Tuning Statistical Machine Translation Parameters[1]

Ahmed Ragab Nabhan and Ahmed Rafea

Abstract— Word alignment is the basis of statistical machine translation. GIZA++ is a popular tool for producing word alignments and translation models. It uses a set of parameters that affect the quality of word alignments and translation models. These parameters exist to overcome some problems such as overfitting. This paper addresses the problem of tuning GIZA++ parameter for better translation quality. The results show that our systematic procedure for parameter tuning can improve the translation quality.

Keywords— Parameter tuning, smoothing factors, training scheme, overfitting.

I. Introduction

GIZA++ is the most popular statistical machine translation toolkit available for public domain [1]. A team of researchers has developed this toolkit in order to boost the research in statistical machine translation [2]. GIZA++ is an extension of the program GIZA that has enhanced features and implements Hidden Markov Model (HMM).

In this paper, we investigate the key parameters that GIZA++ uses during training, and the impact on the training process. Moreover, we apply a systematic procedure for tuning these parameters on a limited corpus. We conducted experiments on GIZA++, a major landmark in statistical machine translation. The results show that our systematic procedure for tuning GIZA++ can improve the translation quality.

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 and discuss the function of key parameters of GIZA++. In section 4, we provide a methodology for parameter tuning. In section 5, we report our experiments and the results.

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

Ahmed Ragab Nabhan is with Mathematics Department, Faculty of Science, Fayoum branch, Cairo University, Egypt, (e-mail: ragab@claes.sci.eg).

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

[1] Proceedings of the International Conference on Computational Intelligence (ICCI 2004), Turkey, 2004

II. Background

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 [3]. Figure 1 shows a possible alignment between an Arabic sentence and an English sentence.

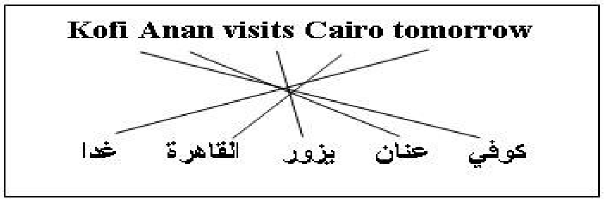

Figure1. An Alignment between two sentences.

Alignments are the basis on which translation models are 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.

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

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

Summing over ***a*** for both sides yields:

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

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

$$P(f \mid e) = \sum_a P(a, f \mid e) \qquad (3)$$

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

In order to estimate the probability ***P* (a, *f* | *e*),** Brown et al introduced a series of five statistical models each of which contributes to the calculation of p (a, f | e) [3]. Vogel et al introduced Hidden Markov Model (HMM) as an alternative to the standard Model 2 [4].

The Estimate Maximization (EM) algorithm is used to estimate the parameter values for the models [4], [5]. GIZA++ applies the EM algorithm on a parallel text to build the translation model in the form of a set of tables.

III. Motivation

GIZA++ has a set of parameters that affect the training process. These parameters have default values. However, these values should be adjusted for every new task [6]. The default parameter values may not be adequate for the training corpus we have. Some authors reported that a poor performance was achieved when using the default settings of GIZA++ [7]. These parameters exist to modify the training scheme, to achieve efficient training and to overcome some problems such as overfitting. The motivation beyond this work is to define a methodology to set these parameters. The following subsections describe how parameter settings affect overfitting and the training scheme used to build the translation model.

A. *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. Overfitting occurs because some of the relationships that appear statistically significant are actually noise (this happens with rare words in the corpus). To reduce the effect of overfitting, GIZA++ uses a number of smoothing parameters for various models. The purpose of smoothing factors is to smooth sharp probability values and decrease the gap between high probability values and low ones. In every 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 alignment probabilities:

$$P(aj \mid aj-1, I) = \alpha . I / I + (1-\alpha) . P(aj \mid aj-1, I) \quad (4)$$

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.

To turn off smoothing, we can set the smoothing parameter to zero. There are smoothing parameters for Models 2 through 4. The smoothing factor for model 3 is set to zero in the default settings of GIZA++, but adjusting the value of this parameter may increase the quality of the translation model.

B. *Training Scheme*

A training scheme specifies the sequence of used models and the number of training iterations used for each model [8]. For example, we may choose a training scheme that uses five iterations for Model 1 and three iterations for Model 2, three iterations for Model 3 and three iterations for Model 4. Another training scheme may use HMM instead of Model 2.

By default, GIZA++ uses HMM instead of Model 2. 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 [7]. In our experiments on Arabic–English translation system, the performance of Model 2 outperforms the HMM model. We believe that the number of iterations should be adjusted also. Convergence of various models may occur at less number of iterations than the default. We think that unnecessary extra iterations may lead to the trap of overfitting. 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.

IV. 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.

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. After tuning, we optimized the number of training operations for each model.

In this section we present the key parameters of GIZA++. Next, we present the performance metric we used in the optimization program. Then, we explain the algorithm of parameter tuning.

A. *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, probsutoff, probsmooth, and countincreasecutoff

4.2.2 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 and cutoff values.

B. Performance Metrics

There are many ways to measure the goodness of the translation model that GIZA++ produces. One way is to calculate the perplexity of the model. GIZA++ produces a file that summarizes perplexity values at the end of each training iteration.

Another way of evaluating translation model is measuring alignment quality by comparing viterbi alignments against reference alignments [9]. A viterbi alignment is the best alignment between a pair of sentences (the alignment with the highest probability value). In this case, a portion of the corpus has to be manually aligned. The quality of alignments is measured on the basis of alignment error rate (AER) [9]. These methods are typically used when the goal is to evaluate and compare various translation models. For more information about alignments evaluation methodology, see [9].

Since our ultimate goal is to improve translation quality, we tuned GIZA++ directly using the NIST score of the output translation of test data.

C. Parameter Tuning

We first tune general parameters. Then, optimize the parameters in an increasing model order in the training scheme.

We used GA for tuning real valued parameters rather than using AI searching algorithms such as Hill Climbing or Greedy search. The reason is that Genetic Algorithms have the advantage of not stopping at local optima and can perform well on noisy data. We think that GA is more suitable for the limited corpus we have. The proposed tuning methodology is adequate for small size data, since the cost of tuning GIZA++ to a larger corpus is high. It is suitable to tune GIZA++ using a portion of the corpus.

We used a population size of 16 members. Each member represents a valid value for the parameter. The first generation was initiated randomly. To get the fitness of each member, we run GIZA++ with the new parameter settings (the member value and previously tuned parameters). Next, we used the produced translation model in generating translations of a test sample. We used a standard baseline composed of GIZA++, CMU language modeling toolkit [10] and ISI ReWrite decoder [11]. We evaluate the translation using the NIST score [12]. Members with the highest NIST 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
    Measure translation quality using NIST metric
    Next j
3. For i:=1 to Num_of_optimization_runs do
   Choose the Best members(values) with the highest score
   Produce next generation using GA operators
       For each value j in the new generation do
           Run GIZA++ with the parameter value j
           Measure translation quality using NIST metric
       Next j
   Next i
 4. Report the member with the highest NIST score
    End.
```

V. Experiments and Results

We used a limited corpus of size 2500 sentence pairs in our experiments. We have two test sets: a tuning test set that is used during the optimization phase and a final test set to compare the tuned parameters effect against the default parameter setting. The domain of the corpus is a news domain. We performed five experiments each of which has certain objective.

The objective of the first experiment is to get the score that the default configuration of GIZA++ yields, in order to compare its performance against the optimized configuration. In this experiment, we ran GIZA++ with its default parameter settings. We use the resulting translation model to translate the test set data and measured the quality of translation using the NIST metric. The default parameter values of GIZA++ yields a score of 3.3948.

The objective of the second experiment is to measure the translation quality that the optimized parameter setting yields. We ran the tuning program on the training corpus and the tuning test set. At the end of the tuning program, we get the optimized parameter values and ran GIZA++ with these values. We used the translation model to translate the test set. The optimized parameter values yield a NIST score of 3.9490.

The objective of the third and fourth experiment is to compare the effect of parameter tuning against the effect of using linguistic resources such as dictionaries and lemmatizers. In both experiments, we augmented the corpus by adding bilingual dictionary entries for source and target vocabulary into the training corpus. The trick was proposed by [9]. The idea is to boost training by adding dictionary entries to the training corpus. The one-to-one correspondence between dictionary words enhances the alignments for words that are in the dictionary and besides it improves the alignments for neighboring words that are not in the dictionary.

In the third experiment, we ran GIZA++ with the default parameter settings on the augmented corpus. Then, we measure the performance on test data. The NIST score for this experiment was 3.8170.

In the fourth experiment, we ran GIZA++ with the optimized parameter values we get in experiment two on the augmented corpus. The NIST score achieved by this experiment was 4.061.

The objective of the fifth experiment was to test the effect of parameter tuning in the presence of linguistic resources such as dictionaries. In this experiment, we ran the tuning program on the augmented corpus. We take the optimized parameter values and ran GIZA++ with it. Then we used the translation model to translate the test set. We measured the translation quality and the NIST score was 4.2073.

The procedure we used for parameter tuning enhances the translation quality. The results show that parameter tuning is needed even with using linguistic resources.

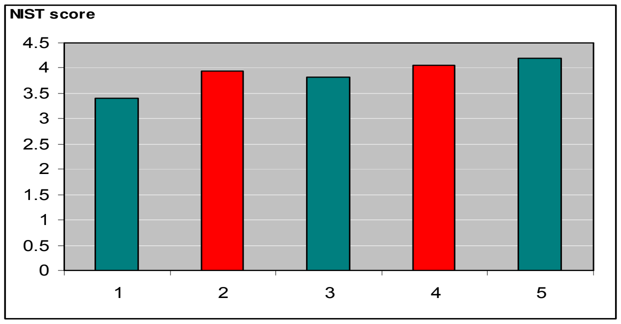

Figure 2. NIST scores for experiments 1-5

VI. Acknowledgement

The authors would like to thank Prof. Kevin Knight, Dr. Franz Och, and Dr. Alexander Fraser for suggesting tuning GIZA++. We also thank Dr. Harrington and Chris Callison-Burch for their valuable advices.

VII. References

[1] Franz Josef Och, Hermann Ney. (2000b), Improved Statistical Alignments Models. Proc. of the 38th Annual Meeting of the Association for Computational Linguistics, pp. 440-447, Hong Kong, China, 2000.

[2] 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

[3] 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

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

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

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

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

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

[9] Franz Josef Och, Hermann Ney. (2000a), A Comparison of Alignment Models for Statistical Machine Translation. In Proceedings of the 18th International Conference on Computational Linguistics, Saarbrucken, Germany, July. http://citeseer.ist.psu.edu/och00comparison.html

[10] The language modeling toolkit is available at: ftp://ftp.cs.cmu.edu/project/fgdata/CMU_SLM/CMU_SLM_Toolkit_V1.0_release.tar.Z

[11] The ISI ReWrite decoder is available at: http://www.isi.edu/naturallanguage/software/decoder/index.html

[12] The NIST scoring software is available at: http://www.nist.gov/speech/tests/mt/resources/scoring.htm