International Journal of Computer Applications (0975 – 8887)
Volume 46– No.2, May 2012

Performance Analysis of Different Smoothing Methods on *n*-grams for Statistical Machine Translation

A. S. M Mahmudul Hasan[1], Saria Islam[2] and M. Arifur Rahman[3]
[1] Department of CSE, IBAIS University, Dhaka
[2] Department of CSE, IBAIS University, Dhaka
[3] Department of Physics, Jahangirnagar University, Savar

ABSTRACT

Smoothing techniques adjust the maximum likelihood estimate of probabilities to produce more accurate probabilities. This is one of the most important tasks while building a language model with a limited number of training data. Our main contribution of this paper is to analyze the performance of different smoothing techniques on *n*-grams. Here we considered three most widely-used smoothing algorithms for language modeling: Witten-Bell smoothing, Kneser-Ney smoothing, and Modified Kneser-Ney smoothing. For the evaluation we use BLEU (Bilingual Evaluation Understudy) and NIST (National Institute of Standards and Technology) scoring techniques. A detailed evaluation of these models is performed by comparing the automatically produced word alignment. We use Moses Statistical Machine Translation System for our work (i.e.Moses decoder, GIZA++, mkcls, SRILM, IRSTLM, Pharaoh, BLEU Scoring Tool). Here machine translation approach has been tested on German to English and English to German task. Our obtain results are significantly better than those obtained with alternative approaches to machine translation. This paper addresses several aspects of Statistical Machine Translation (SMT). The emphasis is put on the architecture and modeling of an SMT system.

Keywords

Machine Translation, SMT, smoothing, n-gram, parallel corpora, BLEU, NIST.

1. INTRODUCTION

Machine translation, sometimes referred to by the abbreviation MT, is a sub-field of computational linguistics that investigates the use of computer software to translate text or speech from one natural language to another. Current machine translation software often allows for customization by domain or profession (such as weather reports), improving output by limiting the scope of allowable substitutions.
This technique is particularly effective in domains where formal or formulaic language is used. It follows that machine translation of government and legal documents more readily produces usable output than conversation or less standardized text.
Improved output quality can also be achieved by human intervention: for example, some systems are able to translate more accurately if the user has unambiguously identified which words in the text are names. With the assistance of these techniques, MT has proven useful as a tool to assist human translators and, in a very limited number of cases, can even produce output that can be used as is (e.g., weather reports)[1].
The machine translation process may be stated as: a) Decoding the meaning of the source text and b) Re-encoding this meaning in the target language.

Statistical Machine Translation (SMT) utilizes statistical translation models whose parameters stem from the analysis of monolingual and bilingual corpora [2]. Statistical machine translation is based on a channel model. Given a sentence S in one language (German) to be translated into another language (English), it considers T as the target of a communication channel, and its translation S as the source of the channel. [3]

Statistical machine translation has emerged as the dominant paradigm in machine translation research. In statistical machine translation, it using probabilities which are estimated using collections of translated texts, called parallel corpora.

If we assign a probability P(S | T) to each pair of sentences (S, T), then the problem of translation is to find the source S for a given target T, such that P(S | T) is the maximum. According to Bayes rule, [13]

$$\mathrm{P(S|T)} = \frac{\mathrm{P(S)P(T|S)}}{\mathrm{P(T)}}$$

Since the denominator is independent of S, we have

$$\hat{\mathrm{S}} = \mathrm{arg_s\,max\,P(S)P(T|S)}$$

Statistical machine translation system must deal with the following three phrases [3]. **Modeling** Most statistical machine translation systems use N-gram for language modeling. Translation models rely on the concept of alignment. **Training (or Parameter estimation)** Training the free model parameters of the chosen statistical translation model using parallel and monolingual training data. **Search (or decoding)** The decoding algorithm is another crucial part of the statistical machine translation. Its performance directly affects translation quality and efficiency.

2. LANGUAGE MODELING AND N-GRAM MODELS

A language model is usually formulated as a probability distribution P(S) over string S that attempts to reflect, how frequently a string S occurs as a sentence. For example, for a language model describing spoken dialog, we might have P(HELLO) ≈ 0.01 since perhaps one out of every hundred sentences a person speaks is HELLO. On the other hand, we would have P(VOLVIC MONKEY MANGO PAPER) ≈ 0 and P(DOVE SYSTEM FUNKY) ≈ 0 seems it is extremely unlikely anyone would utter either string. Unlike the linguistics, grammatically it is irrelevant in language modeling; even though the string DOVE SYSTEM FUNKY is grammatical, we still assign a near-zero probability.
The most widely language models, by far n-gram language model. We introduce these models by considering the case n=2; this models are called bigram models. First, we notice that for the sentence S composed of the words w1, w2, … wl, without loss of generality we can express P(S) as

$$P(S) = P(w_1)P(w_2 \mid w_1).P(w_3 \mid w_2 w_1)\ldots.P(w_l \mid w_{l-1}\ldots w_1)$$

$$\Rightarrow P(S) = \prod_i^l P(w_i \mid w_{i-1}\ldots w_1)$$

In bigram models, we making the approximation that the probability of the word depends only on the identity of the immediately preceding word, giving us-

$$P(S) = \prod_i^l P(w_i \mid w_{i-1}\ldots w_1) \approx \prod_i^l P(w_i \mid w_{i-1})$$

For 3-gram will give us-

$$P(S) = \prod_i^l P(w_i \mid w_{i-1}\ldots w_1) \approx \prod_i^l P(w_i \mid w_{i-1}.w_{i-2})$$

3. SMOOTHING SCHEME

The term smoothing describes techniques for adjusting the maximum likelihood estimate of probabilities to produce more accurate probabilities. The name smoothing comes from the fact that these techniques tend to make distributions more uniform, by adjusting low probabilities such as zero probabilities upward, and high probabilities downward. Smoothing method generally not only prevents zero probabilities, but they also attempt to improve the accuracy of the model as a whole. Here, we used the most widely-used smoothing algorithms for language modeling: Witten-Bell smoothing, Kneser-Ney smoothing and modified Kneser-Ney smoothing.

3.1 Witten Bell Smoothing

Witten Bell smoothing was developed for the task of text compression and can be considered to be an instance of Jelinek-Mercer smoothing. The *n-th* order smoothed model are defined recursively as a linear interpolation between the nth order maximum likelyhood model and the *(n-1)th* order smooth model as following equation-

$$p_{WB}(w_i|w_{i-n+1}^{i-1}) = \lambda_{w_{i-n+1}^{i-1}} p_{ML}(w_i|w_{i-n+1}^{i-1}) + \left(1 - \lambda_{w_{i-n+1}^{i-1}}\right) p_{WB}(w_i|w_{i-n+2}^{i-1})$$

To compute the parameter $\lambda_{w_{i-n+1}^{i-1}}$ for Witten-Bell smoothing, we will need to use the number of unique words follow the history w_{i-n+1}^{i}. We can write the value as formally defined as

$$N_{1+}(w_{i-n+1}^{i-1}\bullet) = |\{w_i : c(w_{i-n+1}^{i-1}w_i) > 0\}|$$

The notation N_{1+} is meant to evoke the number of words that have one or more counts, and the ● formally defined as

$$1 + \lambda_{w_{i-n+1}^{i-1}} = \frac{N_{1+}(w_{i-n+1}^{i-1}\bullet)}{N_{1+}(w_{i-n+1}^{i-1}\bullet) + \sum_{w_i} c(w_{i-n+1}^{i})}$$

Substituting, we can rewrite equation as

$$p_{WB}(w_i|w_{i-n+1}^{i-1}) = \frac{c(w_{i-n+1}^{i} + N_{1+}(w_{i-n+1}^{i-1}\bullet) p_{WB}(w_i|w_{i-n+2}^{i-1})}{\sum_{w_i} c(w_{i-n+1}^{i}) + N_{1+}(w_{i-n+1}^{i-1}\bullet)}$$

To motivate Witten-Bell smoothing, with probability $\boldsymbol{\lambda_{w_{i-n+1}^{i-1}}}$ we should use the higher order model, and with probability $\boldsymbol{1 + \lambda_{w_{i-n+1}^{i-1}}}$ we should use the lower order model. The Good-Turing estimate provides another perspective on the estimation of the probability of novel words following a history. The Good-Turing estimate predicts that the probability of an event not seen in the training data is ***n₁/N***, the fraction of counts devoted to items that occur exactly once[4],[10]. Translating this value into the previous notation, we get-

$$\frac{N_1(w_{i-n+1}^{i-1}\bullet)}{\sum_{w_i} c(w_{i-n+1}^{i})}$$

Where,

$$N_{1+}(w_{i-n+1}^{i-1}\bullet) = |\{w_i : c(w_{i-n+1}^{i-1}w_i) = 1\}|$$

3.2 Kneser Ney Smoothing

Kneser and Ney (1995) have introduced this smoothing method[11]. Here, a lower-order distribution is a significant factor in the combined model only when few or no counts are present in the higher-order distribution. Consequently, they should be optimized to perform well in these situations. The motivation given in the original text is that we should select the lower-order distribution such that the marginals of the higher-order smoothed distribution should match the marginals of the training data. For example, for a bigram model we would like to select a smoothed distribution P_{kN} and the right-hand side is the unigram frequency of w_i found in the training data.[5]

For example, for a bigram model we would like to select a smoothed distribution P_{kN} that satisfies the following constraint on unigram marginals for all w_i:

$$\sum_{w_{i-1}} p_{KN}(w_{i-1}w_i) = \frac{c(w_i)}{\sum_{w_i} c(w_i)}$$

Here, a different derivation of the resulting distribution than is presented by Kneser and Ney (1995).

$$p_{KN}(w_i|w_{i-n+1}^{i-1}) = \frac{max\{c(w_{i-n+1}^{i}) - D, 0)\}}{\sum_{w_i} c(w_{i-n+1}^{i})} + \frac{D}{\sum_{w_i} c(w_{i-n+1}^{i})} N_{1+}(w_{i-n+1}^{i-1}\bullet) p_{KN}(w_i|w_{i-n+1}^{i-1})$$

Now our aim to find the unigram distribution $p_{KN}(w_i)$

$$\frac{c(w_i)}{\sum_{w_i} c(w_i)} = \sum_{w_{i-1}} p_{KN}(w_i|w_{i-1}) p(w_{i-1})$$

For (w_{i-1}), we simply take the distribution found in the training data

$$p(w_{i-1}) = \frac{c(w_{i-1})}{\sum_{w_{i-1}} c(w_{i-1})}$$

Substituting and simplifying, we have

$$c(w_i) = \sum_{w_{i-1}} c(w_{i-1}) p_{KN}(w_i|w_{i-1})$$

Substituting in main equation

$$c(w_i) = c(w_i) - N_{1+}(\bullet w_i)D + D p_{KN}(w_i) N_{1+}(\bullet\bullet)$$

Where

$$N_{1+}(\bullet w_i) = |\{c(w_{i-1}w_i) > 0\}|$$

Is the number of different words w_{i-1} that proceed w_i in the training data and where

$$N_{1+}(\bullet\bullet) = \sum_{w_{i-1}} N_{1+}(w_{i-1}\bullet) = |\{(w_{i-1}, w_i) : c(w_{i-1}w_i) > 0\}|$$
$$= \sum_{w_i} N_{1+}(\bullet \mathrm{w_i})$$

Solving for $p_{KN}(w_i)$, we get

$$p_{KN}(w_i) = \frac{N_{1+}(\bullet \mathrm{w_i})}{N_{1+}(\bullet\bullet)}$$

Generalizing to higher-order models, we have that [5]

$$p_{KN}(w_i|w_{i-n+2}^{i-1}) = \frac{N_{1+}(\bullet \mathrm{w_{i-n+2}^{i}})}{N_{1+}(\bullet \mathrm{w_{i-n+2}^{i-1}}\bullet)}$$

Where

$N_{1+}(\bullet \mathrm{w_{i-n+2}^{i}}) = |\{w_{i-n+1} : c(w_{i-n+1}^{i}) > 0\}|$ and

$$N_{1+}(\bullet w_{i-n+2}^{i-1}\bullet) = \left|\{(w_{i-n+1}, w_i): c(w_{i-n+1}^{i}) > 0\}\right| = \sum_{w_i} N_{1+}(\bullet w_{i-n+2}^{i})$$

3.3 Modified Kneser Ney Smoothing

Chen and Goodman introduced modified Kneser-Ney that have found excellent performance[12]. Instead of using, a single discount D for all nonzero counts as in Kneser-Ney smoothing, we have three different parameters D_1, D_2 and D_{3+} that are applied to n-grams with one, two, and three or more counts respectively.

$$p_{KN}(w_i|w_{i-n+1}^{i-1}) = \frac{c(w_{i-n+1}^{i}) - D(c(w_{i-n+1}^{i}))}{\sum_{w_i} c(w_{i-n+1}^{i})} + \gamma(w_{i-n+1}^{i-1}) p_{KN}(w_i|w_{i-n+2}^{i-1})$$

Where

$$D(c) = f(x) = \begin{cases} 0, & if\ c = 0 \\ D_1, & if\ c = 1 \\ D_2, & if\ c = 2 \\ D_{3+}, & if\ c \geq 3 \end{cases}$$

To make the distribution sum, we take

$$\gamma(w_{i-n+1}^{i-1}) = \frac{D_1 N_1(w_{i-n+1}^{i-1}\bullet) + D_2 N_2(w_{i-n+1}^{i-1}\bullet) + D_{3+} N_{3+}(w_{i-n+1}^{i-1}\bullet)}{\sum_{w_i} c(w_{i-n+1}^{i})}$$

Where $N_2(w_{i-n+1}^{i-1}\bullet)$ and $N_{3+}(w_{i-n+1}^{i-1}\bullet)$ defined analogously to $N_1(w_{i-n+1}^{i-1}\bullet)$.

This modification is the ideal average discount for n-grams with one or two counts is substantially different from the ideal average discount for n-grams with higher counts. The analogous relation for modified Kneser –Ney smoothing are, [5]

$$\gamma = \frac{n_1}{n_1 + 2n_2}$$

$$D_1 = 1 - 2\gamma \frac{n_2}{n_1}$$

$$D_2 = 2 - 3\gamma \frac{n_3}{n_2}$$

$$D_{3+} = 3 - 4\gamma \frac{n_4}{n_3}$$

4. EVALUATION

Evaluation is needed to compare the performance of different MT engines or to improve the performance of a specific MT engine. Automatic evaluation metrics for machine translation system, such as BLEU and NIST are now well established. We used this two technique for analysis the performance of our experiment.

BLEU (Bilingual Evaluation Understudy) is an algorithm for evaluating the quality of text which has been machine-translated from one natural language to another. BLEU heavily rewards large N-gram matches between the source and target translations. [5].

NIST evaluation metric is based on the BLEU metric, but with some alterations. Whereas BLEU simply calculates n-gram precision considering of equal importance each n-gram, NIST calculates how informative a particular n-gram is, and the rarer a correct n-gram is, the more weight it will be given. NIST also differs from BLEU in its calculation of the brevity penalty, and small variations in translation length do not impact the overall score as much. Again, NIST score is always referred to a given n-gram order ($NIST_n$, usually n being 4). [6].

5. SIMULATION

One of the major advantages of Statistical Machine Translation (SMT) is that it can be learned automatically. This is also the main reason why the process of developing a SMT system differs significantly from a classical rule-based system. Here we will show a work flow of SMT after installation of all the required tools. The steps for SMT are [7] –

1. Prepare Data
2. Build Language Model
3. Train Model
4. Tuning (i.e., Optimize System Component Weights, a.k.a. Minimum Error Rate Training)
5. Run System on Development Test Set
6. Evaluation

Two figures are given bellow to show the simulation process. Figure 1 shows how to create a language model and figure 2 represents how to score a language model.

For preparing data the parallel corpus has to be converted into a format that is suitable to the GIZA++ toolkit. Two vocabulary files are generated and the parallel corpus is converted into a numberized format. The language model should be trained on a corpus that is suitable to the domain. If the translation model is trained on a parallel corpus, then the language model should be trained on the output side of that corpus, although using additional training data is often beneficial. There are many tools available for building a language model. Our decoder works with the following language models:

- The SRI language modeling toolkit
- The IRST language modeling toolkit

The training process takes place in nine steps, all of them executed by the script train-factored-phrase-model.perl. [8]

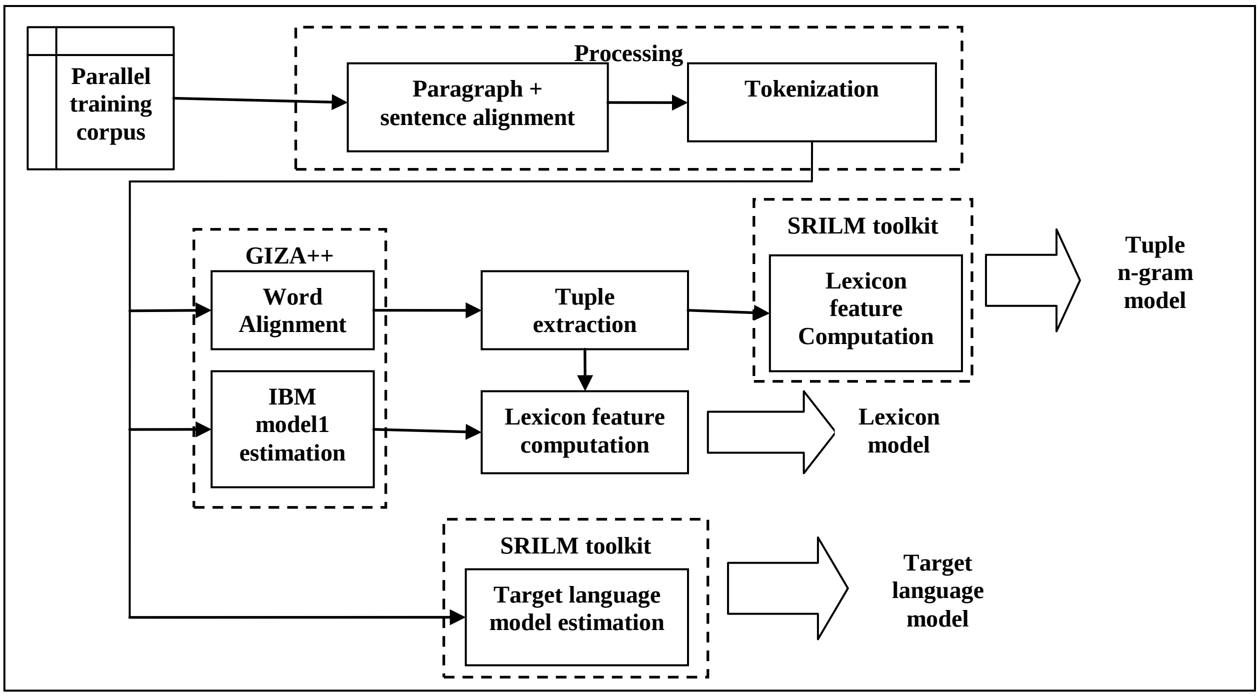


Figure 1: Creating a language model

The steps are:

1. Prepare data
2. Run GIZA++
3. Align words
4. Get lexical translation table
5. Extract phrases
6. Score phrases
7. Build lexicalized reordering model
8. Build generation models
9. Create configuration file

The training script produces a configuration file `moses.ini` which has default weights of questionable quality. That's why we need to obtain better weights by optimizing translation performance on a development set. This is done with the tuning script `mert-moses-new.pl`. This new version of the minimum error rate tuning script is based on a new C++ software. The new mert implementation is a stand-alone open-source software. The only interaction between Moses and the new software is given by the script mert-moses-new.pl itself. This new implementation of mert stores feature scores and error statistics in separate files) possibly in a binary format) for each nbest-list (at each iteration), and use (some of) these files to optimize weights. At the moment weight optimization can be based on either BLEU or PER.

Run system on development test requires four steps. They are:

- Tokenize test set (First tokenize the test set)
- Lowercase test set (Convert them to lowercase data)
- Filter the model to fit into memory
- Decode with Moses

Evaluation will be done both automatically as well as by human judgment.

- Manual Scoring: We will collect subjective judgments about translation quality from human annotators. If you participate in the evaluation, we ask you to commit about 8 hours of time to do the manual evaluation. The evaluation will be done with an online tool.
- In difference to the previous years, we expect the translated submissions to be in recased, detokenized, XML format, just as in most other translation campaigns (NIST, TC-Star). The official BLEU scoring tool will be the NIST scoring tool.

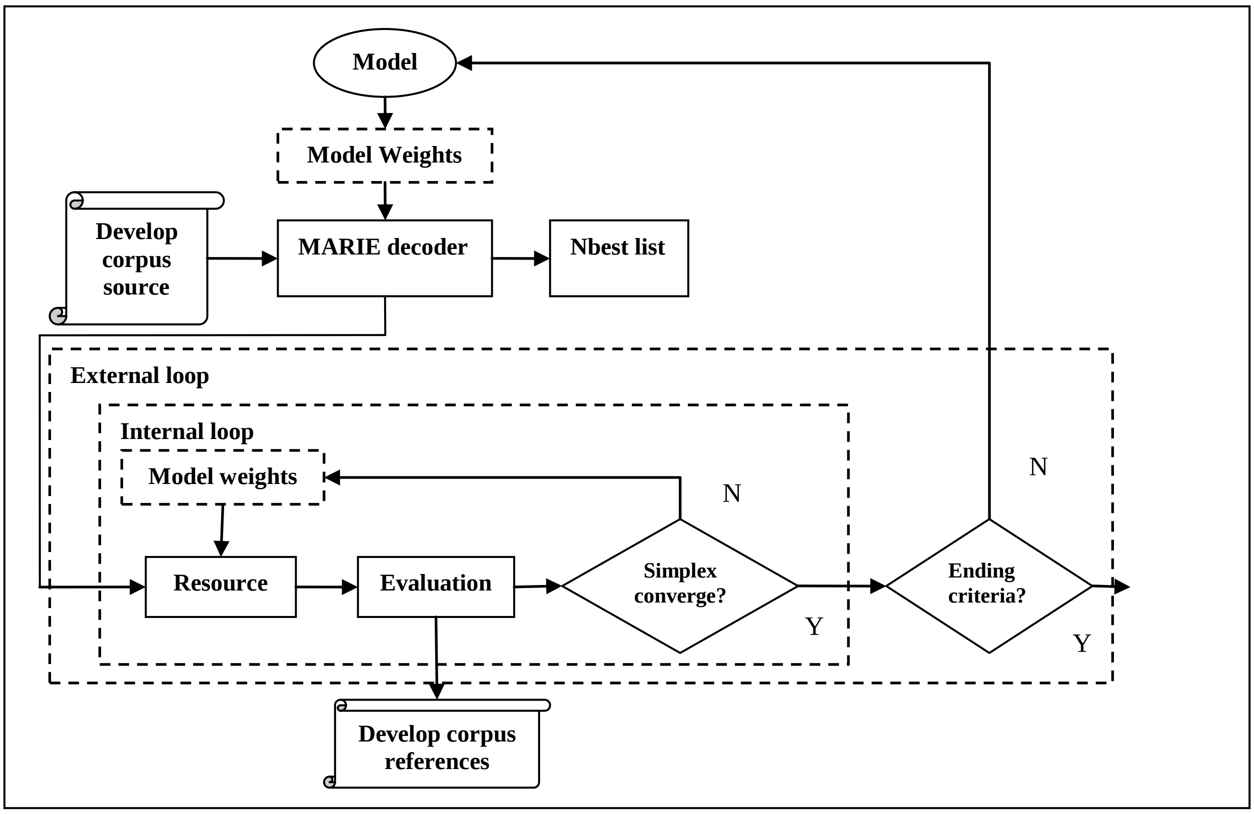


Figure 2: Scoring of language model

BLEU score measures the precision of unigrams, bigrams, trigrams and four grams with respect to a whole set of reference translations with a penalty for too short sentences. Unlike all other evaluation criteria used here, BLEU measures accuracy, i.e. the opposite of error rate. Hence, large BLEU scores are better. [9]

6. RESULTS AND ANALYSIS

We have observed their performances on the same dataset of these three language models by re-using the given interpolation weights, estimating weights from scratch through the MERT and a comparative study of their BLEU and NIST scores and give some decision on the obtained results. The system specification of the machine that we use for our experiment is-

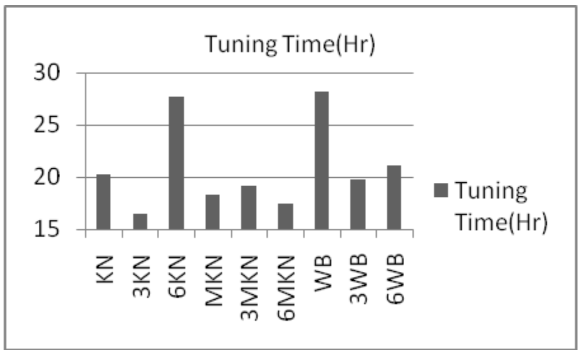
- Intel Core 2 Duo CPU
- Processor 2.66 GHz
- RAM 1GB

6.1 Tuning and Decoding Time

The time complexity of the statistical machine translation is very high. Though tuning time and decoding time depends on the processor clock speed and the capacity of memory, Here we show the time for tuning the weights and decoding for some sample cases-

Table 1: Tuning and Decoding time

Smoothing Method	Tuning Time(Hr)	Decoding Time(Hr)
Original Kneser-Ney discounting**(KN)**	20.3	3.1
3gram original Kneser-Ney discounting**(3KN)**	16.5	2.45
6gram original Kneser-Ney discounting**(6KN)**	27.75	3.1
modified Kneser-Ney discounting**(MKN)**	18.25	1.5
3gram modified Kneser-Ney discounting**(3MKN)**	19.15	2.5
6gram modified Kneser-Ney discounting**(6MKN)**	17.5	2.1
Witten-Bell discounting**(WB)**	28.2	3.15
3gram Witten-Bell discounting**(3WB)**	19.8	2.55
6gram Witten-Bell discounting**(6WB)**	21.1	2.33



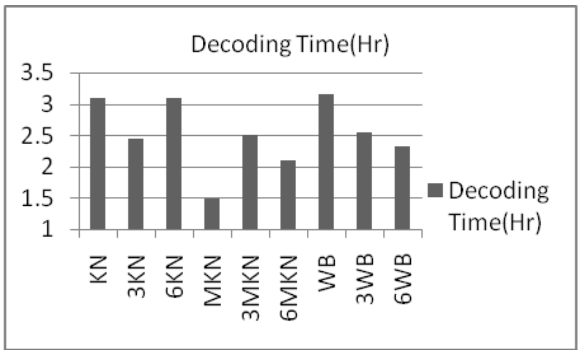


Figure 3: Tuning and decoding time for different smoothing method

6.2 BLEU Score Analysis

The BLEU scores of the smoothing methods without recasing and detokenization are given bellow:

Table 2: BLEU Score Analysis

Smoothing Methods	BLEU Scores
Original Kneser-Ney discounting	16.35
3gram original Kneser-Ney discounting	16.21
6gram original Kneser-Ney discounting	16.26
Modified Kneser-Ney discounting	16.43
3gram modified Kneser-Ney discounting	16.32
6gram modified Kneser-Ney discounting	16.26
Witten-Bell discounting	16.29
3gram Witten-Bell discounting	16.20
6gram Witten-Bell discounting	16.26

Table 3: BLEU score for different gram

Smoothing Methods	BLEU Scores		
	Order 0	Order 3	Order 6
KN	16.35	16.21	16.26
MKN	16.43	16.32	16.26
WB	16.29	16.20	16.26

Form the BLEU scores of the Table 3 and form the Figures above, we can draw some decision. They are-

- Here we can see that modified Kneser-Ney discounting method for order 0 has the maximum BLEU score. So we can say that modified Kneser-Ney discounting method perform best for the order 0.
- In case of modified Kneser-Ney discounting method, the BLEU score decreases with the increment of the order of n. But for Original Kneser-Ney and Witten Bell smoothing technique, this characteristic is absent.
- For order 6, we found that the BLEU scores for all the discounting methods are same. So, we can say that with the increment of order the all the methods show quite similar performance.
- Weight obtained from one smoothing method may significantly reduce the performance if it is used in different method.

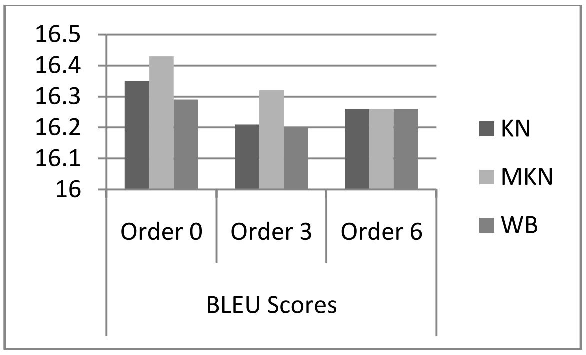


Figure 4: BLEU scores

6.3 NIST Score Analysis

The NIST and BLEU (with recaseing and detokenization) for the three discounting methods are given bellow with Evaluation of any-to-en translation using src set "devtest2006" (1 docs, 2000 segs), ref set "devtest2006" (1 refs) and tst set "devtest2006" (1 systems).

Table 4: The NIST and BLEU (with recaseing and detokenization)

Smoothing Methods	NIST Scores	BLEU Score
KN	3.8079	0.0884
3KN	3.7558	0.0884
6KN	4.8485	0.1513
MKN	3.8624	0.0908
3MKN	3.8023	0.0893
6MKN	4.8485	0.1513
WB	3.7559	0.0892
3WB	3.7723	0.0879
6WB	4.8485	0.1513

Figure 5 and 6 shows the graphical representation of NIST and BLEU score recaseing and detokenization.

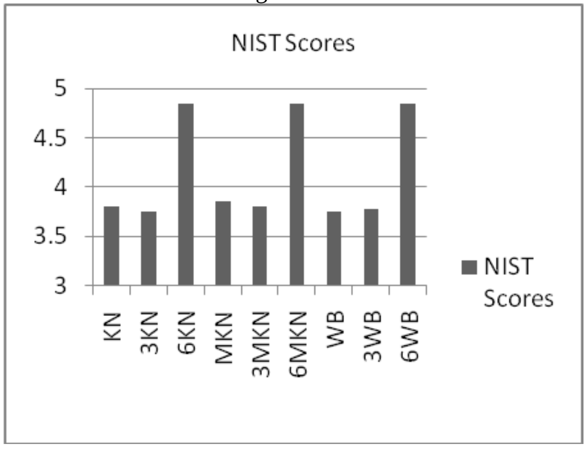


Figure 5: The NIST score (with recaseing and detokenization)

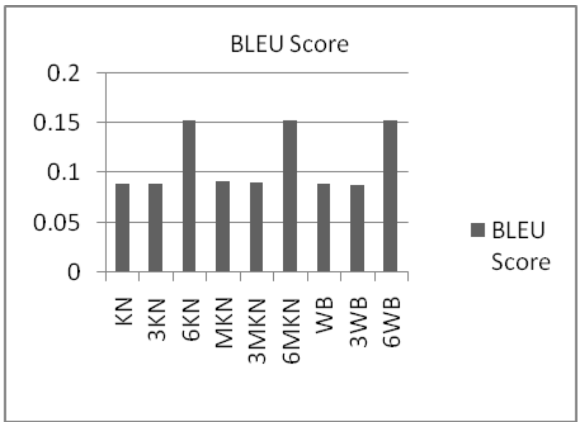


Figure 6: The BLEU (with recaseing and detokenization)

7. CONCLUSION

This paper has considered the smoothing techniques for adjusting the maximum likelihood estimate of probabilities to produce more accurate probabilities. It considered n-grams (up to 6-grams) based approach for the SMT. Using NIST/BLEU scoring tools and figure out different performance of smoothing method. The 6-grams on Original Kneser-Ney, Modified Kneser-Ney and Witten-Bell discounting methods gives best score. Similar performance was observed both for BLEU and NIST score. Only for Witten-Bell smoothing method NIST scores increases with the increment of grams, but this is not true for the other two smoothing methods.

8. REFERENCES

[1]. Machine Translation, Wikipedia, en.wikipedia.org/wiki/Machine_translation, [last access: 06-04-2012].

[2] F.J. Och, and H. Ney (2004), “The alignment template approach to statistical machine translation”, Computational Linguistics, Vol. 30, no 4,.

[3] Ye-Yi Wang and Alex Waibel, (1997) “Decoding Algorithm in Statistical Machine Translation”.

[4] Kishore Papineni, Salim Roukos, Todd Ward, Wei-Jing Zhu, (2001) “IBM Research Report Bleu: a Method for Automatic Evaluation of Machine Translation”, RC22176 (W0109-022).

[5] Enrique Alfonseca and Diana Perez, (2004) “Automatic Assessment of Open Ended Questions with a BLEU-inspired Algorithm and shallow NLP”.

[6] Josep M. Crego Clemente, (2008) “Architecture and Modeling for N-gram-based Statistical Machine Translation”.

[7] Pharaoh, www.isi.edu/licensed-sw/pharaoh/, [last access: 06-04-2012].

[8] Philipp Koehn, (2009) “Statistical Machine Translation System User Manual and Code Guide”, University of Edinburgh.

[9] K. A. Papineni, S. Roukos, T. Ward, W. J. Zhu, (2001) “BLEU: a method for automatic evaluation of machine translation”. Technical Report RC22176 (W0109-022), IBM Research Division, Thomas J. Watson Research Center, Yorktown Heights, NY.

[10] Timothy C. Bell, John G. Cleary, Ian H. Witten, (1990) "Text Compression" Prentice Hall.

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

[12] Stanly F. Chan and Josua Goodman (1998), "An Emperical Study of Smoothing technique for Language Modeling", Computer Science group, Harvard University, Cambridge, Massachusetts.

[13] Bayes, Thomas, and Price, Richard (1763). "An Essay towards solving a Problem in the Doctrine of Chance. By the late Rev. Mr. Bayes, communicated by Mr. Price, in a letter to John Canton, M. A. and F. R. S.". Philosophical Transactions of the Royal Society of London 53 (0): 370–418.

9. AUTHORS PROFILE

A.S.M Mahmudul Hasan is the lecturer of IBAIS (International Business Administration and Information Technology) University in Dhaka, Bangladesh. He received a B.Sc degree from Jahangirnagar University in 2010 and continuing his M.Sc from the same university in Computer Science and Engineering. His research interest includes natural language processing, system administration and networking and telecommunication.

Saria Islam obtained her B.Sc (Hons) and continuing M.Sc in Computer Science and Engineering from Jahangirnagar University She is now serving the IBAIS University as a lecturer in computer science and engineering department. Her current interest research areas include machine learning, web programming simulation and modeling in computer science.

M. Arifur Rahman has completed his Masters degree in Human Language Technology and Interfaces (HLTI) under the Department of Information Engineering and Computer Science at University of Trento, Italy. He received his B.Sc. (Honors) degree from the Department of Computer Science and Engineering, Jahangirnagar University, Bangladesh at 2001. His major areas of interest include Machine Learning, Natural Language Processing, and Digital Signal Processing. He has authored a book titled “Fundamentals of Communication” and more than 10 international journal and conference papers. At present he is serving as an Assistant Professor in the Department of Physics, Jahangirnagar University, Savar, Dhaka, Bangladesh.