Measuring MT Quality using vector representations

Master's thesis
submitted by

YIRAN HUANG

at the Interactive Systems Lab
Institute for Anthropomatics and Robotics
Karlsruhe Institute of Technology(KIT)

Reviewer:	Prof. Dr. Alexander Waibel
Second reviewer:	Prof. Dr. Tamin Asfour
Advisor:	Dr. Jan Niehues

Process Period: 26.March.2018 - 25.September.2018

Karlsruher Institut für Technologie

Declaration

I hereby declare and confirm that this thesis is entirely the result of my own original work. Where other sources of information have been used, they have been indicated as such and properly acknowledged. I further declare that this or similar work has not been submitted for credit elsewhere.

Karlsruhe, September 25, 2018

Abstract

Human evaluation of the translation system is expensive. Automatic evaluation metric, which can be used to evaluate the translation system automatically is getting more and more critical. They are not only used to judge a translation system but also used to orient the training of the neural machine translation system nowadays. Concerning this, we design a new automatic evaluation metrics, which can be a better fit for this task. We take good use of the vector representation of the given sentence, either from the internal statuses of a machine translation system or from a Word2Vec(w2v) model, which is used to produce word embeddings, to judge its quality. Comparing with other metrics, the advantages of the metrics we introduced in this thesis are: contextual, it depends on the training data of the machine translation system; character base, so that it can deal with the words that not appeared in the vocabulary; deep network, neural machine translation contains many Long Short Term Memory (LSTM) layers in the decoder that can represent the meaning of a sentence in different aspects; vector representation, we represent the system output sentence with a vector; the last but not the least, the quality of the metric depends on the machine translation system that is used during the evaluation. The performance of the metrics we introduced here surpass the performance of the sentBLEU, which often used as baseline in the NLP task, in 'ACL First Conference on Machine Translation' in both direct assessment and relative ranking task. The result of our metrics are close to the State of the art(METRICS-F) in relative ranking task.

Zusammenfassung

Die menschliche Bewertung des Übersetzungssystems ist teuer. Automatische Bewertungsmetriken, mit denen das Translationssystem automatisch bewertet werden kann, werden immer kritischer. Sie werden nicht nur dazu verwendet, ein Übersetzungssystem zu beurteilen, sondern werden auch verwendet, um das Training des neuralen maschinellen Übersetzungssystems heutzutage zu orientieren. Hierzu entwerfen wir eine neue automatische Bewertungsmetrik, die für diese Aufgabe besser geeignet ist. Wir verwenden die Vektordarstellung des gegebenen Satzes entweder aus dem internen Status eines maschinellen Übersetzungssystems oder aus einem Word2Vec (w2v) -Modell, das zur Erzeugung von Worteinbettungen verwendet wird. Im Vergleich zu anderen Metriken sind die Vorteile der Metriken, die wir in dieser Arbeit eingeführt haben: kontextbezogen, abhängig von den Trainingsdaten des maschinellen Übersetzungssystems; Zeichen basiert, damit sie mit den Wörtern umgehen kann, die nicht im Vokabular vorkommen; tiefes Netzwerk, neuronale maschinelle Übersetzung enthält viele 'Long Short Term Memory'(LSTM) -Schichten im Decoder, die die Bedeutung eines Satzes in verschiedenen Aspekten darstellen können; Vektordarstellung, wir repräsentieren den Systemausgabesatz mit einem Vektor; zum Schluss hängt die Qualität der Metrik vom maschinellen Übersetzungssystem ab, das bei der Auswertung verwendet wird. Die Leistung der hier vorgestellten Metriken übertrifft die Leistung von sentBLEU, die häufig als Grundlage in der NLP-Aufgabe verwendet wird, in 'ACL First Conference on Machine Translation' sowohl in der direkten als auch in der relativen Ranging-Aufgabe.

Contents

Chapter 1

Introduce

1.1 Motivation

How to judge the quality of one translation?

The closer a machine translation is to a professional human translation, the better it is.

It is the central idea of evaluation metrics. Human evaluation is an extensive but expensive method that used to evaluate a translation. It weighs many aspects of the translation, including adequacy, fidelity, and fluency of the translation[10][22].
Comparing with human evaluation the primary bilingual evaluation understudy(BLEU) in 2002 regarded to be much more expedient. It compares n-grams of the candidate with the n-grams of the reference translation and uses the number of matches to judge the candidate, the more the better[16]. Nowadays it is still used as a baseline in many translation matches.
Based on BLEU, many new matrices are created, for example, character n-grams F-score(chrF). The general formula for the CHRF score is:

$$chrF_\beta = \frac{CHR_P \quad CHR_R}{\beta^2 \quad CHR_P + CHR_R} \tag{1.1}$$

where CHR_P is the percentage of n-grams in the hypothesis which has a counterpart in the reference, and CHR_F is the percentage of character n-grams in the reference which is also present in the hypothesis, and β is a parameter which assigns β times more important to recall than precision[17].
They are also other kinds of evaluation metric, like BEER. It trains a simple linear model exploiting 33 relatively dense features, including adequacy features, fluency features and Features based on PETs[19].
Comparing with metrics, machine translation system also develops quickly in the recent years. Early in the late 1980s, Time Delay Neural Network

(TDNN), a prime deep-learning approach, was proposed by Prof. Waibel for phoneme recognition[20]. In the 2000s, the recurrent neural networks (RNN)[21] and long-short-term-memory (LSTM)[8] are proven to be efficiently dealing with sequence-to-sequence tasks. Afterwards, the invention of the word embedding[15] solves the sparsity problem by learning a representative feature vector. Together with the recently proposed Attention mechanism[1] which enhances the power of coping with long sentences, the mature encoder-decoder paradigm has become the state-of-art solution for NMT.[4]
Seeing the development of the neural translation system, we come to an idea: can we use a vector to represent the sentence, and judge its quality throught this vector, for example, use the internal statuses in the encoder-decoder neural translation system to evaluate a sentence. It is also the core question in this thesis.

1.2 Basic Idea

We divide the model into two parts. For the inputted sentences we first transform them to some vectors with the help of word2vec model or neural machine translation system. Then we put these vectors in the second part of the model to get the result we want. We can get the vector representation for a sentence from a word2vec directly, which is the role of a word2vec model. But how to utilise the neural translation system to get the vector representation?
To translate a sentence with an encoder-decoder neural translation system, we put the source sentence as the input of the encoder. The input of the decoder should be the output of the decoder in the last timestamp. To utilise the neural translation system in our metrics model, same as usual translation, we put the source sentence as input of the encoder, but instead of using the output of the decoder in the last timestamp as the new input data, we use the translation candidate, that we want to score, as input of the decoder. Then we represent the candidate sentence with the internal statuses of the decoder. We use a neural network to judge the quality of the original sentence according to this representation.

Chapter 2

Background

In this chapter, we introduce some knowledge that essential to the model introduced in later chapters. In the first section, we focus our attention on the sentence embedding. We introduce different ways to represent a sentence inputted with a vector so that it can be used in a neural network or some other model directly. After that, we introduce the distance methods, which are used to calculate the distance between two vectors, which can dedicate the similarity of these two vectors. At last, we introduce an algorithm that can be used to optimise the category parameters in a model.

2.1 Sentence embedding

In this section, we introduce different methods to do the sentence embedding. Generally, for each word in one sentence, we first transform it to a vector with a word embedding method and then we use an aggregation function to aggregate all these vector representations from the same sentence and use the output vector as the representation of the sentence.

The word embedding methods used here can be roughly distributed into two types, word2Vec and vector representation learned in neural machine translation system(NMT). We introduce these methods concretely in the following subsection.

2.1.1 Word2Vec

The concept Word2vec introduced here is a tool provided by Google. It takes a text corpus as input and construct a vocabulary from these texts and then learns the vector representation of each word through training of a neural network. Concrete processes are introduced in the articles: [14] [15].

We use the word2vec model to get the word embeddings firstly and then combine the words embeddings in the same sentence with an aggregation function and use it as the representation of that sentence.

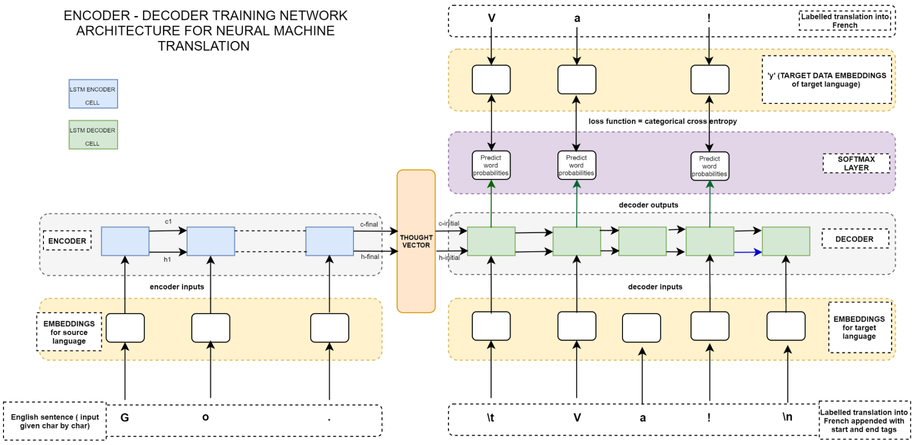
ENCODER - DECODER TRAINING NETWORK ARCHITECTURE FOR NEURAL MACHINE TRANSLATION
LSTM ENCODER CELL
LSTM DECODER CELL
ENCODER
c1
h1
c-final
h-final
THOUGHT VECTOR
encoder inputs
EMBEDDINGS for source language
English sentence (input given char by char)
G
o
.
V
a
!
Labelled translation into French
'y' (TARGET DATA EMBEDDINGS of target language)
loss function = categorical cross entropy
Predict word probabilities
Predict word probabilities
Predict word probabilities
SOFTMAX LAYER
decoder outputs
c-initial
h-initial
DECODER
decoder inputs
EMBEDDINGS for target language
\t
V
a
!
\n
Labelled translation into French appended with start and end tags

Figure 2.1: Encoder-decoder architecture[11]

$$
\begin{aligned}
i_t &= \sigma(W_{ii}x_t + b_{ii} + W_{hi}h_{(t-1)} + b_{hi}) \\
f_t &= \sigma(W_{if}x_t + b_{if} + W_{hf}h_{(t-1)} + b_{hf}) \\
g_t &= \tanh(W_{ig}x_t + b_{ig} + W_{hg}h_{(t-1)} + b_{hg}) \\
o_t &= \sigma(W_{io}x_t + b_{io} + W_{ho}h_{(t-1)} + b_{ho}) \\
c_t &= f_t c_{(t-1)} + i_t g_t \\
h_t &= o_t \tanh(c_t)
\end{aligned}
$$

Figure 2.2: Long short term architecture[11]

Here we use the pre-trained model directly. There are two different word2Vec models used in the experiment:

- *GoogleNews-vectors-negative300* (named w2v in the experiment result table) This model is trained on parts of Google news dataset(about 100 billion words). It contains 3 million words and phrases. It represents each word or phrase with a 300 dimensions vector.
- *GoogleNews-vectors-negative300-slim* (named w2v-slim in the experiment result table) This model is a subset of the model above. The compressed file of the original model is 1.6GB, and it takes times to load the model with gensim. That is why we use the slimmer version here.

2.1.2 Vector presentation learned in NMT

Because most of the vector representation we used in the experiment come directly from the neural machine translation(NMT) system. Firstly we take a brief overview of the neural machine translation system. More specifically, the encoder-decoder architecture in neural machine translation system. The first step of any neural machine translation system is to convert each atomic symbol into a corresponding continuous vector(500 dimension vector here), which is often called as a word embedding. This step is done for each source word independently of the other words and results in a source sequence of word embeddings. The encoder network which is implemented as a bi-direction long short-term memory network with two hidden layers, it encodes this source sentence into a single context vector.

The decoder network, again the long short-term memory network, generates a translation word-by-word while being conditioned on the context representation of the source sentence. At each step of generation in the decoder, the internal hidden states of the decoder are updated first and then combine the last layer hidden states with the attention information. The dot product between the output of the hidden layers and the output word embedding vector of each word in the target vocabulary is computed and normalised across all the target words, resulting in a probability distribution over the target vocabulary. A target word is selected based on this distribution, and the whole process is recursively repeated until the end-of-sequence symbol is generated. Details in [1]

Figure 1 (2.1) shows the encoder-decoder architecture. Figure 2 (2.2) shows the internal calculation in the Long Short-Term Memory network[8] where h_t is the hidden state at time t, c_t is the ceil states at time t. x is the input data. i_t, f_t, g_t, o_t are the input, forget, ceil, forget gates. σ is the sigmoid function. The long short-term memory network we use here, for each input data, output three different outputs including hidden states, ceil states and output states. The output states are almost the same as the hidden states. It is the output of a dropout layer[18] taking the hidden state as input.

According to this architecture, we summarise the following methods from the decoder side of a neural machine translation system to represent the words in the sentence.

- *NMT Decoder Embedding* It's the same as NMT Encoder Embedding, and it's the first step of the decoder part of the neural machine translation system. It also converts each atomic symbol into a vector.
- *NMT Decoder hidden states* It is one of the outputs of the long short-term memory network in the decoder. It represents the hidden states.
- *NMT Decoder hidden ceils* It is one of the outputs of the long short-term memory network in the decoder. It represents the ceil states
- *NMT Decoder ouptut* It is the output of the decoder.

So far, we only get the word embedding, to transform the word embeddings into a sentence embedding, we make use of the aggregation function. We aggregate the representations of all the words in one sentence to form a sentence representation. The aggregation function we use here include sum, max and mean. Besides, because the neural machine translation system we use here, contains two hidden layers. Therefore, for one candidate sentence, we get 18 different representations from the decoder side of the neural machine translation system. They are the core representations that are used in the experiments.

Besides the sentence representations from the decoder of the neural machine translation system, we also use the Encoder states in the neural translation system to represent the words and sentence. These states include:

- *Encoder hidden states* We use two layer bi-direction LSTM in the encoder side of the neural translation system, For each word inputted the hidden layers of the encoder output a vector. We use this vector to represent the word inputted. The encoder hidden layer of the neural translation system model we used here 500 dimension which means when we input a word into the encoder we get a 500 dimension vector in return. As describe before, then we use an aggregate function to transform the words embedding in one sentence into sentence embedding.
- *Encoder context value* We use the last output of the LSTM in the encoder side of the neural translation system to represent the sentence inputted.

2.2 Distance Measures

In this subsection, we introduce the distance methods that are used in the experiment. As mentioned above, we represent the candidate sentence with a vector, usually a 500 dimensions vector. For example we have two vectors here, we represent them with $(x_1, ..., x_n)$, $(y_1, ..., y_n)$. The distance methods take these vectors and input and output a score. We use this score to represent the similarity of the vectors inputted.

The formulas of the distance methods that are used in the experiments are as shown below.

- *L1(cityblock)*

$$L1 = \sum_i |x_i - y_i| \tag{2.1}$$

- *L2*

$$L2 = \sum_i \sqrt{x_i^2 - y_i^2} \tag{2.2}$$

- *Cosine similarity*

$$cos = \frac{\sum_i x_i y_i}{\sqrt{\sum_i x_i^2}\sqrt{\sum_i y_i^2}} \tag{2.3}$$

- *Braycurtis*[5]

$$braycurtis = \frac{\sum_i x_i - y_i}{\sum_i x_i + y_i} \tag{2.4}$$

- *Canberra*[9]

$$canberra = \sum_i \frac{|x_i - v_i|}{|x_i| + |y_i|} \tag{2.5}$$

- *Pearson correlation*[2]

$$corr = 1 - pearson\_correlatioin \tag{2.6}$$

- *Mahalanobis* where V is the covariance matrix[6]

$$mahalanobis = \sqrt{(x - y)V^{-1}(x - y)} \tag{2.7}$$

- *Standard euclidean* where V is the covariance matrix

$$seuclidean = \sqrt{(x - E(x))^T V^{-1}(y - E(y))} \tag{2.8}$$

2.3 Quality of Metrics

Before we can create a new metric by ourselves, we need to look for a method to measure the goodness of a metrics. In the metrics task of WMT, two different methods are used according to the type of data they are using.

- *Direct Assessment* In this case, we use Pearson correlation to measure the quality of a metric. Generally, the direct assessment dataset contains machine translator output and human scores. We use our metric to scores the output sentence and calculate the Pearson correlation between the result and the human scores. The larger the correlation, the better the metric is. To compute pearson correlation we make use of the Formula

  $$\rho_{X,Y} = \frac{E(X - \mu_X)(Y - \mu_Y)}{\sigma_X \sigma_Y} \tag{2.9}$$

  In the formular, μ_X is the mean of the X, σ_X is the standard diviation of X and E is the expectation.

- *Relative Ranking* In this case, we use Kendall's Tau-like formula to measure the quality of a metric. It is a variant of Kendall's Tau coefficient[7]. Because we don't have the ranking data of the whole test set, we can't use Kendall's Tau coefficient. To compute Kendall's Tau-like coefficient we use the following matrix(see Table 2.1) and Formula [13].

  $$\tau = \frac{\sum_{h,m \quad \{<,=,>\}, C_{h,m}=0} C_{h,m}|S_{h,m}|}{\sum_{h,m \quad \{<,=,>\}, C_{h,m}=0} |S_{h,m}} \tag{2.10}$$

  In the formula, we read the C value direct from the matrix. $|S_{h,m}|$ is the size of the specified data set. For example, $|S_{<,>}|$ is the size of the data set, which human assessment is '<' but the metrics assessment is '>'.

		metric		
		<	=	>
human	<	1	0	-1
	=	X	X	X
	>	-1	0	1

Table 2.1: The matrix that use to calculate Kendall’s Tau like coefficient. ’<’ means sentence from translator one is worse than the sentence from translator two. ’=’ means sentence from translator one is better than the sentence from translator two. ’=’ means the qualities of the sentences from both translators are the same

Chapter 3

Related work

In recent year, automatic machine translation evaluation has received much attention in recent year. Various metrics have been proposed with the aim to provide quick and stable measurements of the performance of a machine translation system. Most of them compute the similarity between the machine translation hypothesis and the reference translation. However, difference metric based on different perspectives regarding measuring similarity, for example, BELU metric is based on lexical. However, Maxsim metric is based on syntactic.

3.1 BLEU

Bilingual evaluation understudy(BLEU) is an algorithm used to evaluate the quality of a text translated from a translator. Given a translation sentence and a reference sentence, the BLEU metric output a score between 0 and 1, which indicate the similarity between the given sentences. The larger is the output, the better is the translation sentence.
BLEU make use of the so-called modified n-gram precision to calculate the similarity between the inputted sentences. An n-gram is a segment of consecutive n words in a sentence. A sentence with 18 words has 18 1-gram and 17 2-gram. Precision is the probability of an n-gram in a candidate sentence found in the reference sentence, for example:
Example 1
Candidate 1: It is a guide to action which ensures that the military always obeys the commands of the party.
Candidate 2: It is to insure the troops forever hearing the activity guidebook that party direct.
Reference 1: It is a guide to action that ensures that the military will forever heed Party commands.
Reference 2: It is the guiding principle which guarantees the military forces always being under the command of the Party.

Reference 3: It is the practical guide for the army always to heed the directions of the party.

There are 18 words in the sentence candidate 1. 17 of them appear in the reference sentences. So we can say the precision of 1-gram is 17/18. For the same reason, the precision of 2-gram is 10/17.

It looks good here, but precision has a shortcoming: the words in the reference sentence may be reused. See the example below:

Example 2

Candidate: the the the the the the the.

Reference 1: The cat is on the mat.

Reference 2: There is a cat on the mat.

The precision for 1-gram is 7/7. But we know that the candidate sentence is bad.

To overcome this problem, BLEU uses modified n-gram precision. The modified n-gram precision calculate a value $m_{max}$for each word in the candidate sentence, by taking its maximum total count, in any of the reference sentences. For example, in example 2, the word 'the' appears twice in the reference 1 sentence, and once in the reference 2 sentences. So the m_{max} is 2 for the word 'the' in the candidate sentence. For each word in the sentence BLEU calculates the m_{max}, and then BLEU sum over all the m_{max} in the candidate sentence. This sum is then divided by the total number of 1-gram in the candidate sentence. It is the modified 1-gram precision for this sentence.

There are still other problems with BLEU scores. One of them is that they tend to favour short translations, which can produce very high precision scores, even using modified precision.

3.2 METEOR

METEOR is the abbreviation of 'Metric for Evaluation of Translation with Explicit ORdering'. The metric is designed for the evaluation of machine translation output in 2014 to fix some problems found in the more popular BLEU metric. To understand the design of METERO metric, some concepts need to explain first.

Harmonic mean

There are three different kinds of means in the classical Pythagorean means: the arithmetic mean, the geometric mean and the harmonic mean. Typically, Harmonic mean is appropriate for situations when the average of rates is desired, it can be expressed as the multiplicative inverse of the arithmetic mean of the multiplicative inverse of the given set of observations. The equation of the arithmetic mean are shown below:

$$a_{mean} = \frac{1}{n}\sum_{i=1}^{n} a_i = \frac{a_1 + a_2 + ... + a_n}{n} \tag{3.1}$$

Corresponding to the arithmetic mean, the harmonic mean is:

$$h_{mean} = (\frac{1}{n}\sum_{i=1}^{n} a_i^{-1})^{-1} \tag{3.2}$$

Unigram precision and recall

As with BLEU, the algorithm first creates an alignment between two sentences with constraint: the candidate translation sentence and the reference translation sentence. The alignment is a set of mappings between unigrams. A mapping can be described as a line between the same unigram in two sentences. There is a constraint for the mappings. If there is more than one alignment with the same number of mappings, the alignment is chosen with fewest intersections of two mappings.

Once the final alignment is computed, the unigram precision P and unigram recall R are calculated as:

$$P = \frac{m}{w_t} \tag{3.3}$$

$$R = \frac{m}{w_r} \tag{3.4}$$

Where m is the number of mappings between the candidate translation and the reference translation, w_t is the number of unigrams in the candidate translation and w_r is the number of unigrams in the reference translation.

We then calculate the harmonic mean of the precision and recall with recall weighted 9 times more than precision:

$$F_{mean} = \frac{10PR}{R + 9P} \tag{3.5}$$

Penalty

So far we only concern the unigram, to take the longer n-gram into account, we use them to compute a penalty for the alignment. When we more mappings that are not adjacent in the reference and candidate sentence found, the higher the penalty will be.

To compute the penalty, unigrams are grouped into the fewest possible chunks, that contain a set of unigrams that are adjacent in the hypotheses and the reference. The longer the adjacent mappings between the candidate and the reference, the fewer chunks there are.

The penalty is computed as follows:

$$penalty = 0.5(\frac{c}{u_m})^3 \tag{3.6}$$

Where c is the number of chunks, and u_m is the number of unigrams that have been mapped.
The final for sentence is computed as below:

$$M = F_{mean}(1 - penalty) \tag{3.7}$$

3.3 BLEND

BLEND[12] is a combined machine translation metric which makes good use of the existing metrics. Contrary to other combined metric BLEN use SVM regression to train the model and use direct assessment scores as the golden standard.
Comparing with DPMFcomb[1] which use SVM-rank to train the model and use relative ranking as the golden standard, BLEND make a vast reduction in required training data and achieves improved performance over DPMF-comb when they incorporated the same metrics.
BLEND can be applied to any language pair if in-corporated metrics support the specific language pair.
Generally, BLEND is building upon scores provided by 25 lexical based metrics and 4 other metrics for to-English language pair. Since some lexical based metrics are merely different variants of the same metric, there are only 9 kinds of lexical based metrics, namely BLEU, NIST, GTM, METEOR, ROUGE, Ol, WER, TER and PER. 4 other metrics include CharacTer, BEER, DPMF and ENTF.[3]

[1] a combined metric proposed in 2015

Chapter 4

Data

4.1 Quality judgments

When we use a machine translator to translate a sentence, how to judge the quality of its output? In this section, we are going to introduce several ways to assess the output of a machine translation system that ware used in WMT(ACL First Conference on Machine Translation). They are also the methods that used in the quality judgment of the metric. The closer a metric score is to a human score, the better this metric is.

- *Direct Assessment(DA)* Instead of source sentence, the assessor compares a translation of a machine translator with the reference translation sentence of the same source sentence. Moreover, the assessor rates each machine translator output sentence with a score between 0 and 100. Direct Assessment is a new golden standard that was used in WMT. It has two advantages:
  - Monolingual knowledge is required
  - The similarity of the way to judge a sentence between human and machine is getting closer. Both of them compare the difference between the machine translator output and reference sentence.
- *Relative Ranking(RR)* In the task of DA, assessor only compare the machine translator output with the reference sentence, however in the RR task, the assessor have to compare more than one outputs from different machine translator (of the same source sentence) with not only the reference sentence but also its source sentence. Instead of rating an output with a score, the assessor here is asked to rank these outputs of the different machine translators according to their quality with ties[1] allow.
- *DaRR* When there are no relative ranking data, and the direct assessment data that we have collected are not enough to get a reliable

[1] two sentences from different machine translator get the same assess

	2017	2016	2015
de-en	3004	2999	2169
en-de	3004	2999	2169

Table 4.1: Number of Sentences that were used in the data set from 2013 to 2017 in WMT

judgment of a metrics. We can convert the direct assessment data to relative ranking data and use the judgment method for relative ranking data to assess the metrics. The only difference between the DaRR data and relative ranking data, we introduce above , is the way to get the data. To translate a DA data to DaRR, it should fulfil the following conditions:

- There are assessments for the more than one machine translators output sentences of the source sentence.
- We can judge that one translation sentence is better than the other according to their assessments.

4.2 Data Set

For the training and testing of the metric model, we use the direct assessment and relative ranking data that were used in WMT from 2015 to 2017.
The direct assessment data set consist of four parts:

- The source sentence
- The machine translator output sentence
- Reference sentence
- Human score

The relative ranking data set consist of five parts:

- The source sentence
- The machine translator output sentence one
- The machine translator output sentence two[2]
- Reference sentence
- human score

The number of source sentence that was used in WMT differs from year to year, they are shown in table 4.1.

[2] the output from a different machine translator of the same source sentence as machine translator output sentence one

dataset	type	language-pair	number
ad-deen-good-stnd	da	ger-eng	560
ad-seg-scores-de-en	da	ger-eng	24158
ad-seg-scores-en-de	da	eng-ger	7025

Table 4.2: Data sets that are collected in WMT17. 'type' column points out the quality judgment method, 'da' means direct assessment. In language-pair column, 'ger' is German and 'eng' is English.

4.2.1 WMT17

Three data sets are collected in WMT17. All of them are direct assessment data as shown in the table 4.2.

- *ad-deen-good-stnd.csv* Direct assessment scores of language pair German to English. They are created by taking the mean of a minimum of 15 assessments of the same machine translator output sentence. There are reliable to be used as a estimation of the quality of a translation
- *ad-seg-scores-de-en.csv* Direct assessment scores of language pair German to English. They are created by taking the mean of a maximum of 14 assessments of the same machine translation. They are not expect to provide an accurate reflection of the quality of the translation
- *ad-seg-scores-en-de.csv* Direct assessment scores of language pair English to German. They are created by taking the mean of a maximum of 14 assessments of the same machine translation. They are not expect to provide an accurate reflection of the quality of the translation

4.2.2 WMT16

Three data sets are collected in WMT16. One of them is direct assessment data set, and the others are relative ranking data set. Concrete descriptions are shown below.(see table 4.3):

- *DAseg.newstest2016.human.de-en* Direct assessment scores of language pair German to English. They are created by taking the mean of a minimum of 15 assessments of the same machine translator output sentence. There are reliable to be used as an estimation of the quality of a translation
- *wmt16-dump-20160610-1226.deu-eng.cs* Relative ranking data of language pair German to English.
- *wmt16-dump-20160610-1226.eng-deu.cs* Relative ranking data of language pair English to German.

dataset	type	lang-pair	number
DAseg.newstest2016.human	da	ger-eng	560
wmt16-dump-20160610-1226.deu-eng	rr	ger-eng	20937/17701
wmt16-dump-20160610-1226.eng-deu	rr	eng-ger	50989/39689

Table 4.3: Data sets that are collected in WMT16. Because for the same machine systems and sentence pair there exist multiple assessments in relative ranking data set, here two different numbers are recorded in the table. The first one is the original number of ranking in the data set and the second one is the number of ranking after combining the repeated items.

dataset	type	lang-pair	number
DAseg.newstest2015.human	da	ger-eng	500
judgements.20150817 part1	rr	ger-eng	40535/32856
judgements.20150817 part2	rr	eng-ger	54447/42936

Table 4.4: Data sets that are collected in WMT15. Because for the same machine systems and sentence pair there exist multiple assessments in relative ranking data set, here two different numbers are recorded in the table. The first one is the original number of ranking in the data set and the second one is the number of ranking after combining the repeated items.

4.2.3 WMT15

Two data sets are collected in WMT15. One of them is direct assessment data set, and the other is a relative ranking data set. Concrete descriptions are shown below.

- *DAseg.newstest2015.human.de-en* Direct assessment scores of language pair German to English. They are created by taking the mean of a minimum of 15 assessments of the same machine translator output sentence. There are reliable to be used as an estimation of the quality of a translation.
- *judgements.20150817* It includes both the relative ranking data of language pair English to German and the relative ranking data of language pair German to English

To have a bright look on the data set, we distribute the data into two parts according to the language pair as shown in table 4.4

Chapter 5

Model

5.1 Framework

We introduce the basic models in this chapter. Generally, we can divide the model into two parts as shown in the Figure 5.1. For the inputted sentences we first transform them to some vectors through 'Sentence embedding', After that we use the 'Combination model' to combine these vector to get the result. The type of the result depends on the type of the task: direct assessment or relative ranking.

- *Sentence embedding* Given a sentence, we present it with a vector. Sentence embedding does not depend on the type of the task. In the next section, we introduce different ways of sentence embedding.
- *Combination model* Taken the representations of the sentences as inputs, we compute a result through this model. Because the result is a combination of the inputted representations, we name this model 'combination model'. Combination model depends on the type of the task, direct assessment and relative ranking, because for the different task the input and output data are different. We introduce the framework of the combination models in the subsections separately according to the type of the task.

5.1.1 Direct assessment

The target of the direct assessment(DA) task is, given a system output translation and a reference translation, score the system output translation

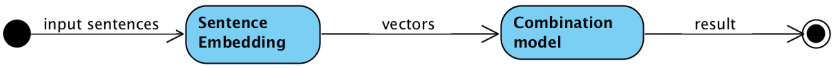

Figure 5.1: General framework

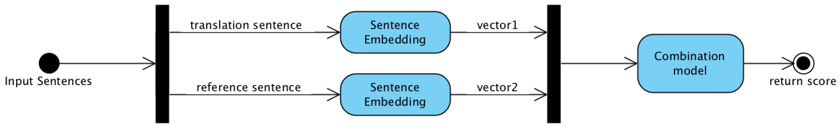

Figure 5.2: Direct assessment framework

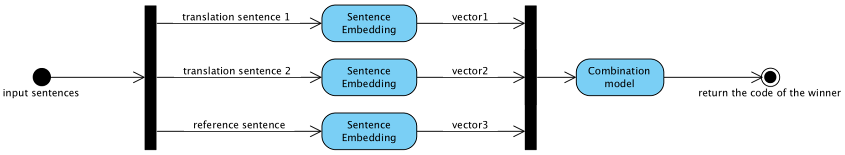

Figure 5.3: Relative ranking framework

between 0 and 100.
The framework of the model to predict the direct assessment is shown in the Figure 5.2. The sentences inputted are first translated into two vectors, Then we put the vectors into the combination model to get a score.

5.1.2 Relative ranking

The target of the relative ranking task is, given two different system output translations of the same source sentence and a reference sentence, through the model, we can judge which translation is better, or the qualities of both translations are the same.
The framework of the model to predict the relative ranking are shown in the Figure 5.3. First, we use three vectors to represent the inputted sentence, two candidate translations and the reference sentence, and then we input them into a combination model to get the ranking.

5.2 Combination model for direct assessment task

5.2.1 Distance based model

The direct assessment combination model takes two vectors as input. One represents the system output translation and the other the reference sentence. Then it outputs the score of the translation.
To combine these vectors, the distance based model calculates the 'distance' between these vectors and scores the system output translation with this distance. Depending on weather neural network is used or not, we divide the distance based model into two categories, direct model and mapped model.

Direct model

Given the system output sentence and reference sentence, we represent the system output sentence with vector X $(x_1, ..., x_n)$ and the reference sentence with vector Y $(y_1, ..., y_n)$. The direct model scores the system output sentence X with a distance method as shown in the equation.

$$score = dis(X, Y) \tag{5.1}$$

Where 'dis' is the distance function.
There are different kinds of the distance function, all of them are introduced in the last subsection.

Mapped model

Given the system output sentence and reference sentence, we represent the system output sentence with vector X $(x_1, ..., x_n)$ and the reference sentence with vector Y $(y_1, ..., y_n)$. The mapped model does not use the X, Y directly. Instead, it map X and Y to X' and Y' with the same linear model and use the distance between the X' and Y' as the final score. The equation are shown below:

$$score = dis(li(X), li(Y)) \tag{5.2}$$

where 'li' is the linear function and 'dis' is the distance function.
There are two reasons for the mapping phase: We use a vector to represent a sentence. Different dimension of the vector may represent sentence in different aspect. Through the linear model we can weight these dimensions and focus on the essential dimensions; Like the idea of Autoencoder, we restructure the inputted vector so that it can fit the task better.

5.2.2 Neural based model

The direct assessment combination model takes two vectors as input. One represents the system output translation and the other the reference sentence. Neural based model takes both vectors as input and output the score and depending on the ways to combine the vectors, we divide the model into two types: concatenate model and the separate model

Concatenate model

Given the system output sentence and reference sentence, we represent the system output sentence with vector X $(x_1, ..., x_n)$ and the reference sentence with vector Y $(y_1, ..., y_n)$. The concatenate model concatenate vector X and Y to form a new vector Z $(x_1, ..., x_n, y_1, ..., y_n)$ and then use the vector Z as the input of the neural network to get a score. The model is shown in Figure 5.4:

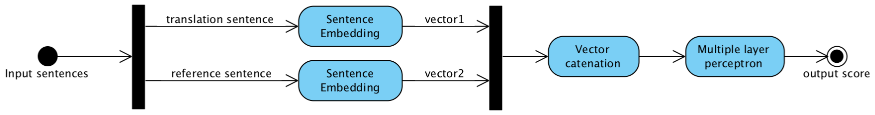

Figure 5.4: Concatenate model for direct assessment task

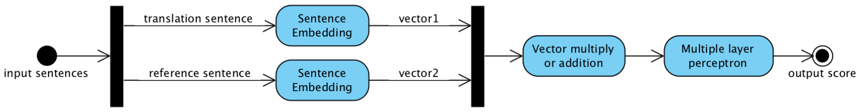

Figure 5.5: Multiply and additive separate model for direct assessment task

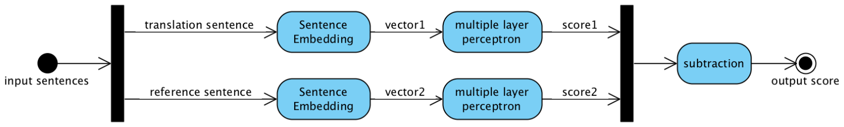

Figure 5.6: Mapping separate model for direct assessment task

Separate model

Given the system output sentence and reference sentence, we represent the system output sentence with vector X $(x_1, ..., x_n)$ and the reference sentence with vector Y $(y_1, ..., y_n)$. Instead of catenation, the separate model treats the inputted vectors differently. We have designed three models here, there are:

- *Multiply separate model* We multiply X by Y to form a new vector and use it as the input of the neural network.
- *Additive separate model* We sum X by Y to form a new vector and use it as the input of the neural network.
- *Mapping separate mdoel* We put the vectors into the neural network separately and generate two scores, s1 and s2, from the network, and then we make the final score as s1 minus s2.

5.3 Combination model for relative ranking task

5.3.1 Distance based model

The relative ranking combination model takes three vectors as input. Two of them represent the translations from two different translation system, and the last one represents the reference sentence. The output of the model is the ranking of the inputted translations with ties allow[1]. To combine these vectors, distance based model calculate the 'distance' between the translation and reference sentences separately and become a score for each translation. Then the model ranks the translation with these scores. The smaller the score is, the better is the translation. Depending on weather neural network is used or not, we divide the distance based model into two categories, direct model and mapped model.

Direct model

Given two system output sentences and reference sentence, we represent the first system output sentence with vector X $(x_1, ..., x_n)$ and the second system output sentence with vector Y $(y_1, ..., y_n)$, the reference sentence with vector Z $(z_1, ..., z_n)$. The direct model chooses the better translation with the equation shown below:

$$best = argmin_{i \quad (X,Y)} dis(i, Z) \tag{5.3}$$

Where 'dis' is the distance function.
There are different kinds of the distance function, and they are the same as the distance function used in the direct assessment task.

Mapped model

Given two system output sentences and reference sentence, we represent the first system output sentence with vector X $(x_1, ..., x_n)$ and the second system output sentence with vector Y $(y_1, ..., y_n)$, the reference sentence with vector Z $(z_1, ..., z_n)$. The mapped model does not use the X, Y, Z directly. Instead, it maps X, Y and Z to X', Y' and Z' with the same linear model and calculates the 'distance' between the X', Z' and Y', Z' to get two scores. Then the model ranks the translation with these scores. The smaller the score is, the better is the translation. The equation is shown below:

$$best = argmin_{i \quad (li(X),li(Y))} dis(i, li(Z)) \tag{5.4}$$

Where 'li' is the linear function, and 'dis' is the distance function.
There are two reasons for the mapping phase:We use a vector to represent a sentence. The different dimension of the vector may represent the different meaning of the sentence. Through the linear model, we can weight these dimensions and focus on the essential dimensions; Like the idea of Autoencoder, we restructure the inputted vector so that it can fit the task better.

[1] two sentences from different machine translator gets the same assess

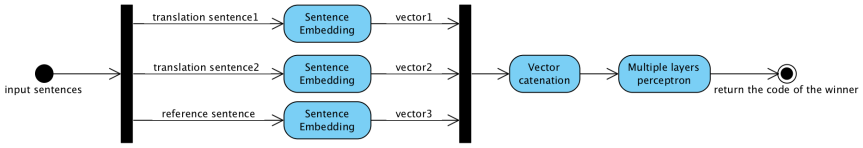

Figure 5.7: Concatenate model for relative ranking task

5.3.2 Neural based model

The relative ranking combination model takes three vectors as input. Two of them represent the translations from two different translation system, and the last one represents the reference sentence. Neural based model takes three vectors as input and outputs the ranking of the inputted translation with ties allowed. Depending on the methods of combining the three vectors, we divide the model into two types: concatenate model and the separate model

Concatenate model

Given two system output sentences and reference sentence, we represent the first system output sentence with vector X $(x_1, ..., x_n)$ and the second system output sentence with vector Y $(y_1, ..., y_n)$, the reference sentence with vector Z $(z_1, ..., z_n)$. The concatenate model concatenate vector X, Y and Z to form a new vector V $(x_1, ..., x_n, y_1, ..., y_n, z_1, ..., z_n)$ and then use the vector V as the input of the neural network. The neural network then outputs the ranking. The whole model is as shown in Figure 5.7:

Separate model

Given the system output sentence and reference sentence, we represent the first system output sentence with vector X $(x_1, ..., x_n)$ and the second system output sentence with vector Y $(y_1, ..., y_n)$, the reference sentence with vector Z $(z_1, ..., z_n)$. Instead of catenation, the separate model treats the inputted vectors differently. We have designed two models here, they are:

- *Half separate model* We concatenate X, Z and Y, Z separately and make the concatenated vectors go through the network to get two scores. We rank the translation sentences according to these scores.
- *Multiply separate model* We multiply X by Z in each dimension to get a new vector X' and multiply Y by Z in each dimension to get the second vector Y'. Then we put the new vectors into a neural network separately to generate two scores. We rank the translation sentences according to these scores.

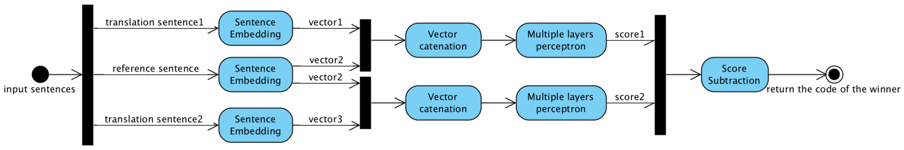

Figure 5.8: Half separate model for relative ranking task

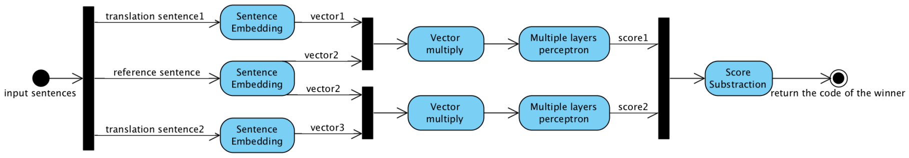

Figure 5.9: Multiply separate model for relative ranking task

Chapter 6

Experiment

In this section, we introduce the experiments that we have done to test the quality of the models introduced in the last chapter. We divide the experiments into two parts. In the first part we concentrate on the direct assessment experiment, and in the second part, we focus on the relative ranking experiment. In each part of the experiment, we again distribute experiments in two subsections: one for distance based experiments and one for neural based experiments.

6.1 Direct assessment experiment

6.1.1 Distance based experiment

The target of the direct assessment(DA) task is, given a system output translation and a reference translation, score the system output translation between 0 and 100.
As mentioned in the last chapter, three points can affect the result of the distance based model strongly, namely:

- *Word embedding* An essential part of sentence embedding. It decides the source of the sentence embedding like w2v or neural translation system.
- *Aggregation function* An essential part of sentence embedding. Once you get the word embedding of each word in a sentence, you need to aggregate these word embeddings to form the representation of the sentence.
- *Distance function* It is the function that used to compute the distance between two different vectors. Distance function only affects the distance based model.

The target of the experiment in this subsection is to test the effect of these elements and figure out the best selection of the elements mentioned above.

sentence embedding	max	mean	sum
w2v	0.502	0.356	0.327
w2v-slim	0.492	0.370	0.259
encoderEmbd	0.454	0.240	0.255
decoderEmbd	0.361	0.334	0.238
decoderOut	0.352	0.349	0.211
decoderState1	0.419	0.373	0.236
decoderState2	0.380	0.400	0.212
decoderCeil1	0.429	0.348	0.211
decoderCeil2	0.25	0.385	0.200

Table 6.1: SentEmbd-agg experiment for the direct assessment task, the result is the Pearson correlation between the target scores and the output of the model, the model is testing with direct assessment accuracy data in wmt17

Direct model experiment

For the direct model, we have designed two experiments. One to find out the best aggregation function and sentence embedding, the other focus on the distance function.

- *SentEmbd-agg experiment* Fix the distance function to L1 distance function and change the sentence embedding and aggregation function during the experiment. We use the direct assessment direct model as described in the last chapter and evaluate the result with Pearson correlation. We test the experiment both with direct assessment data in wmt16 and wmt17 to improve the reliability.
- *SentEmbd-dis experiment* Fix the aggregation function to max aggregation function and change the sentence embedding and distance function during the experiment. We use the direct assessment direct model as described in the last chapter and evaluate the result with Pearson correlation. We test the experiment both with direct assessment data in wmt16 and wmt17 to improve the reliability.

The pearson correlations of SentEmbed-agg experiment are shown in Tables 6.1 and 6.2. We can see that:
The combination of w2v and max aggregation function performs best with data in wmt17. However, the combination of decoder ceil1 and max aggregation function performs best with data in wmt16.
Concerning only sentence embedding, w2v and decoder state perform better than the others.

sentence embedding	max	mean	sum
w2v	0.430	0.224	0.408
w2v-slim	0.435	0.223	0.375
encoderEmbd	0.451	0.420	0.394
decoderEmbd	0.377	0.287	0.379
decoderOut	0.436	0.367	0.407
decoderState1	0.538	0.341	0.432
decoderState2	0.539	0.459	0.423
decoderCeil1	0.466	0.350	0.418
decoderCeil2	0.537	0.455	0.399

Table 6.2: SentEmbd-agg experiment for the direct assessment task, the result is the Pearson correlation between the target scores and the output of the model, the model is testing with direct assessment accuracy data in wmt16

sentence embedding	L1	L2	cos
w2v	0.500	0.452	0.413
w2v-slim	0.492	0.478	0.369
encoderEmbd	0.444	0.415	0.382
decoderEmbd	0.401	0.361	0.330
decoderOut	0.359	0.340	0.318
decoderState1	0.426	0.411	0.368
decoderState2	0.404	0.366	0.350
decoderCeil1	0.403	0.355	0.334
decoderCeil2	0.421	0.357	0.257

Table 6.3: SentEmbd-dis experiment for the direct assessment task1, the result is the Pearson correlation between the target scores and the output of the model, the model is testing with direct assessment accuracy data in wmt17

The effect of aggregation is also complicated. In most of the case max aggregation function performs best, but for some sentence embeddings from a neural translation system, mean aggregation function performs better.
We only get 500 data in wmt17 and 560 data in wmt16, which make the result unstable. The Pearson correlations of SentEmbd-dis experiment are

sentence embedding	braycurtis	canberra	correlation
w2v	0.433	0.420	0.478
w2v-slim	0.410	0.379	0.507
encoderEmbd	0402	0.399	0.402
decoderEmbd	0.380	0.380	0.336
decoderOut	0.322	0.327	0.343
decoderState1	0.400	0.396	0.439
decoderState2	0.417	0.412	0.394
decoderCeil1	0.364	0.346	0.400
decoderCeil2	0.396	0.407	0.277

Table 6.4: SentEmbd-dis experiment for the direct assessment task2, the result is the Pearson correlation between the target scores and the output of the model, the model is testing with direct assessment accuracy data in wmt17

shown in Tables 6.3, 6.4. We only care about the distance function in this experiment. Therefore the conclusion is quite straightforward:
At most of the times, the L1 distance function performs best, although distance function 'correlation' performs better in some case.

Mapped model experiment

We already knew that L1 distance function performs better than the other distance functions. Therefore we continue to use the L1 distance function here. For the Mapped model experiment, we focus on the aggregation functions.
We fix the distance function to L1 distance function and change the sentence embedding and aggregation function during the experiment. We use the direct assessment mapped model as described in the last chapter. We train the model with direct assessment data in wmt16 and test the model with direct assessment data in wmt17. We evaluate the result with Pearson correlation.
The result is shown in the Table 6.5. From the data we can see that:
The combination of w2v sentence embedding and max aggregation function performs best.
Train Pearson correlation is much larger than the test Pearson correlation. We can say that the model is overfitting. It may be due to the lack of training data. We only get 560 data to train the model.
Besides w2v, decoder states1 also performs well in the experiment.

	max		mean		sum	
sentence embedding	test	train	test	train	test	train
w2v	0.409	0.876	0.339	0.873	0.237	0.793
w2v-slim	0.344	0.871	0.352	0.868	0.311	0.628
encoderEmbd	0.400	0.741	0.297	0.856	0.285	-0.163
decoderEmbd	0.298	0.689	0.378	0.829	0.275	-0.104
decoderOut	0.310	0.533	0.315	0.7967	0.213	-0.154
decoderState1	0.399	0.474	0.332	0.870	0.259	0.233
decoderState2	0.333	0.669	0.343	0.868	0.210	0.200
decoderCeil1	0.348	0.396	0.315	0.683	0.226	-0.253
decoderCeil2	0.379	0.319	0.365	0.763	0.214	-0.197

Table 6.5: Mapped model experiment for the direct assessment task, the result is the Pearson correlation between the target scores and the output of the model, the model is training with direct assessment accuracy data in wmt16 and testing with direct assessment accuracy data in wmt17

Compare

We make conclusion of the experiments so far. We already knew that:
Only considering the sentence embedding, w2v and decoder states1 perform well in both experiments: the direct model experiments and the mapped model experiments.
In terms of aggregation function, max aggregation function performs best.
If only considering the distance function, L1 distance function gets the best result.
To compare the direct model and mapped model, we pick some of the results from the experiment above and put them in a single Table 6.6. Both model, direct model and mapped model, here use max aggregation function and L1 distance function.
We conclude that:
Only considering the test result, the direct model performs better then mapped model.
Considering the training result, although direct model performs better than the mapped model in the test set. However, if we get more training data or a better neural model, the mapped model may have the chance to beat the distance model.

	direct model	mapped model	
sentence embedding	test	test	train
w2v	0.500	0.409	0.876
decoder States1	0.426	0.399	0.474

Table 6.6: Pearson correlation between the target scores and the output of the model, the model is testing with direct assessment accuracy data in wmt17

6.1.2 Neural based experiment

The target of the direct assessment(DA) task is, given a system output translation and a reference translation, score the system output translation between 0 and 100.

As mentioned in the last chapter, three points can affect the result of the model strongly, namely:

- *Word embedding* An essential part of sentence embedding. It decides the source of the sentence embedding like w2v or neural translation system.
- *Aggregation function* An essential part of sentence embedding. Once you get the word embedding of each word in a sentence, you need to aggregate these word embeddings to form the representation of the sentence.
- *Neural network* The neural network used in the model.

The target of the experiment in this subsection is to test the effect of these elements and figure out the best selection of the elements mentioned above.

Catenate model experiment

Catenate model first catenate the representaions of the inputted translation and reference sentences. And then it use the the catenation representation as input of a nueral network and get the scores direct from this network. We design an experiment to choose the best aggregation function and sentence embedding.

- *SentEmbd-agg experiment* We vary the sentence embedding and aggregation function during the experiment. The neural network we used here is a multiple layers perceptron with 500 units in the hidden layer. We train the model with direct assessment data in wmt16 and test the model with direct assessment data in wmt17.

The Pearson correlations are shown in the Table 6.7. We can see that: For sentence embedding, hidden stats of the translation system perform much

	max		mean		sum	
sentence embedding	test	train	test	train	test	train
w2v	-0.016	0.744	0.097	0.976	0.084	0.981
w2v-slim	0.058	0.656	0.058	0.885	0.123	0.968
encoderEmbd	0.130	0.591	0.136	0.852	0.118	0.938
decoderEmbd	0.047	0.439	0.083	0.956	0.086	0.952
decoderOut	0.073	0.749	0.040	0.983	0.016	0.974
decoderState1	0.167	0.913	0.191	0.987	0.186	0.987
decoderState2	0.195	0.962	0.221	0.980	0.269	0.983
decoderCeil1	0.156	0.688	0.209	0.983	0.156	0.974
decoderCeil2	0.273	0.882	0.272	0.968	0.263	0.968

Table 6.7: SentEmbd-agg catenate model experiment for the direct assessment task, The result is the Pearson correlation between the target scores and the output of the model, the model is training with direct assessment accuracy data in wmt16 and testing with direct assessment accuracy data in wmt17

better than the others. Among them decoder states1 get the best scores.
Only considering the aggregation function, sum aggregation function performs better than the other two aggregation function. It is different from the distance based model.
All the result show that the model is strongly overfitting.
The number of the training accuracy data is limited. Maybe we can try to use the sub-accuracy data to train the model.
The sentence embedding which working good in the distance based model performs poorly here.

Separate model experiment

The model used in this subsection is different from the catenate model. Therefore, we repeat the experiment for catenate model here but use the separate model instead. Besides we knew that w2v sentence embedding, encoder embedding and decoder embedding do not work with this model, we only try decoder internal states in the experiment. In the experiment, we vary the sentence embedding and aggregation function. We train the model with direct assessment data in wmt16 and test the model with direct assessment data in wmt17. As we have three different separate model, we show the result in three Tables: 6.8, 6.9 and 6.10

	max		mean		sum	
sentence embedding	test	train	test	train	test	train
decoderState1	0.112	0.888	0.206	0.724	0.119	0.922
decoderState2	0.160	0.914	0.261	0.676	0.171	0.853
decoderCeil1	0.132	0.729	0.172	0.880	0.021	0.503
decoderCeil2	0.237	0.649	0.270	0.895	0.230	0.858

Table 6.8: Multiply separate model experiment for the direct assessment task. The result is the Pearson correlation between the target scores and the output of the model, the model is training with direct assessment accuracy data in wmt16 and testing with direct assessment accuracy data in wmt17

	max		mean		sum	
sentence embedding	test	train	test	train	test	train
decoderState1	0.172	0.763	0.205	0.956	0.222	0.953
decoderState2	0.206	0.868	0.223	0.933	0.203	0.950
decoderCeil1	0.565	0.342	0.145	0.954	0.195	0.921
decoderCeil2	0.256	0.713	0.256	0.893	0251	0.847

Table 6.9: Additive separate model experiment for the direct assessment task. The result is the Pearson correlation between the target scores and the output of the model, the model is training with direct assessment accuracy data in wmt16 and testing with direct assessment accuracy data in wmt17

From the result we can see that:
Multiply separate model works better than the other two models.
Instead of sum aggregation function mean aggregation function performs best with the separate model like multiply, additive and mapping separate model we designed here.

Compare

Here is a little conclusion for the neural based model. From the experiment above we see that:
Catenate model works better than separate model in direct assessment task.
The hidden states in the decoder work better than other sentence embedding methods.
The performance of the aggregation function depends on the model that is used in the experiment. For catenate model sum aggregation function works

	max		mean		sum	
sentence embedding	test	train	test	train	test	train
decoderState1	0.116	0.637	0.072	0.863	0.126	0.889
decoderState2	0.041	0.791	0.164	0.817	0.185	0.868
decoderCeil1	0.059	0.100	0.105	0.870	0.100	0.849
decoderCeil2	0.121	0.536	0.143	0.749	0.155	0.740

Table 6.10: Mapping separate model experiment for the direct assessment task. The result is the Pearson correlation between the target scores and the output of the model, the model is training with direct assessment accuracy data in wmt16 and testing with direct assessment accuracy data in wmt17

better, however for separate model mean aggregation function lead the experiment.

6.1.3 Result for the direct assessment experiment

We conclude the direct assessment experiment here. In this subsection, we compare the aggregation function, sentence embedding method and Pearson correlation score for each model when they get the best result. We test these models with the direct assessment data in wmt17. From the Table 6.30 we know that: The direct distance based model with w2v sentence embedding and max aggregation function works best in the direct assessment task.
The pearson correlation of the direct model are better than the baseline but still worse as the best result(0.571).
Comparing the results of w2v and w2v-slim, we can see that, the better the model is, the better the result.

6.2 Relative ranking experiment

6.2.1 Distance based experiment

The target of the relative ranking task is, given two different system output translation of the same source sentence and a reference sentence, through the model, we can judge which translation is better, or the qualities of both translations are the same.
As mentioned in the last chapter, three points can affect the result of the model strongly, namely:

- *Word embedding* An essential part of sentence embedding. It decides the source of the sentence embedding like w2v or neural translation

model	corr	agg	sentEmbd	'dis'
direct model(distance based)	0.500	max	w2v	L1
mapped model(distance based)	0.409	max	w2v	L1
catenate separate model	0.273	max	decoderCeil2	X
multiply separate model	0.270	mean	decoderCeil2	X
additive separate model	0.256	mean	decoderCeil2	X
mapping separte model	0.185	sum	decoderState2	X
state of the art(BLEND)	0.571	X	X	X
baseline(sentBLEU)	0.432	X	X	X

Table 6.11: Conclusion for the direct assessment task. 'model' indicate the model that was used in the experiment. 'agg' is the aggregation function, that was used to get this result. 'corr' is the pearson correlation, 'sentEmbed' is the sentence embedding that was used to get this result, 'dis' is the distance function that was used to get the result, 'X' means that the model does not need this element.

system.

- *Aggregation function* An essential part of sentence embedding. Once you get the word embedding of each word in a sentence, you need to aggregate these word embeddings to form the representation of the sentence.
- *Distance function* It is the function that used to compute the distance between two different vectors. Distance function only affects the distance based model.
- *Neural network* The neural network used in the model.

The target of the experiment in this subsection is to test the effect of these elements and figure out the best selection of the elements mentioned above.

Direct model experiment

For the direct model, we have designed two experiments. One to find out the best aggregation function and sentence embedding, the other focus on the distance function.

- *SentEmbd-agg experiment* Fix the distance function to L1 distance function and change the sentence embedding and aggregation function during the experiment. We use the relative ranking direct model as described in the last chapter and evaluate the result with Tau like correlation. We test the experiment with relative ranking data in wmt16.
- *SentEmbd-dis experiment* Fix the aggregation function to max aggre-

sentence embedding	max	mean	sum
w2v	0.355	0.374	0.359
w2v-slim	0.354	0.368	0.338
encoderEmbd	0.340	0.359	0.359
decoderEmbd	0.324	0.348	0.330
decoderOut	0.350	0.373	0.352
decoderState1	0.391	0.382	0.374
decoderState2	0.373	0.381	0.370
decoderCeil1	0.395	0.374	0.369
decoderCeil2	0.379	0.375	0.356

Table 6.12: SentEmbd-agg experiment for the relative ranking task, the result is the tau like correlation between the target scores and the output of the model, the model is testing with relative ranking data in wmt16

gation function and change the sentence embedding and distance function during the experiment. We use the relative ranking direct model as described in the last chapter and evaluate the result with Tau like correlation. We test the experiment both with relative ranking data in wmt16.

The Tau like correlations of SentEmbed-agg experiment are shown in Table 6.12. We can see that:

The combination of decoderCeil1 and max aggregation function performs best with test data in wmt16.

Concerning only sentence embedding, decoder ceil and decoder state perform better than the others.

The effect of aggregation is also complicated. In most of the case max aggregation function performs best, but for some sentence embeddings from a neural translation system, mean aggregation function performs better.

The Tau like correlations of SentEmbd-dis experiment are shown in Table 6.13. We only care about the distance function in this experiment. Therefore the conclusion is quite straightforward:

At most of the times, the L1 distance function and canberra distance distance function perform best. At some case canberra function works even better then L1 function.

sentence embedding	L1	L2	cos
w2v	0.354	0.344	-0.346
w2v-slim	0.355	0.368	-0.372
encoderEmbd	0.340	0.333	-0.339
decoderEmbd	0.323	0.322	-0.320
decoderOut	0.350	0.330	-0.345
decoderState1	0.391	0.373	-0.384
decoderState2	0.375	0.357	-0.366
decoderCeil1	0.396	0.374	-0.383
decoderCeil2	0.380	0.353	-0.365

Table 6.13: SentEmbd-dis experiment for the relative ranking task, the result is the Tau like correlation between the target scores and the output of the model, the model is testing with relative ranking data in wmt16

sentence embedding	braycurtis	canberra	correlation
w2v	0.366	0.359	0.329
w2v-slim	0.363	0.368	0.360
encoderEmbd	0.342	0.351	0.333
decoderEmbd	0.331	0.338	0.313
decoderOut	0.365	0.373	0.329
decoderState1	0.404	0.409	0.376
decoderState2	0.386	0.398	0.366
decoderCeil1	0.399	0.405	0.375
decoderCeil2	0.394	0.399	0.359

Table 6.14: SentEmbd-dis experiment for the relative ranking task, the result is the Tau like correlation between the target scores and the output of the model, the model is testing with relative ranking data in wmt16

Mapped model experiment

We already knew that canberra and L1 distance function performs better than the other distance functions. Therefore we continue to use the canberra and L1 distance function here. For the Mapped model experiment, we focus our attention on the aggregation functions.

We design two experiments here. In the first experiment we fix the distance

	max		mean		sum	
sentence embedding	test	train	test	train	test	train
w2v	0.318	0.588	0.351	0.717	0.337	0.342
w2v-slim	0.335	0.751	0.355	0.763	0.354	0.502
encoderEmbd	0.306	0.512	0.367	0.788	0.349	0.248
decoderEmbd	0.305	0.512	0.355	0.690	0.332	0.262
decoderOut	0.321	0.469	0.365	0.648	0.353	0.256
decoderState1	0.359	0.354	0.390	0.747	0.368	0.220
decoderState2	0.330	0.480	0.358	0.779	0.347	0.286
decoderCeil1	0.375	0.300	0.385	0.549	0.365	0.163
decoderCeil2	0.352	0.369	0.388	0.625	0.367	0.206

Table 6.15: Mapped model experiment for the relative ranking task with L1 distance function, the result is the Tau like correlation between the target scores and the output of the model, the model is training with relative ranking data in wmt15 and testing with direct assessment accuracy data in wmt16

function to L1 distance function. However in the second experiment we fix the distance function to canberra distance function. In both experiments we change the sentence embedding and aggregation function during the experiment. We use the relative ranking mapped model as described in the last chapter. We train the model with relative ranking data in wmt15 and test the model with relative ranking data in wmt16. We evaluate the model with Tau like correlation.
The result is shown in the Table 6.15, 6.16. From the data we can see that:
The combination of decoder states sentence embedding and mean aggregation function performs best.
Train Tau like correlation is much larger than the test Tau like correlation. We can say that the model is overfitting.
Besides decoder states1, decoder ceil1 also performs well in the experiment.

Compare

Here is a small conclusion for the relative ranking experiments so far. We already knew that:
Only considering sentence embedding, decoder state and decoder ceil perform well in both experiments: the experiment for direct model and experiment for the mapped model.
In terms of aggregation function, max aggregation function performs best.

	max		mean		sum	
sentence embedding	test	train	test	train	test	train
w2v	0.327	0.644	0.346	0.624	0.377	0.367
w2v-slim	0.328	0.734	0.356	0.722	0.328	0.351
encoderEmbd	0.300	0.684	0.353	0.727	0.306	0.323
decoderEmbd	0.307	0.722	0.345	0.656	0.295	0.336
decoderOut	0.309	0.659	0.313	0.418	0.322	0.362
decoderState1	0.336	0.627	0.364	0.671	0343	0.431
decoderState2	0.321	0.666	0.351	0.730	0.338	0.553
decoderCeil1	0.339	0.784	0.353	0.653	0.341	0.422
decoderCeil2	0.331	0.776	0.358	0.667	0.307	0.503

Table 6.16: Mapped model experiment for the relative ranking task with canberra distance function, the result is the Tau like correlation between the target scores and the output of the model, the model is training with relative ranking data in wmt15 and testing with direct assessment accuracy data in wmt16

For distance function, L1 distance function and canberra distance function gets the best result.

To compare the direct model and mapped model, we pick some of the results from the experiment above and put them in a single Table 6.17. Both model, direct model and mapped model, here use max aggregation function. Besides, direct model use canberra distance function, however mapped model use L1 distance function.

We conclude that:

Only considering the test result, the direct model performs better then mapped model.

Considering the training result, although direct model performs better than the mapped model in the test set. However, if we get more training data or a better neural model, the mapped model may have the chance to beat the distance model.

6.2.2 Neural based experiment

The target of the relative ranking task is, given two different system output translation of the same source sentence and a reference sentence, through the model, we can judge which translation is better, or the qualities of both translations are the same.

	direct model	mapped model	
sentence embedding	test	test	train
decoder States1	0.404	0.390	0.747
decoder States2	0.386	0.358	0.779
decoder Ceil1	0.399	0.385	0.549
decoder Ceil2	0.394	0.388	0.625

Table 6.17: Pearson correlation between the target scores and the output of the model, the model is testing with direct assessment accuracy data in wmt17

As mentioned in the last chapter, three points can affect the result of the model strongly, namely:

- *Word embedding* An essential part of sentence embedding. It decides the source of the sentence embedding like w2v or neural translation system.
- *Aggregation function* An essential part of sentence embedding. Once you get the word embedding of each word in a sentence, you need to aggregate these word embeddings to form the representation of the sentence.
- *Neural network* The neural network used in the model.

The target of the experiment in this subsection is to test the effect of these elements and figure out the best selection of the elements mentioned above.

Catenate model experiment

Catenate model fist catenate the representations of the inputted translation and reference sentences. And then it use the catenated representation as input of a neural network. The neural netowrk then output the ranking of the inputted translation sentence. We design two experiment to choose the best aggregation function and sentence embedding.

- *SentEmbd-agg experiment1* We vary the sentence embedding and aggregation function during the experiment. The neural network we used here is a multiple layers perceptron with 500 units in the hidden layer and 1 units in output layer. We train the model with direct assessment data in wmt15 and test the model with direct assessment data in wmt16.
- *SentEmbd-agg experiment2* We vary the sentence embedding and aggregation function during the experiment. The neural network we used here is a multiple layers perceptron with 500 units in the hidden layer

	max		mean		sum	
sentence embedding	test	train	test	train	test	train
w2v	0.091	0.716	0.106	0.670	0.120	0.569
w2v-slim	0.093	0.748	0.125	0.684	0.129	0.607
encoderEmbd	0.029	0.720	0.109	0.619	0.103	0.614
decoderEmbd	0.041	0.668	0.135	0.671	0.130	0.599
decoderOut	0.092	0.600	0.166	0.532	0.139	0.496
decoderState1	0.129	0.787	0.173	0.754	0.195	0.759
decoderState2	0.148	0.756	0.179	0.706	0.180	0.684
decoderCeil1	0.117	0.760	0.177	0.727	0.169	0.698
decoderCeil2	0.149	0.685	0.190	0.653	0.173	0.535

Table 6.18: SentEmbd-agg catenate model experiment1 for the relative ranking task, The result is the Tau like correlation between the target scores and the output of the model, the model is training with relative ranking accuracy data in wmt15 and testing with relative ranking data in wmt16

and 3 units in output layer. We train the model with direct assessment data in wmt15 and test the model with direct assessment data in wmt16.

The Tau like correlations are shown in the Table 6.18 and 6.19. We can see that:
For sentence embedding, hidden states of the translation system perform much better than the others in both experiment 1 (6.18) and experiment 2 (6.19). Among them decoder state1 get the best scores in experiment 1 and decoder ceil2 get the best scores in the experiment 2.
For aggregation function, sum aggregation function performs better than the other two aggregation functions. It is different from the distance based model.
All the result show that the model is strongly overfitting.
The number of the training accuracy data is limited. Maybe we can try to use the sub-accuracy data to train the model.
The sentence embedding which working good in the distance based model performs poorly here.
model with one unit in output layer performs better than the model with three units in the output layer

	max		mean		sum	
sentence embedding	test	train	test	train	test	train
w2v	0.068	0.301	0.080	0.361	0.104	0.434
w2v-slim	0.063	0.321	0.083	0.351	0.093	0.442
encoderEmbd	0.038	0.289	0.076	0.344	0.081	0.444
decoderEmbd	0.021	0.200	0.067	0.356	0.077	0.509
decoderOut	0.064	0.262	0.151	0.434	0.106	0.297
decoderState1	0.138	0.441	0.155	0.451	0.172	0.639
decoderState2	0.149	0.464	0.165	0.440	0.181	0.585
decoderCeil1	0.000	0.165	0.169	0.543	0.146	0.387
decoderCeil2	0.131	0.400	0.195	0.461	0.151	0.342

Table 6.19: SentEmbd-agg catenate model experiment2 for the relative ranking task, The result is the Tau like correlation between the target scores and the output of the model, the model is training with relative ranking accuracy data in wmt15 and testing with relative ranking data in wmt16

Overfitting

The target of relative ranking is, given two translations of a sentence, according to the reference translation we can decide one translation is better than the other one.
In this subsection, we focus on the method to reduce the overfitting.

Choose the number of hidden units in mlp
One reason for overfitting is the capacity of the model is too large. So we try to reduce the capacity of the model in this experiment and compare their performances.
The basic model we used here is a multiple layers perceptron model with 400 dimensions hidden layer, the activation function is ReLU, and the loss function is NLLLoss, the optimizer is Adam and set the learning rate to 1e-3. We train the model with deen data in wmt14 and wmt15 and test it with the German-English data in wmt16 with the method taul like correlation. We only change the dimensions of the hidden layer during the experiment.
Conclusions are shown below (Table 6.20):
If we enlarge the architecture, the training loss will become smaller, but the test loss willget larger

Whiten
It often helps, to do the normalisation or zero-centre before we train the model. We try this out in this experiment.

	loss		tau like	
num of hidden layer	test loss	train loss	test taul	train taul
400	1.24	0.73	0.15	0.59
300	1.24	0.72	0.15	0.59
200	1.24	0.72	0.14	0.59
100	1.25	0.72	0.14	0.56
64	1.23	0.75	0.14	0.52
32	1.21	0.80	0.13	0.48
16	1.17	0.84	0.13	0.41
8	1.13	0.88	0.14	0.40
4	1.09	0.91	0.13	0.37
0	X	X	0.10	0.28

Table 6.20: Selection of model

The basic model we used here is a multiple layers perceptron model with 400 dimensions hidden layer, the activation function is Tanh, and the loss function is NLLLoss, the optimizer is Adam and set the learning rate to 1e-3. We train the model with deen data in wmt14 and wmt15 and test it with the deen data in wmt16 with the method taul like correlation. We only change the method to whiten the data during the experiment.
The candidate methods include:

- *Zero-score*
- *PCA1* do the pca after combining the sys1, sys2 and ref data
- *PCA2* do the pca before combining the sys1, sys2 and ref data

Conclusions of the experiments are (Table 6.21):
When we use the zero-score to whiten the data, the result improves a bit. Same with PCA and PCA2. However, of the three methods, PCA performs best.

Attenuation of learning rate
It always helps, to reduce the learning rate when the loss stays still.
The basic model we used here is a multiple layers perceptron model with 400 dimensions hidden layer, the activation function is Tanh, and the loss function is NLLLoss, the optimizer is Adam and set the learning rate to 1e-3. We train the model with deen data in wmt14 and wmt15 and test it with the deen data in wmt16 with the method taul like correlation. We only change the method to whiten the data during the experiment.

	TwoLayers		OneLayer	
whiten method	test taul	train taul	test taul	train taul
no whiten	0.16	0.81	0.14	0.25
pca	0.18	0.92	0.14	0.29
pca2	0.178	0.91	0.14	0.28

Table 6.21: Selection of whiten method

scheduler	test taul	train taul
no	0.14	0.71
LambdaLR	0.157	0.44

Table 6.22: Attenuation of learning rate

- *Without scheduler*
- $lr\_scheduler.LambdaLR, step = 20$ Multiply the learning rate by 0.1 after 20 epochs.
- $lr\_scheduler.ReduceLROnPlateau$ Reduce the learning rate when the selected value stay still.

Conclusions of the experiments are (Table 6.22):
It's a good idea to add a LambdaLR scheduler in the code. It helps to reduce the test error.

Selection of activation function
The activation function is always crucial in a neural network model. In this experiment, we try some difference activation functions and compare the result.
The basic model we use here is a multiple layers perceptron model with 400 dimensions hidden layer, the activation function is Tanh, and the loss function is NLLLoss, the optimizer is Adam and set the learning rate to 1e-3. We train the model with deen data in wmt14 and wmt15 and test it with the deen data in wmt16 with the method taul like correlation. We only change the activation function during the experiment. The candidate methods include:

- *tanh*

$$tanh = \frac{(e^x - e^{-x})}{(e^x + e^{-x})} \tag{6.1}$$

- *relu*

$$relu = \begin{cases} 0 & forx < 0 \\ x & forx \geq 0 \end{cases} \tag{6.2}$$

activation function	test taul	train taul
tanh	0.14	0.71
selu	0.15	0.58
relu	0.15	0.80

Table 6.23: Selection of activation function

	loss		tau like	
weight decay	test loss	train loss	test taul	train taul
1e-3	1.06	0.93	0.15	0.35
1e-2	1.03	1.02	0.13	0.28
2e-2	1.04	1.04	0.09	0.18
4e-2	1.06	1.07	0.004	0.01

Table 6.24: Selection of weight decay

- *selu*

$$selu = \lambda \begin{cases} \alpha(e^x - 1) & forx < 0 \\ x & forx \geq 0 \end{cases} \tag{6.3}$$

  where $\lambda = 1.0507$ and $\alpha = 1.67326$

Conclusions of the experiments is (Table 6.23):
Comparing with tanh and selu, relu has a better result.

Selection of weight decay
We know that increase the value of weight decay can reduce the effect of the overfitting. In this section, we design an experiment to choose the best weight decay for out model.
The basic model is a multiple layers perceptron model with 400 dimensions hidden layer, the activation function is ReLU, and the loss function is NLL-Loss, the optimizer is Adam and set the learning rate to 1e-3. We train the model with deen data in wmt14 and wmt15 and test it with the deen data in wmt16 with the method taul like correlation. We only change the weight decay during the experiment.
Conclusions are shown below(Table 6.24):
When we set wieght decay to 1e-3, we get the best result. ***Noise in weight***

Adding noise is a useful method against overfitting. In this experiment, we try some method to initialise the weight of the linear model.
The basic model is a multiple layers perceptron model with 400 dimensions hidden layer, the activation function is ReLU, and the loss function is NLL-

init code	test taul	train taul
no init	0.153	0.80
init1	0.167	0.80

Table 6.25: Initialize the parameters of the model before the training. 'init code' indicate weather initialize the parameters or not. no init means initialize with 0 value.

sigma of gauss	test taul	train taul
0	0.14	0.713
0.05	0.14	0.749
0.1	0.14	0.727
0.4	0.11	0.66
1	0.08	0.502

Table 6.26: Adding gauss noise to the input data of the model, 'sigma of gauss' indicate the sigma parameter in gauss function.

Loss, the optimizer is Adam and set the learning rate to 1e-3. We train the model with deen data in wmt14 and wmt15 and test it with the deen data in wmt16 with the method taul like correlation. We only change the initialisation method during the experiment. The candidates of the initialisation method are:

- *Init1:* init.xavier_normal

Conclusions are shown below(Table 6.25):
When we use Xavier to initialize the parameters in the network, the result get better.

Noise in input
In the second last subsection we try adding noise in the weight, now we try to add gauss noise in the input data.
The basic model is a multiple layers perceptron model with 400 dimensions hidden layer, the activation function is Tanh, and the loss function is NLLLoss, the optimizer is Adam and set the learning rate to 1e-3. We train the model with deen data in wmt14 and wmt15 and test it with the deen data in wmt16 with the method taul like correlation. We only change the sigma value of Gauss noise during the experiment. Conclusions are shown below(Table 6.27):
Weather we add noise in the input data or not, it will not affect the result.

	TwoLayer		OneLayer	
num of features	vae	autoencoder	vae	autoencoder
64	0.10|0.58	0.09|0.54	0.10|0.13	0.05|0.07
128	0.13|0.78	0.12|0.76	0.12|0.19	0.05|0.08
256	0.13|0.78	0.12|0.75	0.13|0.25	0.05|0.08

Table 6.27: Test the model with features from the autoencoder, num of features is the number of units in the hidden layer of the autoencoder. There are two scores in each grid, the first one is the train taul like correlation and the second one is the test taul like correlation.

Autoencoder
Use the Autoencoder to extract the features of the sentence embedding. To do that, we train two autoencoder firstly. We use two different Autoencoders here:

- *Normal autoencoder* With three hidden layers with the number of units: 256, 64, 256.
- *VAE*

Apply the features we extracted from the autoencoder in our model, but the result shows no improvement.

Separate model experiment

The model used in this subsection is different from the catenate model. Therefore, we repeat the experiment for catenate model here but use the separate model instead. Besides we knew that w2v sentence embedding, encoder embedding and decoder embedding do not work with this model, we only try decoder internal states in the experiment. In the experiment, we vary the sentence embedding and aggregation function. We train the model with relative ranking data in wmt15 and test the model with relative ranking data in wmt16. As we have two different separate model, we show the result in two Tables: 6.28 and 6.29.
From the result we can see that:
Multiply separate model works better than the other model.
Instead of sum aggregation function mean aggregation function performs best with the separate model like multiply and half separate model we designed here.

	max		mean		sum	
sentence embedding	test	train	test	train	test	train
decoderState1	0.180	0.319	0.220	0.413	0.230	0.491
decoderState2	0.185	0.348	0.223	0.408	0.232	0.435
decoderCeil1	0.096	0.006	0.225	0.393	0.212	0.284
decoderCeil2	0.185	0.294	0.234	0.373	0.226	0.246

Table 6.28: Half separate model experiment for the relative ranking task. The result is the Tau like correlation between the target scores and the output of the model, the model is training with relative ranking data in wmt15 and testing with relative ranking accuracy data in wmt16

	max		mean		sum	
sentence embedding	test	train	test	train	test	train
decoderState1	0.253	0.463	0.337	0.458	0.297	0.593
decoderState2	0.283	0.517	0.348	0.441	0.326	0.498
decoderCeil1	0.197	0.227	0.315	0.524	0.114	0.250
decoderCeil2	0.256	0.290	0.308	0.319	0.136	0.177

Table 6.29: Multiply separate model experiment for the relative ranking task. The result is the Tau like correlation between the target scores and the output of the model, the model is training with direct assessment accuracy data in wmt15 and testing with direct assessment accuracy data in wmt16

Compare

Here is a little conclusion for the neural based model. From the experiment above we see that:
Separate model works better than catenate model in relative ranking task.
The hidden states in the decoder work better than other sentence embedding methods.
The performance of the aggregation function depends on the model that is used in the experiment. For catenate model sum aggregation function works better, however for separate model mean aggregation function lead the experiment.

model	corr	agg	sentEmbd	dis
direct model(distance based)	0.404	max	decoderState1	canberra
mapped model(distance based)	0.364	max	decoderState1	canberra
catenate separate model1	0.195	sum	decoderState1	X
catenate separate model2	0.195	mean	decoderCeil2	X
half separate model	0.234	mean	decoderCeil2	X
multiply separate model	0.348	mean	decoderState2	X
state of the art(METRICS-F)	0.421	X	X	X
baseline(sentBLEU)	0.265	X	X	X

Table 6.30: Conclusion for the relative ranking task. 'model' indicate the model that was used in the experiment. 'agg' is the aggregation function, that was used to get this result. 'corr' is the Tau like correlation, 'sentEmbed' is the sentence embedding that was used to get this result, 'dis' is the distance function that was used to get the result, 'X' means that the model does not need this element.

6.2.3 Result for the relative ranking experiment

We conclude the relative ranking experiment here. In this subsection, we compare the aggregation function, sentence embedding method, Tau like correlation score for each model when they get the best result. The model here are testing with the relative ranking data in wmt16. From the Table 6.30 we know that:
The direct distance based model with decoder states sentence embedding and max aggregation function works best in the relative ranking task.
The tau like correlation of the direct distance based model is worse than the state of the art slightly.

Chapter 7

Conclusion

From the statistic model to the neural network model, the development of machine translation system is fast. Comparing with machine translation system, although many new metrics are created during years, the core idea is the same: based on the features and n-gramm. To cope with the development of neural machine translation system, new metrics are needed.
In this thesis, we propose a new automatic evaluation metrics. The model we designed here are consist of two parts, sentence embedding and combination model. In the first part, we either utilize the w2v model, that provided by google, or the internal states from the machine translation system, so that we can represent a sentence with a vector. To utilize the neural translation system, same as standard translation with neural machine translation, we put the source sentence as input of the encoder, but instead of using the output of the decoder in the last timestamp as the new input data, we use the translation candidate, that we want to score, as input of the decoder. Then we represent the candidate sentence as the internal statuses of the decoder. In the second part, we use a model to combine the vectors we get from the first part and judge the quality of the original sentence according to this vectors.
To test the models, we design many experiments, all of them are described in the experiment chapter. The result of the experiment are exciting.

- *relative rankning task* the distance based model with the combination of sentence embedding 'decoder ceil2', aggregation function 'mean' and distance function 'canberra' works best. Its tau like correlation is near to the state of the art.
- *direct assessment task* the distance based model with the combination of sentence embedding 'w2v', aggregation function 'max' and distance function 'L1' works best.

Besides some exciting points are found during the experiment:
The internal states of the decoder in the machine translation system always perform better than the output of the decoder. When the interval between

the training loss and test loss is large, try to add some constraints to the model, it may enlarge the training loss, but at the same time, it depresses the test loss.

Comparing with other metrics, our model have some advantages, they are:

- *Contextual* It depends on the training data of the machine translation system and w2v model.
- *Character base* So that it can deal with the words that not appeared in the vocabulary(depends on the sentence embedding methods that used in the model).
- *Deep network* Only for relative ranking task, because in this task, we prefer to use 'decoder ceil' sentence embedding.
- *Vector representation* We representation the system output sentence with a vector.
- *Robust* The quality of the metric depends on the machine translation systems and w2v model that are used during the evaluation.

The experiments we have done here are only a small part of the whole research of the new metrics. There are still many points worth to be tested.

The sentence embedding method used in the experiment, is consist of two parts: word embedding and aggregation function. Can we use the sentence embedding model directly? or can we use some start of art word embedding model? Moreover, instead of sentence embedding can we use word embedding directly? Can we integrate some embeddings of the same sentence to get a stronger representation? We only try multiple layer perceptron in the combination model, can we try other neural network, like capsule, convolution layer?

References

Literature

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. “Neural machine translation by jointly learning to align and translate”. In: *arXiv preprint arXiv:1409.0473* (2014) (cit. on pp. 2, 5).

[2] Jacob Benesty, Jingdong Chen, Yiteng Huang, and Israel Cohen. “Pearson correlation coefficient”. In: *Noise reduction in speech processing*. Springer, 2009, pp. 1–4 (cit. on p. 7).

[3] Ondřej Bojar, Yvette Graham, and Amir Kamran. “Results of the wmt16 metrics shared task”. In: 2017 (cit. on p. 12).

[4] Xinwei Cao. *Self-Backtranslation*. 2018 (cit. on p. 2).

[5] K Robert Clarke, Paul J Somerfield, and M Gee Chapman. “On resemblance measures for ecological studies, including taxonomic dissimilarities and a zero-adjusted Bray–Curtis coefficient for denuded assemblages”. In: *Journal of Experimental Marine Biology and Ecology* 330.1 (2006), pp. 55–80 (cit. on p. 6).

[6] Roy De Maesschalck, Delphine Jouan-Rimbaud, and Désiré L Massart. “The mahalanobis distance”. In: *Chemometrics and intelligent laboratory systems* 50.1 (2000), pp. 1–18 (cit. on p. 7).

[7] Yvette Graham, Timothy Baldwin, and Nitika Mathur. “Accurate evaluation of segment-level machine translation metrics”. In: *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 2015, pp. 1183–1191 (cit. on p. 7).

[8] Sepp Hochreiter and Jürgen Schmidhuber. “Long short-term memory”. In: *Neural computation* 9.8 (1997), pp. 1735–1780 (cit. on pp. 2, 5).

[9] Giuseppe Jurman, Samantha Riccadonna, Roberto Visintainer, and Cesare Furlanello. “Canberra distance on ranked lists”. In: *Proceedings of Advances in Ranking NIPS 09 Workshop*. Citeseer. 2009, pp. 22–27 (cit. on p. 7).

[10] Margaret King, Eduard Hovy, John White, Benjamin K T'sou, and Yusoff Zaharin. "MT evaluation". In: *Proceedings of the Machine Translation Summit VII*. 1999 (cit. on p. 1).

[11] Ravindra Kompella. *encoder decoder architecture*. 2018. URL: https://towardsdatascience.com/neural-machine-translation-using-seq2seq-with-keras-c23540453c74 (cit. on p. 4).

[12] Qingsong Ma, Yvette Graham, Shugen Wang, and Qun Liu. "Blend: a Novel Combined MT Metric Based on Direct Assessment—CASICT-DCU submission to WMT17 Metrics Task". In: *Proceedings of the Second Conference on Machine Translation*. 2017, pp. 598–603 (cit. on p. 12).

[13] Matous Machacek and Ondrej Bojar. "Results of the WMT14 metrics shared task". In: *Proceedings of the Ninth Workshop on Statistical Machine Translation*. 2014, pp. 293–301 (cit. on p. 7).

[14] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. "Efficient estimation of word representations in vector space". In: *arXiv preprint arXiv:1301.3781* (2013) (cit. on p. 3).

[15] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. "Distributed representations of words and phrases and their compositionality". In: *Advances in neural information processing systems*. 2013, pp. 3111–3119 (cit. on pp. 2, 3).

[16] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. "BLEU: a method for automatic evaluation of machine translation". In: *Proceedings of the 40th annual meeting on association for computational linguistics*. Association for Computational Linguistics. 2002, pp. 311–318 (cit. on p. 1).

[17] Maja Popović. "chrF: character n-gram F-score for automatic MT evaluation". In: *Proceedings of the Tenth Workshop on Statistical Machine Translation*. 2015, pp. 392–395 (cit. on p. 1).

[18] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. "Dropout: a simple way to prevent neural networks from overfitting". In: *The Journal of Machine Learning Research* 15.1 (2014), pp. 1929–1958 (cit. on p. 5).

[19] Miloš Stanojević and Khalil Sima'an. "Fitting sentence level translation evaluation with many dense features". In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2014, pp. 202–206 (cit. on p. 1).

[20] Alexander Waibel, Toshiyuki Hanazawa, Geoffrey Hinton, Kiyohiro Shikano, and Kevin J Lang. "Phoneme recognition using time-delay neural networks". In: *Readings in speech recognition*. Elsevier, 1990, pp. 393–404 (cit. on p. 2).

[21] Paul J Werbos. “Backpropagation through time: what it does and how to do it”. In: *Proceedings of the IEEE* 78.10 (1990), pp. 1550–1560 (cit. on p. 2).

[22] John White, Theresa O’Connell, and Francis O’Mara. “The ARPA MT evaluation methodologies: evolution, lessons, and future approaches”. In: *Proceedings of the First Conference of the Association for Machine Translation in the Americas*. 1994, pp. 193–205 (cit. on p. 1).