Analysis Different Automatic Evaluation Algorithm for Machine Translation

Fan Liu, Shanghao Chen, Shuo Huang, Yingxiu Chen
School of Computing Science, Simon Fraser University
Burnaby, BC, Canada V5A1S6
{liufanl, sca243, sha185, yingxiuc}@sfu.ca

Abstract

This paper presents the final project of CMPT825, which aims to improve the accuracy of automatic evaluation of machine translations. We experimented a group of machine translation evaluation algorithms, including METEOR, BLEU, GLEU, NIST, ROSE, and LEPOR, and tested their result on the local computer and Leaderboard. After experimenting with different algorithms and comparing a set of different parameters, we concluded the best results of each algorithm. In the end, we introduced the possible ways to improve the accuracy of different algorithms.

1 Introduction

In recent years, machine translation is a cutting-edge topic in the Artificial Intelligence research, in the meanwhile, the development of automatic evaluation of machine translations with high accuracy becomes another crucial challenge. In this project, we were given a database including more than 25,000 records, each record consist of two machine translations as hypothesis sentence and one human translation as reference sentence. We developed an automatic evaluation system based on this database, and recorded the competition result of two machine translations as -1 or 1 if one of them is better than the other one, or 0 if they are equally good. Finally, we compared the output of our automatic evaluation system with the given true result to measure its accuracy. In addition, we used another larger given database, which includes more than 50,000 records, to train several models for the system.

2 Motivation

Among all algorithms we chose for this project, METEOR metric is the simplest one, and it generated a test score of 0.538 using the algorithm mentioned in the Agarwal and Lavie's article (Agarwal and Lavie, 2007). To improve the performance of evaluator, we tested some variants of METEOR, such as BLEU, GLEU, NIST, and LEPOR. The reason for using these metrics is that METEOR is a word-to-word method which means it only matches unigrams, while the other metrics we selected have the capacity to handle *n-gram* word tokens. This would be an advantage compared to Meteor metric and is very likely to improve the accuracy. The metrics mentioned above are all heuristic methods, as a comparison, we also tried to use a supervised method, Rose. On the basis of Song and Cohn's research, it had been proved to have a better result than heuris-

tic methods (Song and Cohn, 2011).

3 Approach

In this project, we experimented METEOR, BLEU, GLEU, NIST, LEPOR, and ROSE to improve the accuracy of automatic evaluation, the details of each method are explained in the following section.

3.1 METEOR

METEOR (Agarwal and Lavie, 2008) is based on the harmonic mean of unigram precision and recall. For a pair of hypotheses sentence *h* and reference sentences *r*, we counted the words that occurred in both sentences and defined this number as $|h \cap e|$, then this number is used computed the unigram precision *P(h, e)* and unigram recall *R(h, e)* with following formula:

$$R(h,e) = \frac{|h \cap e|}{|e|} \tag{1}$$

and

$$P(h,e) = \frac{|h \cap e|}{|h|} \tag{2}$$

where $|e|$ and $|h|$ are the number of unigram words in the hypotheses sentences *h* and reference sentences *r*, respectively.

To find more accurate $|h \cap e|$, we used a mapping vector to keep track of all the words that have been mapped from the following three types of word matches:

Exact Match: Only those words that exactly the same in the two sentences were tagged as matched and recorded in the mapping vector.

Stemmed Match: The words that were not exactly matched in the hypotheses sentence and reference sentence were stemmed for stemmed matching. We used the Porter Stemmer algorithm (Martin Porter, 2006) from the NLTK library for the purpose of stemming the word. The mapping vector was updated if previously unmatched words are matched after they are stemmed.

Synonym Match: The words that were left out after an exact match and stemmed watch were considered for synonym match. We used WordNet *synset* library from the NLTK package to find all the synonyms for every unmatched word in the reference sentence. Then for every remaining unmatched word in the hypotheses sentence, we mapped it to the reference sentence if it is a synonym of a reference sentence word, and updated the mapping vector.

After mapping all possible matching words, we found the minimum number of matched words chunks in the hypothesis sentence which are in the same word order as the reference sentence. The number of chunks *c* and the number of unigram matches *m* then are used to compute the chunk penalty in the harmonic mean of precision and recall with following formula:

$$\ell(h,e) = \left(1 - \gamma \left(\frac{c}{m}\right)^{\beta}\right) \times \frac{P(h,e) \cdot R(h,e)}{(1-\alpha) R(h,e) + \alpha P(h,e)} \tag{3}$$

where α, β, γ are tunable parameters. We experimented the system by tuning those values and found the following combination generates the best results:

α	β	γ	**Dev Score**	**Test Score**
0.73	1.00	0.21	0.523	0.538

Table 1: Best METEOR Result

3.2 BLEU

BLEU (Papineni and Roukos, 2002) adopts the method of comparing and counting the number

of co-occurrence *n-grams*. It counts the number of *n-grams* words appearing in the hypothesis sentences and the reference sentences at the same time, and then gets the evaluation results by dividing the number of matching *n-grams* by the number of words translated to. The computation formula for *n-grams* precision on a multi-sentence test set is p_n as following:

$$p_n = \frac{\sum_{C \in \{Cand\}} \sum_{n-gram \in C} C_{clip}(n)}{\sum_{C' \in \{Cand\}} \sum_{n-gram' \in C'} C(n')} \quad (4)$$

C_{clip} in the above formula means that one truncates each words count. The computation formula is as follows:

$$C_{clip} = min(Count, MaxRefCount) \quad (5)$$

Next, let *c* be the length of the candidate translation and *r* be the effective reference corpus length. We compute the brevity penalty BP as following formula:

$$BP = \begin{cases} 1 & \text{if } c > r \\ e^{(1-\frac{r}{c})} & \text{if } c \leq r \end{cases} \quad (6)$$

Then, we get the computation formula for BLEU score as follows:

$$\log \text{BLEU} = \min\left(1 - \frac{r}{c},\ 0\right) + \sum_{n=1}^{N} w_n \cdot \log p_n \quad (7)$$

w_n means the weight matrix for *n-gram*. In our best results for *BLEU*, we only use unigram and bigram. In the end, we use additive smoothing to improve BLEU. The computation formula is as follows:

$$\textit{n-gram} = \begin{cases} \frac{\epsilon}{C(n')} & C_{clip}(n) = 0 \\ 0 & C(n') = 0 \\ \frac{Cclip(n)}{C(n')} & otherwise \end{cases} \quad (8)$$

We experimented with *n-gram* weights and ϵ to find the following combination to be the best for our results:

$[w_1, w_2]$	ϵ	**Dev Score**	**Test Score**
[0.92, 0.08]	0.10	0.52	0.536

Table 2: Best BLEU Result

3.3 GLEU

GLEU score is an alternative measure to BLEU score. According to the article (Wu and Schuster, 2016), BLEU score may not perform good with sentences, while GLEU score does not have the drawback. GLEU records all 1 to *n-grams* of reference and hypothesis, then calculates recall and precision value of matched *n-grams* between reference and hypothesis. More specifically, the recall is the ratio of the number of matched *n-grams* to the number of *n-grams* in reference sentence and the precision is the ratio of the number of matched *n-grams* to the number of *n-grams* in hypothesis sentence. The GLEU score takes the lower value between recall and precision.

N-gram	**Dev Score**	**Test Score**
2	0.51	0.527

Table 3: Best GLEU Result.

3.4 NIST

NIST score is another alternative measure to BLEU score. George introduced the differences between NIST and BLEU metrics in his article (George Doddington, 2002) that BLEU calculates the geometric mean of the overlapped *n-grams*, while NIST calculates weighted mean of the overlapped *n-grams*. NIST gives higher weights to those *n-grams* that occur less frequently, which means those *n-grams* are more informative. The weights of NIST is calculated using the Equation 9.

$$\textit{info}(w_1...w_n) = \log_2 \frac{\#\ of\ w_1...w_{n-1}}{\#\ of\ w_1...w_n} \quad (9)$$

Besides, NIST has a different brevity penalty calculated in Equation 10.

$$BP = \exp\left\{\beta \log^2 \left[\min\left(\frac{L_{hyp}}{\overline{L}_{ref}}, 1\right)\right]\right\} \quad (10)$$

Where the β is used to make the brevity penalty factor equal to 0.5 when the number of words in the hypothesis sentence is 2/3 of the average number of words in the references sentence. L_{hyp} is the length of hypothesis sentence and $\overline{L}_{ref}$ is the average length of references sentence. And the NIST score is calculated in Equation 11.

$$score = BP\times \sum_{n=1}^{N}\left\{\frac{\sum_{w_1...w_n} info(w_1...w_n)}{\#\ of\ w_1...w_{n-1}\ in\ h}\right\} \quad (11)$$

From above computation formula, we get the best NIST result as shown in Table 4.

3.5 ROSE

ROSE is short for Regression and Ranking based Optimisation for Sentence-Level Machine Translation Evaluation. It trains a model based on human judgments using supervised learning. Combined precision, recall, f-mean for sentence and POS, ROSE looks like BLEU, yet achieves sentence-level accuracy and higher correlation with human judgments. ROSE sentence features as listed in Table 5 (Song and Cohn, 2011) have *n-gram* (1-4) precision, recall, f-mean for words and POS, averaged *n-gram* precision, different words count, and *n-gram* mixed precision precision. Then using sentence features, we trained models using Support Vector Machine (SVM) with different kernels ('linear', 'poly', 'rbf'), RandomForestClassifier, and GradientBoostingClassifier.

Dev Score	Test Score
0.46	0.491

Table 4: Best NIST Result.

Index	Description
1-4	n-gram precision, n=1...4
5-8	n-gram recall, n=1...4
9-12	n-gram f-measure, n=1...4
13	average n-gram precision
14	words count
15	function words (stopwords) counts
16	punctuation count
17	content words count
18-21	n-gram POS precision, n=1...4
22-25	n-gram POS recall, n=1...4
26-29	n-gram POS f-measure, n=1...4
30-33	n-gram POS string mixed precision, n=1...4

Table 5: ROSE Sentence Features

pre-processing: Before calculating those features, training needs to be processed utf-8 encoded, lowercased and tokenized. Such that "Hello, word!" is then "hello, world !". In this

way, different translations, especially reordering translation can map better.

training and validating: We split training corpus into five parts, and trained on the first four parts, validated on the 5th part.

Below table records different scores for various models. For GradientBoosting and RandomForest, *d* in brackets means max-depth. "Dev 1/-1 Score" is the score provided by score-evaluation.py, 1 for using training data label in file "dev.answers" directly, -1 for using the opposite number of those labels.

Model	Train Score	Dev 1/-1 Score
linear SVM	0.523	0.521 / 0.520
poly SVM	0.435	0.315 / 0.429
rbf SVM	0.520	0.326 / 0.520
Gradient Boosting Classifier(d=5)	0.308	0.308 / 0.543
Gradient Boosting Classifier(d=10)	0.501	0.294 / 0.597
Random Forest Classifier(d=10)	0.510	0.315 / 0.538
Random Forest Classifier(d=100)	0.516	0.311 / 0.582

Table 6: Rose Results

3.6 LEPOR

LEPOR (Aaron and Derek, 2012) is designed with the factors of enhanced length penalty, precision, *n-gram* word order penalty, and recall. It provides a simple formula to compute it scores as follows:

$$\begin{aligned} \textit{LEPOR} =& LP \times NPosPenal \\ & \times Harmonic(\alpha R, \beta P) \end{aligned} \tag{12}$$

We only use length penalty, which is defined to embrace the penalty for both longer and shorter system outputs compared with reference translations, and it is calculated as follows:

$$LP = \begin{cases} e^{(1-\frac{r}{c})} & \text{if } c < r \\ 1 & \text{if } c = r \\ e^{(1-\frac{c}{r})} & \text{if } c > r \end{cases} \tag{13}$$

The last formula we used to combine with *n-gram* to calculate the score of the hypothesis sentence:

$$\begin{aligned} Score =& \alpha * lepor + \beta * w \\ & * (n - gram - recall) \end{aligned} \tag{14}$$

We experimented different parameters to best results and got the following best parameters. We get best results that only used unigram, bigram, and trigram. The parameters are as following:

α	β	w	Dev Score	Test Score
0.64	0.36	[1,1,1]	0.524	0.546

Table 7: Best LEPOR Result

4 Data

In most metrics, we used the given data file hyp1-hyp2-ref as the input file to calculate scores of each hypothesis sentence. Besides that, we also downloaded module "porter" from nltk.stem library to use Porter stemming algorithm and an external database WordNet from NLTK library in order to look up the synonyms of words.

In ROSE, hyp1-hyp2-ref and dev.answers are used as test data. train-test.hyp1-hyp2-ref and train.gold are used as training data. Besides, ROSE uses NLTK POS tag package to get the

n-gram POS precision, recall, and f-measure. In the training data, the first 20,000 records are used as training data; reserved 6,208 records are used as validation data. And because of these two files, first 20 rounds validation result looks reasonable, yet poor on test data. Thus, we use the opposite number of train labels to train another set of models. All 18 results are listed in Table 6.

5 Code

5.1 External Code

In this project, we referred Porter Stemmer and WordNet form the NLTK library to do the mapping. The Porter Stemmer algorithm is a process for removing the commoner morphological and inflexional ending from words in English. Its source code can be retrieved from NLTK homepage.

WordNet is a large lexical database for the English language, which was created by Princeton. Nouns, verbs, adjectives, and adverbs are grouped into sets of cognitive synonyms. The source code can be downloaded from WordNet homepage.

During the implement of LEPOR metric, we came up with some ideas and change some codes from the following open source projects in the GitHub which can be retrieved from https://github.com/aaronlifenghan and https://github.com/dhruvils.

5.2 Reused Code

We reused the *n-gram* algorithms code from our homework1, which help us to calculate the *n-gram* accuracy and recall. Then, part of code from homework2 also be reused for chunking. In this project, many algorithms are modified and combined by previous algorithms.

6 Experimental Setup

6.1 Sentence Preprocess

Before the implementation of further evaluation methods, we conducted preprocess works on the input sentences. For better word matches, we converted all text into lower case, then from the results, we observed that the word the was not helping to determine the quality of hypothesis sentence, therefore we remove it from the text. We also stripped all punctuation for better matches. For ROSE, the sentence is tokenized before calculating n-gram for word and POS.

6.2 Translation Evaluate

We compared the performance of different evaluation metrics. We have implemented two different kinds of evaluators that are the heuristic method and supervised method.

Heuristic methods including METEOR, BLEU, GLEU, NIST, and LEPOR calculate similar between translation sentences and reference sentences and vote the translation sentence with a higher score. To improve these models, we tried different parameter combinations to get best results. According to results in article (Agarwal and Lavie, 2008), BLEU model result can be improved by extending the exact matched words, so we used the stemmed math and synonym match idea in METEOR and LEPOR metrics to uniform these flexible matched words in translation and reference to improve the results in BLEU, GLEU, and NIST models.

The supervised method is ROSE, we tested with different models including SVM, RandomForest Classifier, and Gradient Boosting Classifier with various params. Results on split training data and dev data look in consist though. On training data, the more complex model we trained, the higher accuracy we get.

Model	Dev Score	Test Score
METEOR	0.52	0.538
BLEU	0.52	0.536
GLEU	0.51	0.527
NIST	0.46	0.491
LEPOR	0.52	0.546
ROSE	0.61	0.63

Table 8: Best Results Comparison with Different Algorithm

On dev data, the more complex the lower.

7 Results

Table 8 shows the best results we get from the testing data based on each metrics. It shows that ROSE gets the highest score on our project.

8 Analysis of the Results

From the computation formula, METEOR, BLEU, NIST, GLEU are very similar. The BLEU is based on the idea is that "the closer a machine translation is to a professional human translation, the better it is". GLEU is based on the smaller *n-gram* recall and precision. It makes sense that there are not much difference score on METEOR, BLEU, and GLEU. The NIST metric is based on the BLEU metric, but with some alterations. Since BLEU simply calculates *n-gram* precision with different weights to each one, NIST also calculates how informative a particular *n-gram* is. From the reference paper, the results should not have many differences between BLEU and NIST. However, the research suggests using no less than one sentence segments when evaluating corpus. And we also find the test score keeps increasing when *n-gram* increases. So we guess NIST may perform better to evaluate document translation than sentence translation.

The LEPOR methods get a higher score compare to former 4 methods since it combines many different evaluation methods.

In Table 8, we find ROSE has the highest accuracy compared to other metrics. This result is reasonable that it has been proved the ROSE method performs better than heuristic methods according to article (Song and Cohn, 2011).

For tests running using ROSE with training and dev data, we found a peculiar thing. The more data points included and the more complex model we set up, the accuracy of test data is higher, yet the dev accuracy is lower. So when predicting dev data, we time the prediction category with -1. And get higher accuracy. That's how we achieve the highest score in ROSE. So we think the labeling logic contradicts.

9 Future Work

9.1 Word Match

Lemmatization is another type of word match that worth to be considered. Compared to stemming, which usually refers to a heuristic process to remove the derivational affixes, lemmatization uses vocabulary and morphological analysis of words to remove inflectional endings only and to return the base or dictionary form of a word, thus lemmatization generates a more reasonable result.

9.2 Smoothing

We didn't do much smoothing on METEOR, BLEU, GLEU, NIST, ROSE, LEPOR. We can improve over algorithm by testing different smoothing method.

9.3 Parameter Adjustment

We can definitely improve LEPOR by adjusting parameters since we didn't have enough time

do enough test and research. We found a good material about LEPOR can be retrieved from https://arxiv.org/pdf/1703.08748.pdf

9.4 Synonym on Rose

In ROSE, *n-gram* statistics are exactly matched, synonym is not considered. This can be done to use GoogleNet to calculate word vector and setup synonym threshold, and then find the nearest word below that threshold.

9.5 XGBoost

XGBoost is kind of combination RandomForest and GradientBoosting, and most of the cases, combine models will bring benefits.

References

A. Agarwal and A. Lavie. (2007). METEOR: An Automatic Metric for MT Evaluation with High Levels of Correlation with Human Judgments. *In Proceedings of the second the Second Workshop on Statistical Machine Translation*, Pages 288–231. Prague, Czech Republic.

Porter, Martin. (2006, January). *The Porter Stemming Algorithm*. Retrieved from https://tartarus.org/martin/PorterStemmer/.

Papineni, Kishore, Salim Roukos, Todd Ward and Wei-Jing Zhu. (2002). BLEU: a Method for Automatic Evaluation of Machine Translation. *In Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*,Pages 311-318 Philadelphia, Pennsylvania.

Xingyi Song and Trevor Cohn. (2011). Regression and Ranking based Optimisation for Sentence Level Machine Translation Evaluation. *IN Proceedings of the 6th Workshop on Statistical Machine Translation*, Pages 123–129. Edinburgh, Scotland

Y. Wu, M. Schuster, Z. Chen, Q. V.Le, M. Norouzi, W. Macherey, M. Krikun et al.. (2016). Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. *arXiv:1609.08144*.

George Doddington. (2002). Automatic Evaluation of Machine Translation Quality Using N-gram Co-Occurrence Statistics. *In Proceeding of the second international conference on Human Language Technology Research*, Pages 138–145. San Diego, California.

Han, Aaron Li–Feng, Derek F. WONG and Lidia S. CHAO. (2012). LEPOR: A Robust Evaluation Metric for Machine Translation with Augmented Factors. *In Proceedings of the 24th International Conference on Computational Linguistics*, Posters. Mumbai, India.

A. Agarwal and A. Lavie. (2008). METEOR, M-BLEU and M-TER: Evaluation Metrics for High-Correlation with Human Rankings of Machine Translation Output. *In Proceedings of the Third Workshop on Statistical Machine Translation*, Pages 115-118. Columbus, Ohio.