VNU Jounal of science, Mathermatics - Physics 27 (2011) 134-146

Building and Evaluating Vietnamese Language Models

Cao Van Viet, Do Ngoc Quynh, Le Anh Cuong*

University of Engineering and Technology, Vietnam National University, Ha Noi (VNU)
E3-144, Xuân Thuy, Cau Giay, Ha Noi

Received 05 September 2011, received in revised from 28 October 2011

Abstract: A language model assigns a probability to a sequence of words. It is useful for many Natural Language Processing (NLP) tasks such as machine translation, spelling, speech recognition, optical character recognition, parsing, and information retrieval. For Vietnamese, although several studies have used language models in some NLP systems, there is no independent study of language modeling for Vietnamese on both experimental and theoretical aspects. In this paper we will experimently investigate various Language Models (LMs) for Vietnamese, which are based on different smoothing techniques, including Laplace, Witten-Bell, Good-Turing, Interpolation Kneser-Ney, and Back-off Kneser-Ney. These models will be experimental evaluated through a large corpus of texts. For evaluating these language models through an application we will build a statistical machine translation system translating from English to Vietnamese. In the experiment we use about 255 Mb of texts for building language models, and use more than 60,000 parallel sentence pairs of English-Vietnamese for building the machine translation system. b
Key words: Vietnamese Language Models; N-gram; Smoothing techniques in language models; Language models and statistical machine translation

1. Introduction

A Language Model (LM) is a probability distribution over word senquences. It allows us to estimate the probability of a sequence of *m* elements in a language, denoted by $P(w_1, .., w_m)$, where each w_i is usually a word in the language. It means that from a LM we can predict the ability of appearing a sequence of words. By using the Bayesian inference, we easily obtain the following formula:

$$P(w_1 w_2 \ldots w_m) = P(w_1) * P(w_2|w_1) * P(w_3|w_1 w_2) * \ldots * P(w_m|w_1 \ldots w_{m-1}) \quad (1)$$

* Corresponding author: Tel:. (+84) 912 151 220

According to the formula (1), the probability of a sequence of words can be computed through the conditional probability of appearing a word given previous words (note that $P(w_1)=P(w_1|start)$ where *start* is the symbol standing for the beginning of a sentence). In practice, based on the Markov Assumption we usually compute the probability of a word using at most *N* previous words (*N* is usually equal *0,1, 2, or 3*.

From that interpretation, we can use *N-gram* model instead of Language Model (note that *N* is counted including the target word). Each a sequence of *N* words is considered as an *N-gram*. Some popular *N-gram* types are illustrated through the following example.

Suppose that we need to compute the probability $p = P(sách \mid tôi\ đã\ từng\ đọc\ quyển)$:

- 1-gram (unigram) computes the probability of a word without considering any previous word. It means that: $p = P(sách)$

- 2-gram (bigram) computes the probability of a word, conditioned on its one previous word. It means that: $p = P(sách|đọc)$

- 3-gram (trigram) computes the probability of a word, conditioned on its two previous words. It means that: $p = P(sách|đọc\ quyển)$

Many NLP problems using language models can be formulated in the framework of the Noise Channel Model. In this view, suppose that we are having an information quantity, and transfer it through a noise channel. Then, because of the noise environment of the channel, when receiving the information again we may lost some information. The task here is how to recover the original information. For example, in speech recognition problem we receive a sentence which has been transferred through a speech source. In this case, because we may lost some information depending on the speaker, we usually image several words for each sound (of a word). Consequently we may obtain many potential sentences. Then, using a statistical language model we will choose the sentence which has the highest probability.

Therefore, LMs can be applied in such problems which use them in the framework of noise channel model, such as speech recognition [11, 26], optical character recognition [1, 22], spelling [9]. Some other applications use LMs as criteria to represent knowledge resources. For example, in information retrieval, some studies used language model for representing questions and documents, as in [12, 25]. Moreover, the techniques used for estimating *N-gram* and the *N-gram* itself are widely used in many other NLP problems such as part-of-speech tagging, syntactic parsing, text summarization, collocation extraction, etc. In addition, one of the most important application of LMs is statistical machine translation (SMT). It is used for translating fluently. It is also useful for lexical disambiguation.

As well known, Maximum Likelihood Estimation (MLE) is the popular method for estimating *N-gram* probabilities. However, it usually faces to the zero division problem. Therefore, some smoothing techniques for LMs are developed to resolve this problem. There are three common strategies of

smoothing techniques, including Discounting, Back-off, and Interpolation. The popular methods of discounting include Laplace, Good-Turing, and Witten-Bell. The effective methods for interpolation and back-off are Interpolation Kneser-Ney and Back-off Kneser-Ney presented in [15]. Note that these techniques have been being applied widely for building LMs and used for many NLP systems.

Some recent studies have focused on the complex structures for building new LMs, for example a syntax-based LM is used for speech recognition [8], and for machine translation [5]. In other studies, they used a very large number of texts (usually use web-based) for building LMs to improve the task of word sense disambiguation, statistical machine translation [3, 2].

For Vietnamese, there are some studies have tried to apply *N-gram* for some ambiguity NLP problems, for example the authors in [24] used *N-gram* for word segmentation, the authors in [19] used *N-gram* for speech recognition. However, these studies have not worked on evaluating and comparing different LMs. We cannot intuitively separate unigram, bigram, trigram, as well as cannot image how a word depends on previous words for Vietnamese. Therefore, in this paper we focus on experimently investigating these aspects of LMs for Vietnamese, specially on both syllabi and words. In addition, to apply LMs for Vietnamese text processing, we will investigate different LMs when applying them for an English-Vietnamese SMT system to find out the most appropriate LM for this application.

The rest of paper is organized as follows: section 2 presents different N-gram models based on different smoothing techniques/methods; section 3 presents the evaluation of LMs using Perplexity measurement; section 4 presents SMT and the role of Language Models in SMT; section 5 presents our experiments; and section 6 is the conclusion.

2. Smoothing Techniques

To compute the probability $P(w_i|w_{i-n+1}...w_{i-1})$ we usually use a collection of texts which are called the *training data*. Using MLE we have:

$$P(w_i|w_{i-n+1}...w_{i-1}) = \textbf{Error!} \quad (2)$$

where $C(w_{i-n+1}...w_{i-1}w_i)$ and $C(w_{i-n+1}...w_{i-1})$ are the frequencies (or counts) of appearing $w_{i-n+1}...w_{i-1}w_i$ and $w_{i-n+1}...w_{i-1}$ in the training data, respectively. Formula (2) gives a value for $P(w_i|w_{i-n+1}...w_{i-1})$, we call it the "raw probability".

When the training data is sparse, there are many N-grams which do not appear in the training data or appear with a few times. In this situation the "raw probability" will be not correct. For example it is easy to meet a sentence which is correct on both grammar and semantic but its probability is equal to zero because it contains an N-gram which does not appear in the training data. To solve the zero division problem we use some smoothing techniques, each of them corresponds to a LM (see [13, 18] for more detail reference). They are categorized as follows.

Discounting: discounting (lowering) some non-zero counts in order to get the probability mass that will be assigned to the zero counts.

Back-off **:** we only "back-off" to a lower order N-gram if we have zero evidence for a higher-order N-gram.

Interpolation**:** compute the probabilities of an N-gram based on lower order N-grams. Note that we always mix the probability estimates from all the N-gram estimators.

3. Discounting methods

We present here three popular discounting methods: Laplace (one popular method of them is the Add-one method), Witten-Bell, and Good-Turing.

Add-one method:

This method adds 1 to each count of N-grams. Suppose that there are *V* words in the vocabulary, we also need to adjust the denominator to take into account the extra *V* observation. Then, the probability is estimated as:

$$P(w_i|w_{i-n+1}...w_{i-1}) = \textbf{Error!}$$

In generalization we can use the following formula:

$$P(w_1 w_2...w_n) = \textbf{Error!}$$

The value of λ is chosen in the interval [0, 1], with some specific values:

- $\lambda = 0$: without smoothing (MLE)
- $\lambda = 1$: Add-one method
- $\lambda =$ **Error!**: Jeffreys – Perks method

Witten-Bell method:

The Witten-Bell method [27] models the probability of a previously unseen event by estimating the probability of seeing such a new event at each point as one proceeds through the training data. In unigram, denote *T* as the number of different unigram, and denote *M* as the total number of all unigrams. Then, the probability of a new unigram is estimated by: **Error!**

Let *V* is the vocabulary' size and *Z* is the number of unigrams which doesn't appear in the training data, then: $Z = V - T$. Then the probability of a new unigram (i.e. its count is equal 0) is estimated by:

$$p^* = \textbf{Error!}$$

And the probability of an unigram which is not the zero-count is estimated by:

$$P(w) = \textbf{Error!}$$

where *c(w)* is the count of *w*.

When considering the N-grams with N>1, if we replace M by $C(w_{i-n+1}...w_{i-1})$ then the probability of $w_{i-n+1}...w_{i-1}w_i$ (here $C(w_{i-n+1}...w_{i-1}w_i) = 0$) is estimated by:

$$P(w_{,i}|w_{,i-n+1}...w_{,i-1}) = \textbf{Error!}$$

In the case $C(w_{,i-n+1}...w_{,i-1}w_{,i}) > 0$, we have:

$$P(w_{,i}|w_{,i-n+1}...w_{,i-1}) = \textbf{Error!}$$

Good – Turing method:

Denote N_c as the number of N-grams which appear c times. Good-Turing method will replace the count c by c^* by the formula: $c^* = (c+1) * \textbf{Error!}$

Then, the probability of an N-gram with its count c is computed by:

$P(w) = \textbf{Error!}$ where $N = \textbf{Error!}N\textbf{Error!}c = \textbf{Error!}N\textbf{Error!}c^* = \textbf{Error!}\ N\textbf{Error!}(c+1)$

In the practice, we do not replace all c by c^*. We usually choose a threshold k, and only replace c by c^* if c is lower than k.

3.1 Back-off methods

In the discounting methods such as Add-one or Witten-Bell, if the phrase $w_{,i-n+1}...w_{,i-1}w_{,i}$ does not appear in the training data, and the phrase $w_{,i-n+1}...w_{,i-1}$ also does not appear, then the probability of $w_{,i-n+1}...w_{,i-1}w_{,i}$ is still equal zero. The back-off method in [14] avoids this drawback by estimating the probabilities of a new N-gram based on lower order N-grams, as the following formula.

$$P_{,B}(w_{,i}|w_{,i-n+1}...w_{,i-1}) = \begin{cases} P(w_i|w_{i-n+1}...w_{i-1}) & if\ C(w_{i-n+1}...w_{i-1}w_i) > 0 \\ \alpha * P_B((w_i|w_{i-n+2}...w_{i-1}) & if\ C(w_{i-n+1}...w_{i-1}w_i) = 0 \end{cases}$$

For bigram, we have:

$$P_{,B}(w_{,i}|w_{,i-1}) = \begin{cases} P(w_i|w_{i-1}) & if\ C(w_{i-1}w_i) > 0 \\ \alpha * P(w_i) & if\ C(w_{i-1}w_i) = 0 \end{cases}$$

Similarly for trigram:

$$P_{,B}(w_{,i}|w_{,i-2}w_{,i-1}) = \begin{cases} P(w_i|w_{i-2}w_{i-1}) & if\ C(w_{i-2}w_{i-1}w_i) > 0 \\ \alpha * P(w_i|w_{i-1}) & if\ C(w_{i-2}w_{i-1}w_i) = 0\ and\ C(w_{i-1}w_i) > 0 \\ \alpha * P(w_i) & if\ C(w_{i-2}w_{i-1}w_i) = 0\ and\ C(w_{i-1}w_i) = 0 \end{cases}$$

Here, we can choose constant values for $\alpha_{,1}$ and $\alpha_{,2}$. In another way, we can design $\alpha_{,1}$ and $\alpha_{,2}$ as functions of N-gram as: $\alpha_{,1} = \alpha_{,1}(w_{,i-1}w_{,i})$ and $\alpha_{,2} = \alpha_{,2}(w_{,i-1}w_{,i})$.

However it is easy to see that in these above formulas the sum of all probabilities (of all N-grams) is greater than 1. To solve this problem, we usually combine discounting techniques into these formulas. Therefore, in practice, we have the following formulas for the back-off method:

$$P(w_{,i}|w_{,i-2}w_{,i-1}) = \begin{cases} P'(w_i|w_{i-2}w_{i-1}) & if\ C(w_{i-2}w_{i-1}w_i) > 0 \\ \alpha * P'(w_i|w_{i-1}) & if\ C(w_{i-2}w_{i-1}w_i) = 0\ and\ C(w_{i-1}w_i) > 0 \\ \alpha * P'(w_i) & if\ C(w_{i-2}w_{i-1}w_i) = 0\ and\ C(w_{i-1}w_i) = 0 \end{cases}$$

where P' is the probability of the N-gram when using an discounting method.

3.2 Interpolation methods

This approach has the same principle with the back-off approach that uses lower order N-grams to compute the higher order N-grams. However, it is different from back-off methods in the point of view: it always use lower order N-grams without considering that the count of the target N-gram is equal zero or not. We have the formula as follows.

$$P_I(w_i|w_{i-n+1}...w_{i-1}) = \lambda P(w_i|w_{i-n+1}...w_{i-1}) + (1-\lambda)P_I(w_i|w_{i-n+2}...w_{i-1})$$

Apply for bigram and trigram we have:

$$P_I(w_i|w_{i-1}) = \lambda P(w_i|w_{i-1}) + (1-\lambda)P(w_i)$$

$$P(w_i|w_{i-2}w_{i-1}) = \lambda_1 P(w_i|w_{i-2}w_{i-1}) + \lambda_2 P(w_i|w_{i-1}) + \lambda_3 P(w_i) \text{ với } \Sigma_i \lambda_i = 1$$

In the above formulas, the weights can be estimated using the Expectation Maximization (EM) algorithm or by the Powell method presented in (Chen and Goodman 1996).

3.3 Kneser-Ney's smoothing

The Kneser-Ney algorithms [15] have been developed based on the back-off and interpolation approaches. Note that Kneser-Ney algorithms do not use discounting techniques. They are shown as the following (see more detail in [6]).

The formula for Back-off Kneser-Ney is presented as follows.

$$P_{BKN}(w_i|w_{i-n+1}..w_{i-1}) = \begin{cases} \frac{C(w_{i-n+1}...w_i)-D}{C(w_{i-n+1}...w_{i-1})} & if\ C(w_{i-n+1}...w_i) > 0 \\ \alpha(w_{i-n+1}...w_{i-1})P_{BKN}(w_i|w_{i-n+2}...w_{i-1}) & if\ C(w_{i-n+1}...w_i) = 0 \end{cases}$$

where:

$P_{BKN}(w_i) =$ **Error!** where *N(vw)* is the number of different words *v* appearing at right ahead of *w* in the training data.

$$\alpha(w_{i-n+1}..w_{i-1}) = \textbf{Error!}$$

The formula for Interpolation Kneser-Ney is presented as follows.

$P_{IKN}(w_i|w_{i-n+1}..w_{i-1}) =$ **Error!** + λ(w**Error!**..w**Error!**)P**Error!**(w**Error!**|w**Error!**..w**Error!**)

where:

$\lambda(w_{i-n+1}..w_{i-1}) =$ **Error!** where N(w**Error!**..w**Error!**v) is the number of different word *v* appearing right after the phrase $w_{i-n+1}..w_i$ in the training data.

$P_{IKN}(w_i) =$ **Error!** + λ **Error!** where N(vw) is the number of different words *v* appearing at right ahead of *w* in the training data.

$$\lambda = \textbf{Error!}$$

In the both back-off and interpolation models, D is chosen as: $D = \frac{N_1}{N_1+2N_2}$ where N_1 and N_2 are the number of N-grams which appear 1 and 2 times respectively.

4. Evaluating language model by Perplexity

There are usually two approaches for evaluating LMs. The first approach depends on only the LM itself, using a test corpus, called intrinsic evaluation. The second approach is based on the application of the LM, in which the best model is the model which brings the best result for the application, it is called extrinsic evaluation.

This section presents the first approach based on Perplexity measurement. The next section will present the second approach when applying for a SMT system.

Perplexity of a probability distribution p is defined as:

$$2^{H(p)} = 2^{-\sum_x p(x)\log_2 p(x)}$$

where $H(p)$ is the entropy of p.

Suppose that the test corpus is considered as a sequence of words, denoted by $W = w_1 \ldots w_N$, then according to [13] we have the approximation of $H(W)$ as follows.

$$H(W) = -\frac{1}{N} log P(w_1 \ldots w_N)$$

A LM is a probability distribution over entire sentences. The Perplexity of the language model P on W is computed by:

$$Perplexity(W) = \sqrt[N]{\frac{1}{P(w_1 \ldots w_N)}}$$

Note that given two probabilistic models, the better model is the one that has a tighter fit to the test data, or predicts the details of the test data better. Here, it means that the better model gives higher probability (i.e. lower Perplexity) to the test data.

5. Evaluating language models through a SMT system

The problem of Machine Translation (MT) is how to automatically translate texts from one language to another language. MT has a long history and there are many studies focusing on this problem with various discovered techniques. The approaches in MT include direct, transfer (or rule-based), example-based, and recently statistical MT (SMT) has been becoming the most effective approach.

SMT was firstly mentioned in the paper [4]. The beginning systems are word-based SMT. The next development is phrase-based SMT [16], which has shown a very good quality in comparison with the conventional approaches. SMT has the advantage that it doesn't depend on linguistic aspects and uses only a parallel corpus for training the system (note that recent studies concentrates on integrating linguistic knowledge into SMT). In the following we will investigate the basic SMT system and the role of LMs to it.

Suppose that we want to translate an English sentence (denoted by E) to Vietnamese. The SMT approach assumes that we are having all Vietnamese sentences, and V* is the translation sentence in Vietnamese if it satisfies:

$$V^* = \text{argmax}_V P(V|E)$$

(Note that in practice, we will determine *V** among a finite set of sentences which can be potential translation of *E*).

According to Bayesian inference we have:

$$V^* = \text{argmax}_V \frac{P(E|V)*P(V)}{P(E)}$$

Because *P(E)* is fixed for all *V* so we have:

$$V^* = \text{argmax}_V P(E|V) * P(V)$$

We can see that the problem now is how to estimate *P(E|V)*P(V),* where *P(E|V)* represents for the translation between *V* and *E*, and *P(V)* (which is computed by a LM) represents for how the translation is natural, smooth in the target language. Another effect of *P(V)* is that it will remove some wrong translation elements which may be selected in the process of determining *P(E|V)*.

Therefore, LMs play an important role for SMT. In the experiment we will investigate different LMs in a English to Vietnamese SMT system. We will use BLEU score to evaluate which LM is most effective for this machine translation system.

6. Experiment

On the work of conducting necessary experiments, we firstly collect raw data from Internet, and then standardize the texts. We also carry out the task of word segmentation for building LMs at word level. Different LMs will be built based on different smoothing methods: Laplace, Witten-Bell, Good-Turing, Back-off Kneser-Ney, and Interpolation Kneser-Ney. For this work we use the open toolkit SRILM [23].

To build an English-Vietnamese machine translation system we use the open toolkit MOSES [17]. Note that the LMs obtained from the experiment above will be applied in this SMT system.

Data preparation

The data used in LM construction are collected from the news sites (dantri.com.vn, vnexpress.net, vietnamnet.vn). These HTML pages are then processed through some tools for tokenizing and removing noise texts. Finally we acquire a corpus of about 255 Mb (including nearly 47 millions of syllabi). We also use a word segmentation tool on this data and obtain about 42 millions of words. Table 1 shows the statistics of unigrams, bigrams, and trigrams on both syllabi and words. Note that this data is used for building language models, in which we use 210 Mb for training and 45 Mb for testing.

Kind of unit	Number Of units	Number of different Unigram	Number of different Bigram	Number of different Trigram
Syllabus	46,657,168	6,898	1,694,897	11,791,572
Word	41,469,980	35,884	3,573,493	16,169,361

Table 1: Statistics of unigrams, bigrams, and trigrams

To prepare data for SMT, we use about 60 thousands of parallel sentence pairs (from a national project in 2008 aiming to construct labeled corpora for natural language processing). From this corpus, 55 thousands pairs are used for training, and 5 thousands pairs for testing.

Intrinsic evaluation of N-gram models

The smoothing methods used for building LMs are Laplace (includes Jeffreys – Perks and add-one), Witten-Bell, Good-Turing, Knerser-Ney interpolation, and Knerse-Ney back-off. Table 2 shows the Perplexity for these models on the test data at syllabus level. Table 2 shows the similar experiment but at word level.

It is worth to repeat that Perplexity relates to the probability of appearing a word given some previous words. For example in the Table 2, the Good-Turing model gives Perplexity a value of 64.046 on 3-gram means that there are about 64 values (or options) for a word if given the two previous words. Therefore, a LM is considered better than the other if it has lower Perplexity on the test data.

N-gram	Perplexity values					
	Add-0.5 Jeffreys - Perks	Add-one	Witten Bell	Good Turing	Interpolation Kneser-Ney	Kneser-Ney Back-off
1-gram	658.177	658.168	658.291	658.188	658.23	658.23
2-gram	130.045	142.249	116.067	115.422	114.359	114.631

3-gram	227.592	325.746	64.277	64.046	60.876	61.591

Table 2: Perplexity for syllabus

N-gram	Perplexity values					
	Add-0.5 Jeffreys - Perks	Add-one	Witten Bell	Good Turing	Nội suy Kneser-Ney	Truy hồi Kneser-Ney
1-gram	924,571	924,543	924,975	924,639	924,679	924,679
2-gram	348,715	443,225	188,961	187,51	183,475	183,853
3-gram	1035,8	1573,69	125,786	123,856	115,884	117,799

Table 3: Perplexity for words

From Table 2 and Table 3 we can infer the two important remarks as follows.

- Among discounting methods, Good-Turing gives best results (i.e. lowest perplexity) on all unigram, bigram, and trigram. In there, Good-Turing and Witten-Bell have similar results. We can also see that the higher N (of N-gram) is the better Good-Turing and Witten-Bell are, in comparison with Laplace methods. In practice, people simply use Laplace methods, and in such cases they must be noted that Jeffreys-Perks method (i.e. the Add-half method) is much better than Add-one method.

- Interpolation Kneser-Ney is better than Back-off Kneser-Ney and both of them give better results (i.e. lower perplexity) in comparison with Good-Turing and Witten-Bell. We can also see that the quality distance between Kneser-Ney methods and Good-Turing/Witten-Bell will be bigger if we increase N (of N-gram).

Moreover, we can see that the best Perplexity scores for 3-gram are about 61 (computing on syllabi) and 116 (computing on words). These values are still high, therefore in the NLP problems which use Vietnamese language model, if we can use *N-gram* with the higher order then we can obtain better results.

Extrinsic valuation of N-gram models using SMT

In this work we will use the LMs obtained in section 5.2 and integrate them into a SMT system (using MOSES). Because SMT systems treat words as the basic elements so in this work we just use the word-based LMs. Table 4 gives us the BLEU scores [20] of the SMT system on different LMs.

N-gram	BLEU scores					
	Add-0.5 Jeffreys - Perks	Add-One	Witten Bell	Good Turing	Kneser-Ney interpolation	Kneser-Ney Back-off
1-gram	16.53	16.53	16.49	16.52	16.51	16.51
2-gram	20.51	19.52	22.42	22.62	22.56	22.64
3-gram	16.30	15.67	23.64	23.89	23.91	23.83

Table 4: BLEU scores on different N-gram models

From Table 4 we can infer the some important remarks as follows.

- Among discounting methods, Good-Turing and Witten-Bell have similar results and they are much better in comparison with Laplace methods, for all unigram, bigram, and trigram. It is interesting that this correlation is corresponding to the remarks presented in section 5.2.

- From the BLEU scores, we can not see the significant difference between Good-Turing, Interpolation Kneser-Ney, and Back-off Kneser-Ney. However, it is worth to emphasize that the best BLEU score is obtained at using the 3-gram model with Knerser-Ney interpolation. It is also corresponding to the intrinsic evaluation of LMs in section 5.2.

These experimental results and the above remarks allow us to draw a conclusion that Good-Turing is a simple method but good enough for applying to a language model in a SMT system. Beside that if the translation quality is important, we should use Interpolation Kneser-Ney on high order N-gram models.

7. Conclusion

In this paper we have investigated in detail Vietnamese LMs on both experimental and theoretical aspects. The experiments allow us to intuitively compare different LMs based on different smoothing methods. The obtained results when evaluating LMs independently or in applying for a SMT system has shown that Witten-Bell, Good-Turing, Interpolation Kneser-Ney, and Back-off Kneser-Ney are much better than Laplace methods. Among them, Interpolation Kneser-Ney is the best method on the both tests. The experiment also indicates that Good-Turing is a simple method but good enough, so it should be recommended to related NLP applications.

For further study, this work will be extended with higher order N-grams and larger data to get more evidences supporting for the conclusion. However in such the case, the problem becomes more complex in the aspects of computational time and storing memory. We will focus on this problem in the next study.

References

[1] Adrian David Cheok, Zhang Jian, Eng Siong Chng (2008) . Efficient mobile phone Chinese optical character recognition systems by use of heuristic fuzzy rules and bigram Markov language models. Journal of Applied Soft Computing . Volumn 8(2), pp. 1005 – 1017.

[2] S. Bergsma, Dekang Lin, and Randy Goebel.(2009). Web-scale N-gram models for lexical disambiguation. In IJCAI. Pp. 1507-1512.

[3] T. Brants, Ashok C. Popat, Peng Xu, Franz J. Och, and Jeffrey Dean. (2007). Large language models in machine translation. In EMNLP. pp. 858–867.

[4] P. Brown, S. Della Pietra, V. Della Pietra, and R. Mercer (1993). The mathematics of statistical machine translation: parameter estimation. Computational Linguistics, 19(2), pp. 263-311.

[5] E. Charniak, Kevin Knight, and Kenji Yamada. (2003) Syntax-based Language Models for Statistical Machine Translation. In Proceedings of Machine Translation Summit IX, pp. 40-46.

[6] Chen, Stanley F., and Joshua Goodman. (1996). An empirical study of smoothing techniques for language modeling. In ACL 34, pp. 3-18.

[7] S. F. Chen and J. Goodman, ``An Empirical Study of Smoothing Techniques for Language Modeling," TR-10-98, Computer Science Group, Harvard Univ., 1998.

[8] M. Collins , Brian Roark , Murat Saraclar. (2005) Discriminative syntactic language modeling for speech recognition, Proceedings of ACL. pp. 503-514.

[9] Herman Stehouwer, Menno van Zaanen. (2009). Language models for contextual error detection and correction. Proceedings of the EACL 2009 Workshop on Computational Linguistic Aspects of Grammatical Inference. pp. 41-48.

[10] Jay M. Ponte, Bruce W. Croft. (1998) A Language Modeling Approach to Information Retrieval. In Research and Development in Information Retrieval. pp. 275-281.

[11] F. Jelinek, B. Merialdo, S. Roukos, and M. Strauss. (1991), A Dynamic Language Model for Speech Recognition. Human Language Technology Conference , Proceedings of the workshop on Speech and Natural Language table of contents. pp. 293 – 295.

[12] Jin R., Hauptmann A.G. and Zhai C.(2002) Title Language Model for Information Retrieval. In Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval. pp. 42-48.

[13] D. Jurafsky, James H. Martin. (2000) “Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition”. Pages 189-232.

[14] S.M. Katz. (1987) “Estimation of probabilities from sparse data for the language model component of a speech recognizer” , IEEE Trans. on Acoustics, Speech and. Signal Proc. ASSP 35(3), pp. 400-401.

[15] Kneser Reinhard, and Hermann Ney. (1995) Improved backing-off for m-gram language modeling. In Proceedings of ICASSP-95, vol. 1, pp. 181–184.

[16] P. Koehn, F.J. Och, and D. Marcu (2003). Statistical phrase based translation. In Proceedings of the Joint Conference on Human Language Technologies and the Annual Meeting of the North American Chapter of the Association of Computational Linguistics (HLT/NAACL). pp. 127-133.

[17] P. Koehn, H. Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, W. Shen, C. Moran, R. Zens, C. Dyer, O. Bojar, A. Constantin, and E. Herbst. (2007) Moses: Open source toolkit for statistical machine translation. In ACL. Pages 177-180.

[18] C. Manning and Hinrich Schutze,(1999) Foundations of Statistical Natural Language Processing, MIT Press. Cambridge, May 1999.

[19] H. Q. Nguyen, Pascal NOCERA, Eric CASTELLI, TRINH Van Loan., "A novel approach in continuous speech recognition for Vietnamese, an isolating tonal language," in Proc. Interspeech'08, Brisbane, Australia, 2008, pp. 1149-1152.

[20] Papineni, K., Roukos, S., Ward, T., and Zhu, W. J. (2002). "BLEU: a method for automatic evaluation of machine translation" in ACL-2002: 40th Annual meeting of the Association for Computational Linguistics. pp. 311-318.

[21] Petrov, Slav, Aria Haghighi, and Dan Klein. (2008) Coarse-to-fine syntactic machine translation using language projections. In Proceedings of ACL-08. pp.108–116.

[22] Suryaprakash Kompalli, Srirangaraj Setlur, Venu Govindaraj. Devanagari (2009) OCR using a recognition driven segmentation framework and stochastic language models. International Journal on Document Analysis and Recognition. Volume 12 (2) Pages: 123-138

[23] A. Stolcke. (2002) SRILM – an extensible language modeling toolkit . In Proceedings of ICSLP, Vol. 2, pp. 901-904.

[24] O. Tran, A.C. Le, Thuy Ha, 2008. Improving Vietnamese Word Segmentation by Using Multiple Knowledge Resources. Workshop on Emirical Methods for Asian Languages Processing (EMALP), PRICAI.

[25] Zhang Jun-lin , Sun Le , Qu Wei-min , Sun Yu-fang, (2004) A trigger language model-based IR system, Proceedings of the 20th international conference on Computational Linguistics. pp. 680-686.

[26] D. Vergyri, A. Stolcke, and G. Tur, (2009) "Exploiting user feedback for language model adaptation in meeting recognition," in Proc. IEEE ICASSP, (Taipei), pp. 4737-4740.

[27] Witten Ian H., and Timothy C. Bell. (1991) The zero-frequency problem: Estimating the probabilities of novel events in adaptive text compression. IEEE Transactions on Information Theory, 37: 1085-1094.