Using Large N-gram for Vietnamese Spell Checking

Article *in* Advances in Intelligent Systems and Computing · January 2015

DOI: 10.1007/978-3-319-11680-8_49

4 authors, including:

Huong Thixuan Nguyen
Haiphong Private University
4 PUBLICATIONS **16** CITATIONS
SEE PROFILE

Thai Dang
Japan Advanced Institute of Science and Technology
9 PUBLICATIONS **39** CITATIONS
SEE PROFILE

Cuong Anh Le
Ton Duc Thang University
46 PUBLICATIONS **253** CITATIONS
SEE PROFILE

Using Large N-gram for Vietnamese Spell Checking

Nguyen Thi Xuan Huong[1,2], Tran-Thai Dang[1], The-Tung Nguyen[1], and Anh-Cuong Le[1]

[1] University of Engineering and Technology
Vietnam National University, Hanoi
144 Xuanthuy, Caugiay, Hanoi, Vietnam
[2] Haiphong Private University
36 Danlap, Duhangkenh, Lechan, Haiphong, Vietnam
{cuongla@vnu.edu.vn; thaidangtran12@gmail.com;
tungnt_55@vnu.edu.vn; huong_ntxh@hpu.edu.vn}

Abstract Spell checking is a process including detecting, correcting or providing spelling suggestions for misspelled words. In this paper, we present our spell checking system relied on the context and our experimental results when doing for Vietnamese. This system uses N-gram model with large corpus. N-grams is compressed to save the memory. Furthermore, we take the contexts in both sides of syllables to improve the system's performance. Our system got high accuracy approximate 94% F-score on the Vietnamese text.

1 Introduction

In the regulatory documents, spelling is very important because the misspelled words can make some misunderstandings. Hence, a spelling checking system is necessary to make these documents are correct. This is one of pre-processing steps for Natural Language Processing (NLP) problems.

A common spelling system usually includes two main components that are detection and correction. In several cases, the system provides some correction suggestions for the users to choose. The spelling errors are considered in previous researches includes: non-word spelling error and real-word spelling error.

- The non-word errors are strings which do not exist in dictionary. For example, the following sentence in Vietnamese: "Hôm nay tôi **ddi** học". The string "ddi" is a misspelled word, and it does not exist in Vietnamese dictionary (its corresponding correct word is "đi").
- The real word errors occurs in the document when the words are mistakenly used. For example, the following sentence in Vietnamese: "Quyển **xách** này rất hay", the word "xách" (carry) is incorrectly used while the corresponding correct one is "sách" (book).

In general, the real-word error is more difficult to detect and correct than the non-word error.

In Vietnamese, we investigate several spell errors in the documents that are instances of two type which are mentioned above:

The non-word error:

- Typing error.
  Example: “bof” -> “bò”

The real-word error:

- Tones making error.
  Example: “hõi” -> “hỏi”
- Initial consonant error.
  Example: “bức chanh” -> “bức tranh”
- End consonant error.
  Example: “bắt buột” -> “bắt buộc”.
- Region error (Vietnam has many regions with various dialect, we need to change them to the common language).
  Example: “kím” -> “kiếm”

Although several spelling correction tools are used widely for Vietnamese, there is no official publication about this problem. In this paper, we present the context-based method to check spelling with large scale of N-gram model on Vietnamese text. In our work, the major clue helping us correct spelling errors is the surrounding syllables. We take the context at the left side, right side and both sides to make more clues for choosing the correct word.

In order to choose the best syllable in confusion set (the set of candidates for correcting), we need to measure the relation between a syllable and its neighbors. In this case, N-gram model is useful for modeling these relations. This model is created based on statistical approach. Hence, the distribution and variety of vocabulary on the training data have significantly affected to the system’s performance. In order to deal with the spare data, a common problem of statistical methods, we used a large corpus for training. The large corpus are collected from many text resources with various topics to reduce number of unknown words. It also help us increase number of combinations of syllables, meaning that we are able to exploit more relations among syllables and their distribution. Instead of using linguistics information such as word segmentation, POS tags, syntactic parser, the large corpus provide us enough information to get the clues for correcting.

In theoretical term, the large corpus helps N-gram model is better, in fact, we investigate whether the more size of corpus rises, the more the performance rises. That means we consider the correlation and the influence between the size of corpus and the system’s performance. We also determine if there is a limitation of corpus size. We also implement a compression method to save the memory and reduce the running time to deal with the large corpus.

2 Related work

Many researchers have been attempting to solve the spell checking problem and applied on many languages(English, French, Japanese, Chinese) such as winnow-based method was proposed by Golding in 1999 [8]; Zhang 2000 [15] introduced approximate word matching method. Iterative transformation approach was proposed by Cucerzan and Brill, 2004 [5].

Winnow-based method was one of the first approach using contextual information (context-sensitive). Using the neighboring words to detect and fix spelling errors is the main idea of context-sensitive method.

SCInsunSpell model, a spell checking model, was proposed by Li and Wang Xiaolong Juanhua in 2002. This is one of the first attempt to apply N-gram model to Chinese spell checking. The model is a combination of N-gram (trigram) model, Bayesian estimation methods and probability weighted distribution automation. In particular, the N-gram (trigram) was used for detecting phrases. It considers the link between the current word and four words surrounding it. The current word is called misspelled if its links with its four neighbors is less than a certain threshold.

IGHAN-7 is the first Chinese spell checking evaluation project. It includes two subtasks: error detection and error correction. The project was organized based on some research works (Wu et all., 2010 [14]); Chen et al., 2011 [4]; Liu et al., 2011[11]). It was received a lot of attentions from scientists. There are thirteen systems participated in the bake-off and four of them used N-gram model. This proved that despite of being an old approach, N-gram is still a powerful model to solve the spell checking problems.

In Vietnamese, Nguyen Duc Hai and Nguyen Pham Hanh Nhi, 1999 [18] proposed a spell checking model based on sentence parsing. Five years later, Nguyen Thai Ngoc Duy [19] developed a spell checking system based on word-network. These system's performance did not meet the user's requirement. In 2012, Nguyen Huu Tien Quang [20] proposed a spell checking system using N-gram model and Viterbi algorithm. This is one of the earliest study on applying N-gram model to Vietnamese spell checking problem and its results were much better than previous work. Although, there are some researches about this problem for Vietnamese, they just have been presented in bachelor thesis, and no official publication.

Basing on previous research for this problem, we also use context based approach with n-gram model to build the spell checking system. In our empirical work, there are some improvements which extend the context in both sides of syllable and use n-gram model with the large corpus.

3 Our Approach

Similarly to the previous works on English, Chinese, Japanese we use context based approach and N-gram model for our spell checking system. The measure of relation between a syllable and its neighbors is computed and evaluated to select the most likely correct syllable. In order to improve the performance of this system, we extended the context in both sides of syllable and used the large corpus. The N-gram compression is also implemented to optimize the size of memory for storage.

As previous mention, our System includes two main components: making confusion sets, calculating probabilities then comparing to choose a the most likely candidate in the confusion set. Before checking, the data need to be processed such as normalization, sentence splitting, punctuation marks removal. Our system's architecture is illustrated in the figure 1.

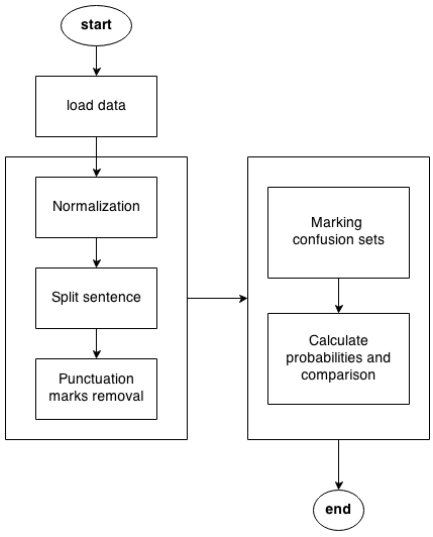

Figure 1. Our system's architecture

3.1 Pre-Processing

Pre-processing stage contains three steps:

- **Step 1:** Recognizing the special syllables such as web address, email, number... and replacing them by special notation. For example:
  The number "5" is replaced by "NUMBER".
  The name of a website "http://facebook.com" is replaced by "WEBSITE".
  Replacing the special patterns by the predefined notations help to reduce the number of unknown words.
- **Step 2:** Splitting the document into sentences because two syllables in different sentences have no relationship with the other.
- **Step 3:** Removing all punctuation marks in the sentences because they do not have relation of meaning with the words.

3.2 Extending the context in both sides

The main component of our spell checking system includes two steps:

- **Step 1:** Building the confusion set for each syllable based on edit distance and chosen Vietnamese characteristics.
- **Step 2:** Calculating the measure of relationship between a syllable and its neighbors based on N-gram model to decide whether the current syllable is incorrect or not then choose the most likely candidate to fix it.

The previous context-sensitive spelling corrections almost take the context in left side. We take the context at both sides of the syllable to improve the system's performance. Using the context in both sides help us get more clues to choose the best candidate in confusion set.

The context used in our system is the 2-radius window of syllables surrounding, which means if we denoted the current syllable as w_0 and its contexts are w_{-1}, w_{-2}, w_1, w_2. We can model the w_0's dependency on its neighbors by the conditional probability below.

$$\mathrm{P}(w_0 \mid w_{-2}\ , w_{-1}, w_1, w_2)$$

This probability above may be estimated by the following function:

$$\mathrm{P}(w_0|w_{-2},w_{-1},w_1,w_2) = \mathrm{f}(\mathrm{P}(w_0|w_{-2},w_{-1}), \mathrm{P}(w_0|w_{-1},w_1), \mathrm{P}(w_0|w_1,w_2))$$

where f is a geometric mean function.

In order to calculate this probability, we need 5-gram and 4-gram. This is not feasible to carry out because there is a large number of combinations and the data is too spare. Therefore, instead of calculating $\mathrm{P}(w_0 \mid w_{-2}, w_{-1}, w_1, w_2)$ we estimate it by three trigram probabilities: $\mathrm{P}(w_0|w_{-2}, w_{-1})$, $\mathrm{P}(w_0|w_{-1}, w_1)$, $\mathrm{P}(w_0| w_1, w_2)$. The N-gram scores *p* is the geometric mean logarithms of these three probabilities. We choose the geometric mean because the name entity such as human name or organization can weaken the link of syllable with its context.

Positive errors occur when a syllable is determined as error, but it is actually spelling-correct. To reduce number of these errors, we used heuristic coefficients "error threshold" and "difference threshold". For short we denoted them as *e_thresh* and *d_thresh*. Assume that our current syllable w_0 has N-gram score *p* and one syllable from the error set w_0' has N-gram score *p'*, w_0' is considered to be "better" than w_0 if and only if it satisfies two inequalities below:

- *p' > e_thresh*
- *p' > p + d_thresh*

e_thresh is a constant determined by using development data, it ensures that if one syllable is going to be used to fix the current syllable, its probability must be higher than a certain threshold; this help us to reduce the false positive caused by named-entity.

3.3 Large scale N-gram and Compression

Our system essentially uses N-gram model, the training data plays an important role which affects to the performance of the system. As the mention in section 3.2, we need to compute the probabilities of trigrams, thus the frequencies of bigrams and trigrams must be determined. The large corpus including many topics help us deal with the spare data problem without extra information such as word segmentation, part-of-speech, syntactic parser. The collected data from variety of resources and topics will reduce the number of unknown words. It also includes many possible combinations of syllables which help us exploit more information of relations among the syllables and

their frequencies. We also use smoothing methods to compute probabilities of trigrams effectively in case of unknown words.

Using large corpus arises a problem that is how to compress the n-gram to save the memory when we load data. Moreover, the compression does not change the accuracy of N-gram model. In our system, we are going to implement a method to encode the n-grams by numbers without the loss of data. In encoding process, we collected the Vietnamese syllable dictionary including approximate 6800 syllables. Each syllable is represented by a number that start at 0. Firstly, we count the frequency of n-gram from the raw corpus and remove the n-grams which appear less than 5 times. Secondly, we start encoding n-gram. 1-gram (unigram) is encoded similarly to syllables because it contains one syllable. There are approximate 6800 syllables, so we need from 0 to 6800 to represent, each number need two bytes to store. For 2-gram (bigram), each syllable need 2 bytes for storing, so each bigram can be stored by 4 bytes (an integer). We also used *bit shifts* operator and *or* operator to encode bigram by an integer as the following:

int x = syllables[0]

x = x << 16

x = x | syllables[1]

Similarly, we need 6 bytes to encode trigram (3-gram).

In this paper, we also investigate the *"saturation"* of the training data to answer the question whether the more data rises, the more performance is improved. We will consider the influence of size of corpus to the performance and identify if there exist a limitation, which the data size reaches to the accuracy of the system does not increase.

4 Experiments

4.1 Experimental data

Training data In order to build N-gram model, we collected the data from various sources such as *Wikipedia.org, dantri.com.vn, vnExpress.net*. The data includes many topics such as mathematics, physics, science, literature, philosophy, history, economy, sport, law, news, entertainment... The size of our corpus is about 2GB. We counted frequencies of unigram, bigram, trigram then remove the n-gram which has frequency less than 5.

Testing data We created two test sets for evaluating our system. Firstly, we collected text from the Internet. In the first set, we manually check to ensure that there is no spelling error in it. After that we generated artificial spelling errors in the test set and marked these errors to evaluate the efficiency of our system. In the second set, we also found and marked spell errors. The first test set was used in experiment 1 and 2. Experiment 3 used the second set. The first test set contains 2500 sentences and second one contains 632 sentences.

4.2 Experimental results

Before evaluating the performance of our spell checking system, we applied the n-gram compression method presented in section 3.3 to compress the training data after counting the frequency. The compression results are illustrated in table 1.

Table 1. N-gram compression results

	# of elements	data size before encoding	data size after encoding
unigram	6776	77.9 KB	13.55 KB
bigram	1208943	15.6 MB	4.6 MB
trigram	4886364	84 MB	28 MB

From table 1, we see that by using the compression, the size of memory reduces significantly. After using encoding, the sizes of bigram, trigram data are one-third of the original one.

In order to evaluate the spelling checking system, we used five metrics:
- Detection precision (DP).
- Detection recall (DR).
- Correction precision (CP).
- Detection F-score (DF).
- False positive rate (FPR).

The formula of each each metrics as the following:

- DP = $\frac{\#\ of\ correct\ detections}{\#\ of\ errors\ detected}$
- DR = $\frac{\#\ of\ correct\ detections}{\#\ of\ actual\ errors}$
- CP = $\frac{\#\ of\ correct\ fix}{\#\ of\ errors\ detected}$
- DF = $\frac{2 * DP * DR}{DP + DR}$
- FPR = $\frac{\#\ of\ incorrect\ detections}{\#\ of\ correct\ syllables}$

Experiment 1: We investigated the influence of the corpus size to the system's performance in order to answer the question whether there exits a limitation of corpus, when the size of data reach to this limitation, the performance does not rise or not. We split the corpus into fragments, each one is about 100 MB. A mini-corpus was created by joining these fragments together. For example: a 200 MB mini-corpus is the combination of two 100 MB fragments. Then each mini-corpus was created their corresponding n-gram dictionary (counting the frequencies of n-grams) and used in our system. The figure 2 illustrates the detection F-score of our system with each mini-corpus.

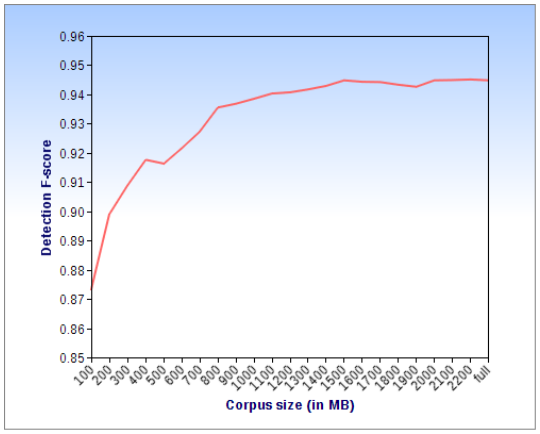

Figure 2. Influence of corpus size to the system's performance

From figure 2, we see that: when the size of corpus is greater than 1.5 MB the system's accuracy does not increase. In our work, we can use approximate 1.5 GB of corpus to train N-gram model. This experimental result also illustrates that there exist a threshold of corpus size and when the size of corpus reach to this threshold, the training data does not affect to the system's performance.

Experiment 2: We evaluated the influence of the context to the system's accuracy. The table 2 show the evaluation results of each context.

Table 2. Influence of context to the system's performance

Context	DP	DR	CP	DF	FPR
w_{-2}, w_{-1}	89.42%	52.22%	97.31%	65.93%	0.12%
w_{-1}, w_1	94.04%	91.53%	98.26%	92.76%	0.11%
w_1, w_2	93.83%	73.63%	96.79%	82.51%	0.09%
w_{-2}, w_{-1}, w_1, w_2	94.68%	94.26%	99.32%	94.46%	0.1%

From the table 2, we can easily observe that two-sided contexts give us much better results than the one-sided contexts, and the context w_{-2}, w_{-1}, w_1, w_2 gives the best result. Furthermore, the last column in table 2 shows that the false positive rate is very low in our system, meaning that our method has reduced these errors effectively.

Experiment 3: We compared our system with an other spell checking system for Vietnamese: copcon 5.0.3 beta [3]. We will compare the accuracy of their detection and their correction on the second test set. The results are shown in table 3.

[3] link: http://chinhta.vn

Table 3. Comparison between our system and Copcon

	DP	DR	CP	DF	FPR
Our system	92.62%	91.12%	95.45%	91.86%	0.2%
Copcon 5.0.3 beta	80.8%	77.6%	87.5%	79.2%	0%

From the table 3, we see that the accuracy of detecting and correcting of our system is greater than the copcon's accuracy.

5 Conclusion

This paper presents the context-sensitive spell checking system for Vietnamese based on N-gram model with large scale. By extending the context at both sides of syllable, we improved the accuracy of spell checking system, and got higher performance than copcon, which is a spelling tool for Vietnamese. Our system just use N-gram model instead of extra linguistic information. Our empirical work also shows the influence of corpus size to the system's performance. In order to deal with the large corpus, a method to compress n-grams is implemented. Moreover, statistical approach is able to be applied to novel domains such as scientific papers, and spoken language in the forums...

Acknowledgment

This paper is supported by the projects QGTĐ.12.21 and QG.14.04 funded by Vietnam National University, Hanoi.

References

1. C.Blair. 1960. *"A program for correcting spelling errors"* Information and Control, 60-70.
2. A.Carlson, J.Rosen, and D.Roth. 2001. *"Scaling up context-sensitive text correction"*. In Proceedings of the 13th Innovative Applications of Artificial Intelligence Conference, 45-50
3. A. Carlson and I. Fette. 2007. *"Memory-based Context-Sensitive Spelling Correction at Web Scale"*. In Proceedings of the 6th International Conference on Machine Learning and Applications, 166-171.
4. Y.Z. Chen, S.H. Wu, P.C. Yang, T. Ku, and G.D. Chen. 2001. *"Improve the detection of improperly used Chinese characters in student's essays with error model"* Int.J.Cont.Engineering Education and Life-Long Learning, 103-116.
5. S.Cucerzan and E.Brill. 2004. *"Spelling correction as an iterative process that exploits the collective knowledge of web users"*. In Proceedings of EMNLP, 293-300.
6. F. Damerau. 1964. *"A technique for computer detection and correction of spelling errors."* Communications of the ACM 7, 171-176.
7. S. Deorowicz and M.G. Ciura. 2005. *"Correcting Spelling Errors by Modelling Their Causes"* International Journal of Applied Mathematics and Computer Science, 15(2), 275-285
8. A. Golding and D. Roth. 1999. *"A winnow-based approach to context-sensitive spelling correction"* Machine Learning, 34(1-3), 107-130.

9. A. Islam and D. Inkpen. 2009. *"Real-word spelling correction using googleweb 1t 3-grams"* Proceedings of Empirical Methods in Natural Language Processing (EMNLP-2009), 1241-1249.
10. W. Liu, B. Allison, and L. Guthrie. 2008. *"Professor or screaming beast? Detecting words misuse in Chinese"* The 6th edition of the Language Resources and Evaluation Conference.
11. C.L. Liu, M.H. Lai, K.W. Tien, Y.H. Chuang, S.H. Wu, and C.Y. Lee. 2011. *"Visually and phonologically similar characters in incorrect Chinese words: Analyses, identification, and applications"* ACM Transactions on Asian Language Information Processing, 1-39.
12. S. Verberne. 2002. *"Context-sensitive spell checking based on word trigram probabilities"*, Master thesis, University of Nijmegen.
13. C. Whitelaw, B. Hutchinson, G.Y. Chung, and G. Ellis. 2009. *"Using the Web for Language Independent Spellchecking and Autocorrection"* Proceedings Of Conference On Empirical Methods In Natural Language Processing (EMNLP-2009), 890-899.
14. S.H. Wu, Y.Z. Chen, P.C. Yang, T. Ku, and C.L. Liu. 2010. *"Reducing the False Alarm Rate of Chinese Character Error Detection and Correction"* Proceedings of CIPS-SIGHAN Joint Conference on Chinese Language Processing (CLP 2010), 54-61.
15. L. Zhang, M. Zhou, C.N. Huang, and H.H. Pan. 2000. *"Automatic detecting/correcting errors in Chinese text by an approximate word-matching algorithm"* Proceedings of the 38th Annual Meeting on Association for Computational Linguistics, 248-254.
16. J. Li, X. Wang, *"Combine trigram and Automatic Weight Distribution in Chinese Spelling ErrorCorrection"*, Journal of Computer Science and Technology archive Volume 17 Issue 6, November 2002, Pages 915-923
17. R. Mitton. 2008. *"Ordering the Suggestions of a Spellchecker Without Using Context"* Natural Language Engineering, 15(2). Pages 173-192.
18. Nguyen Duc Hai and Nguyen Pham Hanh Nhi, *"Syntactic parser in Vietnamese sentences and its application in Spell Checking"* in Vietnamese, bachelor thesis, in University of Science Ho Chi Minh city, 1999.
19. Nguyen Thai Ngoc Duy, Dinh Dien, *" An approach in Vietnamese spell checking"* in Vietnamese, bachelor thesis in University of Science Ho Chi Minh city, 2004.
20. Nguyen Huu Tien Quang, *"Language model and word segmentation in Vietnamese Spell Checking"* in Vietnamese, bachelor thesis in University of Engineering and Technology, Hanoi National University, 2012