Algorithm 4: Pseudo-code of the decompression phase

```
input : The encoded stream
output : The decoded stream

inputstring ← encodedstream

while length of inputstring > 0 do
    firstbyte = read first byte from the inputstring
    delete first byte of the inputstring
    dict = get value of three bits of firstbyte
    if dict ≤ 5 then
        number = getnumberbytreadmore(dict)
        bytreadmore = read number byte more from the inputstring
        delete number byte of the inputstring
        indexstring = get last five bits of the firstbyte + the bytreadmore
        indexvalue = get value of the indexstring
        output += decompress(indexvalue, dict)
    else if dict = 6 then
        output += newline
    else
        number = value of five last bits of the firstbyte
        bytreadmore = read number byte more from the inputstring
        output += decode for the bytreadmore
    end
end
```

a. Code reading unit

First, this unit reads the compressed text from the compression phase. This result becomes the input sequence of the code reading unit. The code reading unit splits this input sequence byte to byte. Then, it reads the first byte of the input sequence, splits and analyzes the first three bits of this byte to classify the dictionary to which this n-gram belongs. Based on this result, this unit will read more bytes from the input sequence.
b. Decompression unit

This unit receives the results from the code reading unit. It decodes these results according to the classification of the dictionary as follows.

- **Decode for n-grams occurring in dictionaries**
  - Identifying the dictionary: based on the classification dictionary from the code reading unit.
  - Identifying the index of an n-gram in the dictionary: based on the value calculated from bytes that were read by the code reading unit.
  - Decode for n-gram: when the classification of the dictionary has a value from one to five, the decompression unit decodes the n-gram in the dictionary based on the index of the n-gram.

- **Decode for n-grams that don’t occur in dictionaries**
  - Decode for newline: when the classification of dictionary is a “newline”, it means that the value of the first three bits is 110. The decompression unit decodes a newline for this n-gram.
  - Decode for others: when the classification of the dictionary is an “others”, based on the value of the remaining bits of the first byte, the decompression unit will decode for all bytes after the first byte.

After finishing the decoding for one n-gram or other cases, the decompression unit reads the next result from the code reading unit and repeats the decompression tasks to decode for other n-grams or other cases until it reads the last byte. Algorithm 4 shows the pseudo-code of the decompression phase.

2.3.4 Experiments

We conducted an experiment to evaluate this method, using a data set that is randomized collection from some Vietnamese news agencies. The data set includes 10 files completely different in size and content.

In order to evaluate the effects of a combination of various n-gram dictionaries, we conducted three experiments with three kinds of systems. In the first case, we build a system with unigram, bigram, and trigram dictionaries. Next, we extend the first one with four-gram dictionary. Lastly, we extend the second one with five-gram dictionary. The results of the three experiments is shown in Table 12. As presented in Table 12, we find out that the compression ratio from the third case is the best, follow-up is the second case, and the last one comes from the first case. The compression ratio in this section was used according to Equation 1. In Table 12, 13, 14, Figure 8, 9, 10, we have some abbreviations and meanings as follows: OFS: original file size in byte; CFS: compressed file size in
byte; CR: compression ratio; C1, C2, C3: three cases above, respectively; O: our method, S: syllable-based method, T: trigram-based method, RAR: WinRAR, ZIP: WinZIP.

<table>
<thead>
<tr>
<th>No.</th>
<th>OFS</th>
<th>CFS-C1</th>
<th>CR-C1</th>
<th>CFS-C2</th>
<th>CR-C2</th>
<th>CFS-C3</th>
<th>CR-C3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1,166</td>
<td>210</td>
<td>81.99%</td>
<td>166</td>
<td>85.76%</td>
<td>136</td>
<td>88.34%</td>
</tr>
<tr>
<td>2</td>
<td>2,240</td>
<td>362</td>
<td>83.84%</td>
<td>274</td>
<td>87.77%</td>
<td>222</td>
<td>90.09%</td>
</tr>
<tr>
<td>3</td>
<td>6,628</td>
<td>1,245</td>
<td>81.22%</td>
<td>999</td>
<td>84.93%</td>
<td>887</td>
<td>86.62%</td>
</tr>
<tr>
<td>4</td>
<td>12,224</td>
<td>1,954</td>
<td>84.02%</td>
<td>1,503</td>
<td>87.70%</td>
<td>1,179</td>
<td>90.36%</td>
</tr>
<tr>
<td>5</td>
<td>22,692</td>
<td>3,565</td>
<td>84.29%</td>
<td>2,652</td>
<td>88.31%</td>
<td>2,180</td>
<td>90.39%</td>
</tr>
<tr>
<td>6</td>
<td>49,428</td>
<td>7,638</td>
<td>84.55%</td>
<td>5,712</td>
<td>88.44%</td>
<td>4,538</td>
<td>90.82%</td>
</tr>
<tr>
<td>7</td>
<td>96,994</td>
<td>15,636</td>
<td>83.88%</td>
<td>12,359</td>
<td>87.26%</td>
<td>10,416</td>
<td>89.26%</td>
</tr>
<tr>
<td>8</td>
<td>156,516</td>
<td>24,974</td>
<td>84.04%</td>
<td>19,188</td>
<td>87.74%</td>
<td>15,889</td>
<td>89.85%</td>
</tr>
<tr>
<td>9</td>
<td>269,000</td>
<td>43,887</td>
<td>83.69%</td>
<td>34,182</td>
<td>87.29%</td>
<td>28,937</td>
<td>89.24%</td>
</tr>
<tr>
<td>10</td>
<td>489,530</td>
<td>80,685</td>
<td>83.52%</td>
<td>63,472</td>
<td>87.03%</td>
<td>54,117</td>
<td>88.95%</td>
</tr>
</tbody>
</table>

Table 12: Compression ratio of three experience cases

Figure 8: Comparison between the three cases
In Figure 8, the compression ratio when we combine all five dictionaries is the highest.

<table>
<thead>
<tr>
<th>No.</th>
<th>OFS</th>
<th>CFS-S</th>
<th>CR-S %</th>
<th>CFS-T</th>
<th>CR-T %</th>
<th>CFS-O</th>
<th>CR-O %</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1,166</td>
<td>345</td>
<td>70.41%</td>
<td>185</td>
<td>84.13%</td>
<td>136</td>
<td>88.34%</td>
</tr>
<tr>
<td>2</td>
<td>2,240</td>
<td>599</td>
<td>73.26%</td>
<td>359</td>
<td>83.97%</td>
<td>222</td>
<td>90.09%</td>
</tr>
<tr>
<td>3</td>
<td>6,628</td>
<td>1,803</td>
<td>72.80%</td>
<td>1,710</td>
<td>74.20%</td>
<td>887</td>
<td>86.62%</td>
</tr>
<tr>
<td>4</td>
<td>12,224</td>
<td>3,495</td>
<td>71.41%</td>
<td>2,057</td>
<td>83.17%</td>
<td>1,179</td>
<td>90.36%</td>
</tr>
<tr>
<td>5</td>
<td>22,692</td>
<td>6,418</td>
<td>71.72%</td>
<td>3,702</td>
<td>83.69%</td>
<td>2,180</td>
<td>90.39%</td>
</tr>
<tr>
<td>6</td>
<td>49,428</td>
<td>13,881</td>
<td>71.92%</td>
<td>7,870</td>
<td>84.08%</td>
<td>4,538</td>
<td>90.82%</td>
</tr>
<tr>
<td>7</td>
<td>96,994</td>
<td>26,772</td>
<td>72.40%</td>
<td>17,723</td>
<td>81.73%</td>
<td>10,416</td>
<td>89.26%</td>
</tr>
<tr>
<td>8</td>
<td>156,516</td>
<td>43,701</td>
<td>72.08%</td>
<td>27,434</td>
<td>82.47%</td>
<td>15,889</td>
<td>89.85%</td>
</tr>
<tr>
<td>9</td>
<td>269,000</td>
<td>74,504</td>
<td>72.30%</td>
<td>49,902</td>
<td>81.45%</td>
<td>28,937</td>
<td>89.24%</td>
</tr>
<tr>
<td>10</td>
<td>489,530</td>
<td>139,985</td>
<td>71.40%</td>
<td>92,739</td>
<td>81.06%</td>
<td>54,117</td>
<td>88.95%</td>
</tr>
</tbody>
</table>

Table 13: Compression ratio of the current method with the two previous methods

Figure 9: Comparison of compression ratio of three methods
In order to evaluate our method with the two previous methods, we compress the input files using these methods. As seen in Table 13 and Figure 9, the compression ratio of this method is better than the two previous methods for any size of text in our test cases.

<table>
<thead>
<tr>
<th>No.</th>
<th>OFS</th>
<th>CFS-O</th>
<th>CR-O</th>
<th>CFS-RAR</th>
<th>CR-RAR</th>
<th>CFS-ZIP</th>
<th>CR-ZIP</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1,166</td>
<td>136</td>
<td>88.34%</td>
<td>617</td>
<td>47.08%</td>
<td>676</td>
<td>42.02%</td>
</tr>
<tr>
<td>2</td>
<td>2,240</td>
<td>222</td>
<td>90.09%</td>
<td>887</td>
<td>60.40%</td>
<td>946</td>
<td>57.77%</td>
</tr>
<tr>
<td>3</td>
<td>6,628</td>
<td>887</td>
<td>86.62%</td>
<td>2,052</td>
<td>69.04%</td>
<td>2,111</td>
<td>68.15%</td>
</tr>
<tr>
<td>4</td>
<td>12,224</td>
<td>1,179</td>
<td>90.36%</td>
<td>3,378</td>
<td>72.37%</td>
<td>3,442</td>
<td>71.84%</td>
</tr>
<tr>
<td>5</td>
<td>22,692</td>
<td>2,180</td>
<td>90.39%</td>
<td>6,162</td>
<td>72.85%</td>
<td>6,150</td>
<td>72.90%</td>
</tr>
<tr>
<td>6</td>
<td>49,428</td>
<td>4,538</td>
<td>90.82%</td>
<td>12,504</td>
<td>74.70%</td>
<td>12,286</td>
<td>75.14%</td>
</tr>
<tr>
<td>7</td>
<td>96,994</td>
<td>10,416</td>
<td>89.26%</td>
<td>21,389</td>
<td>77.95%</td>
<td>21,321</td>
<td>78.02%</td>
</tr>
<tr>
<td>8</td>
<td>156,516</td>
<td>15,889</td>
<td>89.85%</td>
<td>34,162</td>
<td>78.17%</td>
<td>34,362</td>
<td>78.05%</td>
</tr>
<tr>
<td>9</td>
<td>269,000</td>
<td>28,937</td>
<td>89.24%</td>
<td>56,152</td>
<td>79.13%</td>
<td>57,671</td>
<td>78.56%</td>
</tr>
<tr>
<td>10</td>
<td>489,530</td>
<td>54,117</td>
<td>88.95%</td>
<td>101,269</td>
<td>79.31%</td>
<td>108,175</td>
<td>77.90%</td>
</tr>
</tbody>
</table>

Table 14: Compression ratio of our method, WinRAR and WinZIP

Figure 10: Compression ratio of our method, WinRAR and WinZIP
Table 14 and Figure 10 show the results of our method in comparison with those of other methods, such as WinZIP version 19.5, the software combines LZ77 [Ziv 1978] and Huffman coding, and WinRAR version 5.21, the software combines LZSS [Storer 1982] and Prediction by Partial Matching [Cleary 1984]. The experimental results show that our method achieves the highest compression ratio on the same testing set.

In this section, we presented some methods to Vietnamese text compression. Each method has its own strengths and weaknesses. The application of each method depends on the purpose of the users. The syllable-based method uses a small dictionary of all Vietnamese morphosyllables with more than 7,300 morphosyllables, and the compression ratio of this method converges to 73%. This method is useful when applied to small files, personal systems, or systems that focus more on the speed of compression. The trigram- and n-gram-based methods are suitable for systems that need a high compression ratio, however.

3 Normalization of Vietnamese informal text

In this section, we present a method for normalization of Vietnamese informal text, focused on Vietnamese tweets. This method has three main parts, preprocessing Vietnamese tweets, detecting spelling errors, and error correction. Figure 11 describes model. We describe more detail in the following subsections.

Figure 11: Normalization model for Vietnamese tweets

---

7 http://www.rarlab.com/download.htm
3.1 Preprocessing

3.1.1 Clean up noisy symbols

The original tweet can contain various noisy content, such as emotion symbols, e.g., ❤️❤️; the hashtag symbol, e.g., #news,#movie; link urls, e.g., https://t.co/b02R1yXrp, https://t.co/7qod8hy1ZC; etc. Those noisy symbols can affect the accuracy of the system. Therefore, in our system, first, we clean up those noisy symbols.

3.1.2 Clean up repeated characters

Many tweets have repeated characters, e.g., Anh yêu yêu yêu yêu yêu yêu yêu, to express the user’s feelings. These repeated characters do not have meaning and affect the accuracy of the system. Therefore, we also need to clean up these tweets.

3.2 Spelling errors detection

Spelling errors are a big problem for any information extraction system. There are several methods to detect spelling errors. In our method, a morphosyllable in a tweet will be identified as an error if it does not appear in the standard morphosyllables dictionary. After a morphosyllable in a tweet is identified as an error, it will be analyzed to classify the error and process it. Normally, Vietnamese morphosyllables in tweets include two kind of errors, i.e., typing errors and spelling errors. We will describe in detail the two kinds of errors in the following subsections.

3.2.1 Typing error

Vietnamese uses Telex typing and VNI typing to compose Vietnamese tweets. Each method has its own combination to forming syllables and their marks. Tweets are very short and are prepared quickly, so typing speed can cause errors. For example:

- With the morphosyllable, “Nguyễn,” we could have typing errors such as “nguy-eenx,” “nguyênx,” or “nguyeenxx” with Telex typing, and “nguye6n4,” “nguyên4,” or “nguye6n44” with VNI typing.
- With the morphosyllable, “người”, we could type the followings: “ngươîf,” “nguôfi,” “nguowfi,” “nguowfi,” “nguoîf,” “nguoiw,” or “nguoîf” with VNI typing.

To handle this issue, we built a set of syllable rules with their marks and a set of rules to map these syllables to their errors, as shown in the following examples:

- “án”: “asn,” “ans,” “a1n,” or “an1”
• “àn”: “afn,” “anf,” “a2n,” or “an2”
• “ẳn”: “arn,” “anr,” “a3n,” or “an3”
• “ẳn”: “axn,” “anx,” “a4n,” or “an4”
• “ẳn”: “ajn,” “anj,” “a5n,” or “an5”

3.2.2 Spelling errors
Spelling errors occur frequently in Vietnamese tweets. Normally, they occur due to mistakes in pronunciation.

3.3 Error correction
For the detected typing and spelling errors, first, the system uses vocabulary structures and the set of syllable rules to normalize them. Then the system uses trigrams dictionary to normalize these results based on the degree of similarity between them.

3.3.1 Similarity of two morphosyllables
To measure the similarity of two morphosyllables, we used the results in the research of Dice [Dice 1945] with some improvements we made. To use Dice’s research, we split all the characters of the morphosyllable to bigrams. Assuming that we have two morphosyllables, i.e., “nguyen” and “nguye,” the bigrams of these morphosyllables can be represented as follows: bigram_{nguyen}={ng, gu, uy, yn}, and bigram_{nguyen}={ng, gu, uy, ye, en}.

Dice Coefficient:
The Dice coefficient, developed by Lee Raymond Dice [Dice 1945], is a statistical approach for comparing the similarity of two samples. The Dice coefficient of the two morphosyllables, \( w_i \) and \( w_j \), according to bigram can be calculated using equation 1:

\[
Dice(w_i, w_j) = \frac{2 \times \left| \text{bigram}_{w_i} \cap \text{bigram}_{w_j} \right|}{\left| \text{bigram}_{w_i} \right| + \left| \text{bigram}_{w_j} \right|}
\]

(2)

where:

• \( \left| \text{bigram}_{w_i} \right| \) and \( \left| \text{bigram}_{w_j} \right| \) are the total bigrams of \( w_i \) and \( w_j \)

• \( \left| \text{bigram}_{w_i} \right| \cap \left| \text{bigram}_{w_j} \right| \) are the number of bigrams which appear in \( w_i \) and \( w_j \) at the same time.

If two morphosyllables are the same, the Dice coefficient is 1. The higher of the Dice coefficient, the higher the degree of similarity and vice versa.
Proposed method to improve the Dice Coefficient:

As observed from the experimental data using the Dice coefficient, we found that, the above method will be accurate with misspelled morphosyllables that have the misspelled character at the end. When misspelled characters occur close to the last character, at least we will lose the similarity of the last two grams. For a morphosyllable that has three characters, the degree of similarity is 0. For example: Dice("rát", "rát") = 0; Dice("gân", "gần") = 0;

From the above problem, we proposed a method to improve the Dice coefficient. The improvement of coefficient was performed by combining the first character with the last character of the two morphosyllables to form a new pair of bigrams. If the two members of this pair are different, the system will use the coefficients as shown in equation (1). In contrast, we use equation (2) as below:

\[ iDice(w_i, w_j) = \frac{2 \times (|bigram_{wi} \cap bigram_{wj}| + 1)}{|bigram_{wi}| + |bigram_{wj}| + 2} \]  

Let \( fbigram_w \) be an additional bigram of \( w \). Each \( fbigram \) is the pair of the first and the last character of \( w \). We can express the formula for improving the Dice coefficient as equation (3):

\[ fDice(w_i, w_j) = \begin{cases} 
  Dice(w_i, w_j) & \text{if } fbigram_{wi} \text{ is different from } fbigram_{wj} \\
  iDice(w_i, w_j) & \text{Otherwise}
\end{cases} \]  

To illustrate the improvement of the Dice coefficient, we assumed that we have two morphosyllables to measure the degree of similarity, i.e., "nguyen" and "nguyn," as presented in the previous section, thus we have \(|bigram_{wi} \cap bigram_{wj}| = 3\). Combining the first and the last characters of the two morphosyllables we have the new pair of bigram, which has the same result, i.e., "nn." So, using the improvement of the Dice coefficient, we have \( fDice(\text{"nguyen"}, \text{"nguyn"}) = 0.727 \). If we use the normal coefficient of Dice we have \( Dice(\text{"nguyen"}, \text{"nguyn"}) = 0.667 \).

3.3.2 Similarity of two sentences

Assume that we need to measure the similarity of two sentences, i.e., \( S_1 = w_1, w_2, w_3, \ldots, w_n \) and \( S_2 = w'_1, w'_2, w'_3, \ldots, w'_n \). We compare the similarity of each pair of morphosyllables according to the improved Dice coefficient. Then, we compute the similarity of the two sentences by Equation 4.4:

\[ Sim(S_1, S_2) = \frac{\sum_{i=1}^{n} fDice(w_i, w'_i)}{n} \]
n is the number of morphosyllables

If two sentences are the same, their degree of similarity (Sim) is 1. The higher the Sim coefficient, the higher the degree of similarity becomes, and vice versa. Table 15 shows the results of the normalization of Vietnamese tweets that have spelling errors.

Table 15: tweets with spelling errors and their normalization.

<table>
<thead>
<tr>
<th>Spelling error tweets</th>
<th>Normalized tweets</th>
</tr>
</thead>
<tbody>
<tr>
<td>xe đón hồ ngọc hà gây tai nạn kinhh</td>
<td>xe đón hồ ngọc hà gây tai nạn</td>
</tr>
<tr>
<td>hoàng: sẽ khởi tố tài xe</td>
<td>sê khởi tố tài xe (the car picked</td>
</tr>
<tr>
<td><a href="http://fb.me/2MwvznBbj">http://fb.me/2MwvznBbj</a></td>
<td>up ho ngoc ha caused a terrible</td>
</tr>
<tr>
<td></td>
<td>accident: the driver will</td>
</tr>
<tr>
<td></td>
<td>be prosecuted)</td>
</tr>
<tr>
<td>hôm nay, sinh viên ddaijj học tồn</td>
<td>hôm nay, sinh viên đại học tồn</td>
</tr>
<tr>
<td>dudwess</td>
<td>được nghỉ học</td>
</tr>
<tr>
<td></td>
<td>(today, students of ton duc thang</td>
</tr>
<tr>
<td></td>
<td>university were allowed to absent)</td>
</tr>
</tbody>
</table>

3.4 Experiments

To evaluate our method, we used a data set which randomizes collected Vietnamese tweets. The data set includes 1,360 tweets that are completely different from each other.

In order to make comparisons of the impact of the data set in the language model, we ran the test two times with the language model built from two input data sets: The first set includes 130 MB randomized data from 1,045 MB of data set mention above and the second set includes entire 1,045 MB data. The trigram model with a frequency of more than 5 times the first set is about 8 MB. In this case, we use the improved Dice coefficient to measure the similarity of the two sentences. In this test, we use the precision metric to evaluate our method.

- Precision (P): number of correctly fixed errors divided by the total number of errors detected.

The results of this test was shown in Table 16. From Table 16, the results of the trigram model with data from the second set achieved a higher accuracy than the results of the trigram model with data from the first set.
Table 16: The results using fDice with two data sets in the trigram model

<table>
<thead>
<tr>
<th>Data set</th>
<th>Total error</th>
<th>Detected error</th>
<th>Correct fixed</th>
<th>Wrong fixed</th>
<th>Precision</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1,360</td>
<td>1,342</td>
<td>1,072</td>
<td>270</td>
<td>79.88%</td>
</tr>
<tr>
<td>2</td>
<td>1,360</td>
<td>1,342</td>
<td>1,207</td>
<td>135</td>
<td>89.94%</td>
</tr>
</tbody>
</table>

To evaluate the improvement Dice coefficient with normal Dice coefficient. We ran the test with trigram model built from entire data set, i.e., the data set of 1,045 MB, using Dice and fDice to measure the similarity of two sentences. In this test, we use two more metrics Recall and Balance F-Measure with the precision metric previously mentioned above to evaluate our method.

- Recall (R): number of correctly fixed errors divided by the total error.
- Balance F-measure (F1): $F1 = \frac{2 \times P \times R}{P + R}$

Table 17 shows the results of this test. The table shows that the combination of our improved Dice coefficient and the trigram model achieved better performance than the normal Dice coefficient with the trigram model.

Table 17: The results use fDice and Dice with trigram language model

<table>
<thead>
<tr>
<th>Method</th>
<th>Precision</th>
<th>Recall</th>
<th>F-Measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dice</td>
<td>84.8%</td>
<td>83.68%</td>
<td>84.23%</td>
</tr>
<tr>
<td>fDice</td>
<td>89.94%</td>
<td>88.75%</td>
<td>89.34%</td>
</tr>
</tbody>
</table>

In this section, we presented the first attempt to normalize Vietnamese informal text focused on tweets on Twitter. Our proposed method combines a language model with dictionaries and Vietnamese vocabulary structures. We also extended the original Dice coefficient to improve performance of the similarity measure between two morphosyllables. To evaluate the proposed method, we built a dataset including 1,360 Vietnamese tweets. The experiment results show that our proposed method achieves relative high performance with precision approximating to 90%, recall over 88.7%, and FMeasure over 89%. Moreover, our improvement on measuring the similarity of the two morphosyllables based on the Dice coefficient outperforms the original Dice coefficient.
4 Named entity recognition in Vietnamese informal text

In this section, we present a model for NER in Vietnamese tweets. This model has two main parts, i.e., one for training and one for recognizing. Figure 12 describes our model. In our model, the gazetteers are used for both training and recognizing. We will provide more detail in the following subsections.

![NER model diagram]

Figure 12: NER model in Vietnamese tweets

4.1 Normalization

As presented in previous section, Vietnamese tweets on Twitter are noisy, irregular, brief and consist of acronyms and spelling errors. Therefore, we must normalize them before using NER.

4.2 Capitalization classifier

Capitalization is a key orthographic feature for recognizing named entities [Florian 2002, Downey 2007]. Unfortunately, in tweets, capitalization is much less reliable than in edited
texts. Users usually compose and reply to messages quickly, and they do not care much about capitalization. According to [Chu 2010], a letter is capitalized in the following cases:

1. Capitalize the first letter of the first syllable of a complete sentence, after punctuation (.), question mark (?), exclamation point (!), ellipsis (...) and new line.

2. Capitalize the name of people, locations, and organizations.

3. Other cases of capitalization include, e.g., medal name, position name, days of the week, months of the year, holidays, names of books, and names of magazines.

Because our method focuses on three types of entities, i.e., person, organization, and location, in the capitalization classifier, we take the first and the second cases into account.

4.3 **Word segmentation and part of speech (POS) tagging**

4.3.1 **Word segmentation**

Vietnamese is different from English and other languages in word segmentation. In English and other languages, words can be separated based on the space character. However, Vietnamese is not like that. A Vietnamese word is composed of special linguistic units called Vietnamese morphosyllable. Normally, a word has from one to four morphosyllables. In our method, we used vnTokenizer\(^8\) of [Le 2008] for word segmentation.

4.3.2 **POS tagging**

After performing word segmentation, we apply POS tagging to give more information to the next phase. POS tagging was used to identify the characteristics of the word to enhance the accuracy of the NER system.

4.4 **Extraction of features**

The aim of this phase is to convert each word to a vector of feature values. Our system uses the IOB model to annotate and assign label to data in the training and classification phases. IOB is expressed as follows:

- I: current morphosyllable is inside of a named entity (NE).
- O: current morphosyllable is outside of a NE.
- B: current morphosyllable is the beginning of a NE

\(^8\)http://mim.hus.vnu.edu.vn/phuonglh/softwares/vnTokenizer
The selection of specific attributes from the training set has a key role in identifying the type of entity. Since the nature of the Vietnamese language is different from English, we used the most appropriate and reasonable features in order to achieve optimum accuracy for the system. Our system uses the following features:

1. **Word position**: the position of word in a sentence.
2. **POS**: POS tag of the current word.
3. **Orthographic (ORT)**: In this feature, we focused on several cases of orthographies, such as capitalization of the first character of first morphosyllable, capitalization of first character of each morphosyllable, capitalization of all letters, lowercase, punctuation and numbers.
4. **Gazetteer**: This feature was built based on a Gazetteer which consists of several dictionaries. Each dictionary contains words in specific types of categories such as person name, organization name, location name, prefixes, etc.
5. **Prefix, Suffix**: the first and the second character; the last and the next to the last character of the current word.
6. **POS Prefix, POS Suffix**: POS tags of two previous words and POS tags of two following words of the current word.

### 4.5 NER training set

In Figure 12, before performing feature extraction, we perform word segmentation, POS tagging, and assigning labels for each word in the training set. Then, the system extracts features of the words and represents each of those words as a feature vector. A support vector machine learning algorithm was used to train the model using the training set.

In particular, we assigned labels for words in the training set by using a semi-automatic program, meaning that we assigned labels to those words with a program we wrote and checked in hand. In our self-written program, we considered the noun phrase obtained after the tagging step with a list of dictionary of text files to label for those words. The text files of the dictionary contain:

- The noun prefix for people such as you, sister, uncle, and president
- The noun prefix for organizations such as company, firm, and corporation
- The noun prefix for locations such as province, city, and district
- List of dictionary for states, provinces of Vietnam, and others

After assigning labels for words in Vietnamese tweets, we analyzed these tweets to build feature vectors for those words. The structure of a feature vector includes: `<label> <index1>:<value1> <index2>:<value2> <index3>:<value3> and other pairs, where:
• `<label>`: value from 1 to 7 according to 7 labels (O, B-PER, I-PER, B-LOC, I-LOC, B-ORG, I-ORG).

• `<index>:<value>`: order of feature and value corresponding to feature of a word, respectively.

To understand the process of preparation data for SVM format, we consider the following example with sentence after word segmentation above `Sinh_viên Trường Đại_học Tôn_Đức_Thắng` with four basic features: word position, POS, orthographic, and Gazetteer. The result of this process was presented in Table 18.

Table 18: The result of the process of preparation data for SVM format

<table>
<thead>
<tr>
<th>Word</th>
<th>IOB label</th>
<th>POS label</th>
<th>ORT label</th>
<th>Gazetteer label</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Sinh_viên</code></td>
<td>O</td>
<td>1</td>
<td>N</td>
<td>I_Cap</td>
<td>4</td>
</tr>
<tr>
<td><code>Trường</code></td>
<td>B-ORG</td>
<td>6</td>
<td>N</td>
<td>I_Cap</td>
<td>4</td>
</tr>
<tr>
<td><code>Đại_học</code></td>
<td>I-ORG</td>
<td>7</td>
<td>N</td>
<td>I_Cap</td>
<td>4</td>
</tr>
<tr>
<td><code>Tôn_Đức_Thắng</code></td>
<td>I-ORG</td>
<td>7</td>
<td>Np</td>
<td>AF_Cap</td>
<td>4</td>
</tr>
<tr>
<td><code>Đại_học</code></td>
<td>I-ORG</td>
<td>7</td>
<td>N</td>
<td>I_Cap</td>
<td>4</td>
</tr>
</tbody>
</table>

In this sentence, `Sinh_viên Trường Đại_học Tôn_Đức_Thắng`, the position of `Sinh_viên` is 1, position of `Trường` is 2, position of `Đại_học` is 3, position of `Tôn_Đức_Thắng` is 4. Assuming that the order of features is word position, POS, ORT, and Gazetteer, the feature vectors of the words in the sentence above are presented in Table 19.

Table 19: The result of feature vectors

<table>
<thead>
<tr>
<th>Word</th>
<th>Feature vector</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Sinh_viên</code></td>
<td>1 1:1 2:4 3:1 4:4</td>
</tr>
<tr>
<td><code>Trường</code></td>
<td>6 1:2 2:4 3:1 4:4</td>
</tr>
<tr>
<td><code>Đại_học</code></td>
<td>7 1:3 2:4 3:1 4:4</td>
</tr>
<tr>
<td><code>Tôn_Đức_Thắng</code></td>
<td>7 1:4 2:1 3:4 4:1</td>
</tr>
</tbody>
</table>

After representing words in the training set as feature vectors, we used libSVM⁹ to train the model.

⁹https://www.csie.ntu.edu.tw/~cjlin/libsvm/
4.6 Experiments

We conducted experiments with a test set including 1,668 Vietnamese tweets. To evaluate the NER method and make a comparison of the impact of the normalization of the test set, we conducted two experiments, i.e., one without normalization and capitalization classifier of tweets (Case 1) and one with normalization and capitalization classifier of tweets (Case 2). Table 20 shows our experimental results. In this case, we also used three metrics to evaluate our method, i.e., the precision, the recall, and the Balance F-Measure.

- **Precision (P):** the number of correctly recognized named entities divided by the total number of named entities recognized by the NER system.
- **Recall (R):** the number of correctly recognized named entities divided by the total number of named entities in the test set.
- **Balance F-Measure (F1):**
  
  \[ F1 = \frac{2 \times P \times R}{P + R} \]

<table>
<thead>
<tr>
<th>Case</th>
<th># NEs in testing set</th>
<th># #</th>
<th>correctly recognized NEs</th>
<th># wrong recognized NEs</th>
<th>P</th>
<th>R</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2,446</td>
<td>1,915</td>
<td>1,601</td>
<td>314</td>
<td>83.6%</td>
<td>65.45%</td>
<td>73.42%</td>
</tr>
<tr>
<td>2</td>
<td>2,446</td>
<td>2,266</td>
<td>1,939</td>
<td>327</td>
<td>85.57%</td>
<td>79.27%</td>
<td>82.3%</td>
</tr>
</tbody>
</table>

According to Table 20, when we applied the normalization to the test set, the precision, recall and balance F-Measure of this test were higher than the case of the test set without the application of normalization.

We re-implemented the state-of-the-art method proposed in [Tran 2007] and compared its performance with the performance of our method. The results of this comparison are shown in Table 21.

Table 21: Comparison performance of our method with that of [Tran 2007]

<table>
<thead>
<tr>
<th>System</th>
<th>Precision</th>
<th>Recall</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Our system</td>
<td>85.57%</td>
<td>79.27%</td>
<td>82.3%</td>
</tr>
<tr>
<td>System of [Tran 2007]</td>
<td>83.20%</td>
<td>76.20%</td>
<td>79.55%</td>
</tr>
</tbody>
</table>
In this section, we presented the first attempt to use NER in Vietnamese informal text focused on Vietnamese tweets on Twitter. We proposed a learning model based on SVM to recognize named entities using six different types of features. In our method, we also proposed a method to capitalize for suitable characters in tweets to enhance the accuracy of the system. To evaluate the method, we built a training set of more than 40,000 named entities and a testing set of 2,446 named entities. The experimental results showed that our system achieved encouraging performance, with an 82.3% F1 score.

5 Conclusion

In the first part of this thesis, in Section 2, we presented our achievements in text compression. Section 2.1 proposed a method for Vietnamese text compression that focused on Vietnamese morphosyllables structure, a syllable structure used to compress Vietnamese text. In Section 2.2, we proposed another method for Vietnamese text compression based on trigram. The last method that we pointed out in Section 2.3 was an n-gram based. This method achieves the best compression ratio when compared with the two previous methods and it can apply to any size of text file. In the next part, in Section 4, we presented our achievements in Named Entity Recognition (NER) for Vietnamese informal text on social networks, focused on Twitter. We also presented a method to normalize Vietnamese informal text in Section 3 in combination with the NER model to achieve higher precision. In the next section we present in detail the main achievements of this thesis.

5.1 Contributions to text compression

In this field, we present the first attempt at Vietnamese text compression. We present three methods and achieved some encouraging results based on compression ratio. In these methods, the last method achieves the best result in terms of compression ratio.

1. First we proposed a method for Vietnamese text compression based on Vietnamese morphosyllable structure, Vietnamese syllables, consonants, and vowels. To identify syllables and their marks, we built six dictionaries corresponding to six types of marks. We also proposed a method to recognize marks and capital letters. Our method gives a new approach to classifying capital letters and compressing them.

2. Next, we presented a method for Vietnamese text compression based on the trigram model. This method splits input sequences into trigrams and compresses them based on a trigrams dictionary.

3. The last method we proposed for text compression is n-gram based. In this method, we used five dictionaries from uni-gram to five-gram. We used a slide window to identify the n-gram from input sequences and compressed them corresponding to an n-gram dictionary. This method achieved a the highest compression ratio when compared with the two previous methods. It can apply to text files with of
any size and is easy to configure to compress other languages. To configure, we just collect a text corpus and build five corresponding dictionaries.

5.2 Contribution to Vietnamese informal text normalization

There are several approaches proposed to normalize for Vietnamese formal text. However, none of them proposed and applied to informal text. In this research, we proposed a method to fill the gap. This method was based on structure of Vietnamese morphosyllable, Vietnamese typing methods, syllable rules and trigrams dictionary to normalize for error morphosyllables, to identify error morphosyllables, we use a Vietnamese dictionary of standard morphosyllable. In this research, we also proposed a method to improve Dice coefficient in [Dice 1945]. Our proposed method achieves a better result in the term of precision when compared with the origin method. Table 22 shows our contributions, methods and publications.

<table>
<thead>
<tr>
<th>Contributions</th>
<th>Proposed method</th>
<th>Publication</th>
</tr>
</thead>
<tbody>
<tr>
<td>Text compression</td>
<td>Syllable based method</td>
<td>[Nguyen 2016b]</td>
</tr>
<tr>
<td></td>
<td>Tri-gram based method</td>
<td>[Nguyen 2016c]</td>
</tr>
<tr>
<td></td>
<td>N-gram based method</td>
<td>[Nguyen 2016a]</td>
</tr>
<tr>
<td>Vietnamese informal text normalization</td>
<td>Based on Vietnamese morphosyllable structure, syllable rules, tri-grams dictionary</td>
<td>[Nguyen 2015b]</td>
</tr>
<tr>
<td>NER for Vietnamese informal text</td>
<td>First, it uses Vietnamese informal text normalization to normalize input data first, then using SVMs model with six different types of features, they identify and classify named entities</td>
<td>[Nguyen 2015a],  [Nguyen 2016d]</td>
</tr>
</tbody>
</table>

Table 22: Contributions and publications

5.3 Contributions to NER in Vietnamese informal text

In this field, we present the first attempt to use NER for Vietnamese informal text focused on Twitter. To enhance the precision of the results, we combine the previous results with Vietnamese informal text normalization. Our contributions in this field can be briefly described as follows.

1. Integrate the normalization of Vietnamese informal text into NER model to normalize the input data first.

2. Propose a learning model for NER in Vietnamese informal text, focused on Vietnamese tweets, based on SVM models with six different types of features.
3. Build a training set of more than 40,000 named entities and a testing set of 2,446 named entities to evaluate the NER system of Vietnamese informal text, focused on Vietnamese tweets.

6 List of Student’s Publication

In this dissertation, we proposed methods to deal with the tasks in the thesis objective and scope. The result of these methods has been published in high-quality international conferences and journals. The publications are listed as follows.


5. Vu H Nguyen, Hien T Nguyen, Hieu D Duong and Vaclav Snasel. N-gram-based text compression. Computational Intelligence and Neuroscience, ISSN 1687-5273, 2016. It was indexed in Scopus and WoS.

References


