Tuning Statistical Machine Translation Parameters Using Perplexity[1]

Ahmed Ragab Nabhan
Department of Mathematics, Faculty of Science, Fayoum Branch, Cairo University, Egypt.
E-mail: ragab@claes.sci.eg

Ahmed Rafea
Computer Science Dept., American University in Cairo,113, Sharia Kasr El-Aini, P.O. Box 2511, 11511, Cairo, Egypt.
E-mail: rafea@aucegypt.edu

Abstract

Statistical Machine Translation (SMT) involves many tasks including modeling, training, decoding, and evaluation. In this work, we present a methodology for optimizing the training process to get better translation quality using the well known GIZA++ SMT toolkit. The methodology is based on adjusting the parameters of GIZA++ that affect the generation of the translation model. When applying the methodology, an average improvement of 7% has been achieved in the translation quality.

1. Introduction

Towards the goal of getting reliable statistical translation results, researchers focus their efforts on enhancing the way different tasks of machine translation are performed. Some researchers focus on innovating better models for word based [1], phrases based [2], and syntax based [3] statistical machine translation. Other researchers consider the development of better decoding algorithms [4]. Other efforts are dedicated to develop more reliable evaluation methods [5]. However, little effort has been dedicated to improve the training process, despite its effect on translation quality. The Estimate Maximization (EM) algorithm has been used so long for estimating model parameters. Little effort was dedicated to innovating better training algorithms. Most algorithms of statistical translation are imported from the speech recognition area.

GIZA++ is a popular toolkit for producing translation models and word alignments. Although GIZA++ produces word based translation models, it is still used in building some phrase based translation models through the word alignments it generates as a byproduct of the training process. These alignments can be used to extract bilingual phrases from parallel corpus by some methods such as those used by Koehn et al. [2]. This toolkit comes with parameters that exist to modify the training process. These parameters come with default values. However, the parameters are domain and data specific and should be adjusted for each new corpus we use.

In a previous work, we tuned GIZA++ on a limited corpus of 2500 sentence pair using translation quality as a criterion of the goodness of parameter value [6]. In this paper, we extend the work with two objectives in mind. Firstly, we want to test the hypothesis that tuning GIZA++ using a small corpus can get good parameter values that we can use with larger corpora. Secondly, we investigate using perplexity as a performance metric to tune GIZA++. The rationale of using perplexity other than translation quality as a measurement of how good is a parameter value will be explained in section 4.2.

The article is organized as follows. Section 2 represents a review of statistical machine translation. In section 3, we state the motivation for the tuning procedure. In section 4, we provide a methodology for parameter tuning. In section 5, we report our experiments and the results.

2. Background

Statistical machine translation (SMT) aims at extracting general translation rules from large amount of sentence pairs that are aligned to each other. Models of statistical machine translation are based on the notion of word alignment. An alignment between a pair of sentences is an object that links a word in the target sentence with a word in the source sentence from which it arose [7]. Figure 1 shows a possible alignment between an English sentence and an Arabic sentence.

Alignments are the basis on which translation models are

[1] Proceedings of the IEEE International Conference on Information Reuse and Management (IRI 2005)

built. For two sentences that are translations to each other, there is more than one possible alignment. Each alignment is given a probability value that represents how sure we are about the alignment.

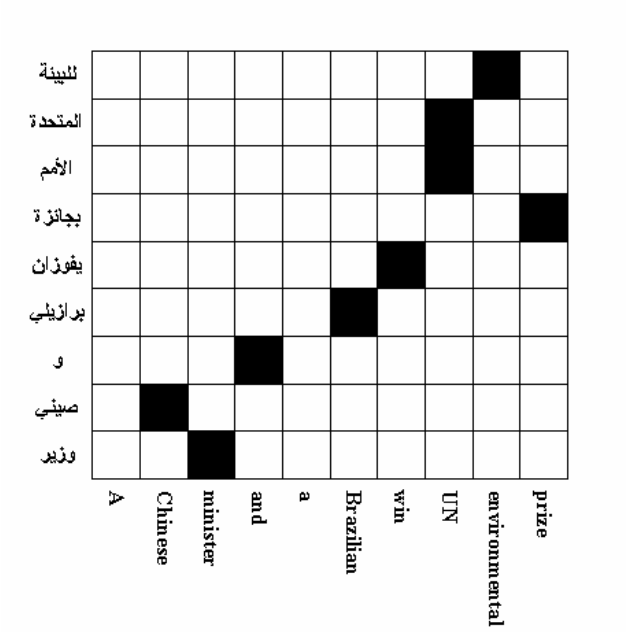

Figure.1 A word alignment example

The probability of an alignment ***a*** given a source sentence ***e*** and its target translation ***f*** can be expressed as follows:

$$P(a \mid e, f) = \frac{P(a, f \mid e)}{p(f \mid e)} \tag{1}$$

Summing over a for both sides yields:

$$\sum_a P(a \mid e, f) = \sum_a \frac{P(a, f \mid e)}{P(f \mid e)} \tag{2}$$

The left hand side sums to 1 and P (f | e) can be factored out, and finally we get the translation model equation:

$$P(f \mid e) = \sum_a P(a, f \mid e) \tag{3}$$

Hence, the translation model probability P (f | e) is the sum of all probabilities of producing a target string ***f*** and an alignment **a** given a source string ***e***.

In order to estimate the probability P (a, f | e), Brown et al [7] introduced a series of five statistical models each of which contributes to the calculation of p (a, f | e). Vogel et al [8] introduced Hidden Markov Model (HMM) as an alternative to the standard Model 2.

The Estimate Maximization (EM) algorithm is used to estimate the parameter values for the models [9]. GIZA++ applies the EM algorithm on a parallel text to build the translation model in the form of a set of tables.

3. Problem

In the following subsections, we present some issues that make good reasons for the existence of GIZA++ parameters.

3.1. Rare Events

Rare events cause a problem in probability distributions. As a solution to this problem, smoothing factors are introduced. Although smoothing is a necessity in language modeling, it was not used in the original IBM system for translation modeling [10]. The vocabulary was chosen to be all words that appear at least twice in the training corpus; single occurrence words were replaced by a special unknown token. Hence, no smoothing was needed.

The designers of GIZA++ proposed to use smoothing while building the translation model. Smoothing the probability distribution is useful to handle rare words in the data, even the ones that appear only once in the corpus.

3.2. Overfitting

Overfitting means fitting too much the training data and the model starts to degrade performance on test data. A model with overfitting does not predict well. To reduce the effect of overfitting, GIZA++ uses a number of smoothing parameters for various models. In each training iteration, alignment probabilities are smoothed using certain formula that contains a smoothing factor. For example, in HMM training iterations, the following formula is used for smoothing probability values:

$$P(aj \mid aj-1, I) = \alpha.I / I + (1-\alpha).P(aj \mid aj-1, I)$$

Where a_j is the source word position and a_{j-1} is position of the previous source word and I is the length of the source sentence. Here α is a smoothing factor for HMM. In GIZA++, this factor is represented by a parameter named emalsmooth.

3.3. Training Scheme

A training scheme specifies the sequence of used

models and the number of training iterations used for each model [1]. For example, we may choose a training scheme that uses five iterations for Model1 and three iterations for Model2, three iterations for Model3 and three iterations for Model4. Another training scheme may use HMM instead of Model2.

By default, GIZA++ uses HMM instead of Model2. However, HMM has a problem that it does not work well if there are large jumps due to different word orderings in the language pairs [11]. In our experiments on Arabic–English translation, the performance of Model2 outperforms the HMM model, with default parameter values. For this reason, the training scheme should be modified to suit the training data. GIZA++ has a number of parameters that define the number of training iterations for each model.

3.4. Other Issues

Aside from smoothing factors and training scheme, some GIZA++ parameters are effective in producing better results if they are set to the right values. These parameters should be adjusted empirically to get the most out of the training corpus. For instance, GIZA++ estimates the probability of not inserting a spurious word; p0. P0 is calculated from word alignments of the training corpus by inspecting how many words are aligned to the special NULL word. Of course, if the word alignments are not accurate, the calculation of p0 is not precise and needs to be calculated separately. P0 can be calculated from a manually aligned corpus or by empirically tuning the value using a test sample until we get the best translation quality.

4. Tuning GIZA++

GIZA++ uses a set of parameters to handle the issues presented in the previous section. These parameters already have default values. However, these values are domain specific and should be optimized for each new corpus [12].

In a previous work, we proposed a tuning methodology to obtain good parameter values for GIZA++ [6]. We used translation quality score as a measurement for the goodness of parameter values.

In this work, we further enhance the tuning methodology by using a more robust measurement to determine how good a parameter value. We also test the hypothesis that we can tune GIZA++ on a small corpus to obtain parameter values that can be used to get improved quality when applied to a larger corpus in the same domain.

4.1. Methodology

In order to tune GIZA++ for translation quality, we classified parameters according to certain criteria. Some parameters are general; in the sense that they are not modifying the training of a specific model and they exist for efficient training or they have a global effect on the training process. We also classified parameters according to whether they are discrete value or real value parameters. Discrete value parameters can be tuned at low cost since there are few discrete values to try out. Real value parameters were optimized by using the Genetic Algorithm (GA). We further classified parameters with regard to the models they modify. GIZA++ uses different smoothing parameters for HMM, Model 2, 3, and 4. In our experiments, we study two basic training schemes: one that uses Model 2 and the other one uses HMM. Each training scheme was evaluated separately.

4.1.2 GIZA++ Parameters

This subsection introduces the parameters that are covered by our study.

General parameters

Most of these parameters exist for efficient training. These parameters mainly determine cutoff and threshold levels. Examples of general parameters are p0, probsmooth, countincreasecutoff, and mincountincrease. Probcutoff is a cutoff threshold value that affects Model1.

HMM Parameters

The following parameters affect HMM training.

emprobforempty

This parameter represents the probability of a transition to NULL in the HMM network.

emalsmooth

A smoothing factor for alignments probabilities of HMM.

Model 2-4 parameters

These parameters are mainly smoothing factors for alignment probabilities. These parameters are model23smoothfactor and model4smoothfactor.

We tuned parameters in the order of the models they affect; that is; we tuned general parameters, then HMM (or Model 2) parameters, then Model 3 and Model 4 parameters

4.2. Performance Metrics

In this section we discuss how to determine the fitness of each member (parameter value) of the generation during the running of the GA.

In a previous work, we used translation quality as a measurement of how a GIZA++ parameter value is better [6]. Although good results were obtained using this method, it has some problems. One problem is the decoding time for the tuning test sample. Another factor is that a language model has a weight in the final scoring of translation formula. In this way we are not measuring the quality of translation model but also the quality of language model. A third problem is the potential of bias of test data. A forth problem is that the searching algorithms used while decoding can yield comparative results.

Another alternative to translation quality is to use training data perplexity as a measurement of the goodness of GIZA++ parameter values. Brown et al used training perplexity to compare the performance of different models [7]. Al-Onaizan et al. used perplexity as a performance metric in [10]. Perplexity is defined by the formula:

$$Perp = 2^{\frac{-1}{N}\sum \log p(f|e)}$$

The summation is over all sentence pairs of the corpus and P (f | e) is the translation model probability and N is the number of words in the corpus.

Using perplexity is an attractive idea as an objective evaluation of models compared to the subjective evaluation using the translation quality [10]. It has many benefits. Firstly, it is calculated over all the training data, and thus is more robust in determining the goodness of GIZA++ parameters values. Secondly, using perplexity is more efficient since GIZA++ already calculates perplexity during training and hence we save decoding time.

Perplexity has been doubted to be a good measure of quality despite the benefits of using it to evaluate models. Mathematically, lower perplexity means better model. But how perplexity related to translation quality? The experiment we conducted answers that question.

4.3. Tuning Algorithm

We used GA for tuning real valued parameters for the same reasons explained in our previous paper [6]. The reason is mainly because Genetic Algorithms have the advantage of not stopping at local optima and can perform well on noisy data.

We used a population size of 16 members. Each member represents a valid value for the parameter. The first generation was initiated randomly, except the first member which is seed with the default value for this parameter. This assures that the best value the GA will get so ever is guaranteed to be more fit that the default value. To get the fitness of each member, we run GIZA++ with the new parameter settings (the member value and previously tuned parameters).

Members with the lowest perplexity score are selected for reproduction phase, and new generation is produced using standard crossover and mutation operators of GA. The following pseudo code represents the tuning of a parameter.

Procedure Parameter_Tuning:

1. Generate random values of the parameter to initialize the first generation
2. For each value j in the generation
   Run GIZA++ with the parameter value j
   Assign model perplexity to member fitness
   Next j
3. For i:=1 to Num_of_optimization_runs do
   Choose the Best members (values) with the lowest perplexity
   Produce next generation using GA operators
   For each value j in the new generation do
   Run GIZA++ with the parameter value j
   Assign model perplexity to member fitness
   Next j
   Next i
4. Report the member with the lowest perplexity

End.

5. Experiments and Results

The objective of the experiments is to check two main assumptions: firstly, testing the hypothesis that tuning GIZA++ using a small corpus can get good parameter values that we can use with larger corpora, and secondly, using perplexity as a performance metric to tune GIZA++.

We conducted our experiments using a corpus of 27,000 sentence pairs and a corpus of 35,000 sentence pairs, of maximum sentence length of 30 words. The domain of the corpus is the news domain. The baseline

system is composed of the CMU language modeling toolkit [13], and the ISI ReWrite decoder [14].

The objective of the first experiment was to test the hypothesis that tuning GIZA++ with translation quality as a metric [6] using a small corpus can yield good parameters values that can be used for a larger corpus. To do this, we select 2,500 sentence pair from the 27,000 corpus as a tuning corpus. We run the tuning program using the small corpus and get the final parameter values to train the larger corpus with these values. Then we used the translation model to translate an in-context test set of 100 sentences extracted using the tool whittle [15]. The translation quality of the test data has increased by 1.1%

The objective of the second experiment was to test the effect of optimizing GIZA++ parameters using perplexity as a fitness function for the genetic algorithm. This time we tune GIZA++ using the whole corpus. At the end of the tuning program, we obtain the final parameter values the tuning algorithm found out, and use them to build the translation model. We measure the translation quality of the 100 test sample translation using the BLEU metric [5]. The translation quality has increased by 7.25%

The third experiment was much like the second one but with the exception that we started the tuning algorithm with a previously tuned parameter values we get from the first experiment. The objective is to let the tuning algorithm, using perplexity, start with good values obtained from tuning by translation quality using the small corpus. The translation quality has increased by 8.9%. Table1 summarize the results of running these three experiments.

We repeated the same three experiments using a larger corpus of 35,000 sentence pairs, with 30 words maximum sentence length. Another test sample was selected from the corpus using whittle. Table2 shows the corresponding results for the 35,000 corpus.

Table.1 Effect of tuning GIZA++ using 27,000 sentence pairs corpus

Exp	Description	BLEU	Percentage
	Default Parameter settings of GIZA++	0.1725	
1	Tuning with translation quality with 2500 sentence pairs corpus	0.1744	1.1%
2	Tuning with perplexity using whole corpus	0.1850	7.25%
3	Start tuning with perplexity with good parameter values	0.1880	8.9%

We observe that tuning with perplexity (experiment 2) yields a higher improvement than with translation quality (experiment 1). It may be due the limitations of using a small corpus to save time in training and make use of the time for decoding the test sample. Tuning with larger corpus with perplexity is feasible because we lose no time in calculating perplexity because GIZA++ does this during training and we do not lose time in decoding. An advantage of using the larger corpus is the robustness of the measurement (perplexity) as it is calculated for the whole data, not for a small sample.

Table.2 Effect of tuning GIZA++ using 35,000 sentence pairs corpus

Exp	Description	BLEU	Percentage
	Default Parameter settings of GIZA++	0.2665	
1	Tuning with translation quality with 2500 sentence pairs corpus	0.2704	1.4%
2	Tuning with perplexity using whole corpus	0.2906	9%
3	Start tuning with perplexity with good parameter values	0.3050	14%

6. Conclusion

The experiments showed that tuning the GIZA++ parameters using a small corpus, using translation quality as performance metric, then using these parameters on a larger corpus enhance the translation quality on average of 1.25%. Using perplexity as a performance metric in tuning the parameters led to enhancing the translation quality by 8%. Combing the two methods led to an enhancement of translation quality by 11%. The disadvantage of using the translation quality as a performance metric is the time that tuning program takes, which approaches 30 hours using the small corpus (2500 sentences) on a 2.4 GHZ processor and 1 GB memory computer. In case of using perplexity, the tuning process took 40 hours on the larger corpus (35000 sentences) and led to better results. It is recommended to use the perplexity as a measure as it needs less work in tuning the translation model.

In the future, we are going to investigate tuning GIZA++ for Phrase-based Statistical Machine Translation (PSMT). To build a flat phrase translation model, word alignments are used to extract bilingual phrase translations. GIZA++ produces the best alignment (called viterbi alignment) for each pair of sentences in the corpus beside the translation model. We believe that tuning GIZA++ parameters may yield better word alignments which, in turn, improve phrase based translation models.

Experimenting with the GA settings can be also investigated and suggested for future work.

7. Acknowledgement

This work is supported in part by a Collaboration Project on Statistical Machine Translation, between Information Science Institute, University of Southern California, and Computer Science Department, American University in Cairo. The project is funded by US-Egypt Science Board. The authors would like to thank professor Kevin Knight for his valuable advices.

8. References

[1] Franz Josef Och, (2002). Statistical Machine Translation: From Single-Word Models to Alignment Templates. http://www-mgi.informatik.rwth-aachen.de/Kolloquium/pastOberseminar.html

[2] Philipp Koehn., Franz Josef Och, Daniel Marcu, "Statistical Phrase-Based Translation", (2003). In Proceedings of the Human Language Technology Conference (HLT), pp. 127–133.

[3] Kenji Yamada and Kevin Knight, "A Decoder for Syntax-based Statistical MT", (2002). Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL), Philadelphia, July 2002, pp. 303-310.

[4] Philipp Koehn, "A Beam Search Decoder for Phrase-Based Statistical Machine Translation Models", (2003). A Technical Manual of the Pharaoh decoder.

[5] Kishore Papeneni, Salim Roukos, Todd Ward, (2001). BLEU: A Method for Automatic Evaluation of Machine Translation, IBM Research Report, RC22176

[6] Ahmed Ragab Nabhan and Ahmed Rafea, (2004). Tuning Statistical Machine Translation Parameters: 181-184

[7] Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, Robert L. Mercer, (1993). The Mathematics of Statistical Machine Translation: Parameter Estimation http://www.clsp.jhu.edu/ws99/projects/mt/ibmpaper.ps

[8] Stephan Vogel, Hermann Ney, Christoph Tillmann, (1996). HMM-Based Word Alignment in Statistical Translation. http://acl.ldc.upenn.edu/C/C96/C96-2141.pdf.

[9] Kevin Knight, (1999). A Statistical MT Tutorial Workbook. www.clsp.jhu.edu/ws99/projects/mt/mt-workbook.htm

[10] Yaser Al-Onaizan, JanCurin, Micheal Jahr, Kevin Knight, John Lafferty, Dan Melamed, Franz Josef Och, David Purdy, Noah A. Smith, David Yarowsky, (1999). Statistical Machine Translation Final Report. www.clsp.jhu.edu/ws99/projects/mt.

[11] Brian Harrington, (2003), Bagging and Boosting for Word Alignment. http://harringtonweb.com:8090/~brh/misc/

[12] Franz Josef Och, (2000). GIZA++ Readme File. http://wasserstoff.informatik.rwth-aachen.de/Colleagues/och/software/GIZA++.html

[13] The language modeling toolkit is available at:

ftp://ftp.cs.cmu.edu/project/fgdata/CMU_SLM/CMU_SLM_Toolkit_V1.0_release.tar.Z, (2005).

[14] The ISI ReWrite decoder is available at:

http://www.isi.edu/naturallanguage/software/decoder/index.html, (2005).

[15] Whittle is a pert of the statistical machine translation toolkit EGYPT. The tool is available at: http://www.clsp.jhu.edu/ws99/projects/mt/index.html, (2005).