University of Sheffield

Natural Language Generation with Different Styles

Jiaxin Zhang

*Supervisor:*Dr Gerasimos Lampouras and Dr Andreas Vlachos

A report submitted in fulfilment of the requirements
for the degree of MSc in Advanced Computer Science

in the

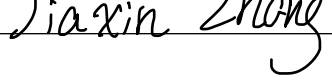
Department of Computer Science

September 9, 2018

Declaration

All sentences or passages quoted in this report from other people's work have been specifically acknowledged by clear cross-referencing to author, work and page(s). Any illustrations that are not the work of the author of this report have been used with the explicit permission of the originator and are specifically acknowledged. I understand that failure to do this amounts to plagiarism and will be considered grounds for failure in this project and the degree examination as a whole.

Name: Jiaxin Zhang

Signature: 

Date: 10/9/2018

Abstract

Natural Language Generation(NLG) is the task of generating natural language from meaning representations. Traditional ways are based on the templates, rule-based, etc. The sentences generated this way seem to be monotonous because it lacks the creativity and the diversity of human language. In this project, we will find out the differences among the styles of the E2E data and use the encoder-decoder model to generate sentences with different styles.

Keywords: Natural Language Generation, E2E dataset, Generating texts with various styles

Acknowledgements

I want to appreciate Gerasimos Lampouras and Andreas Vlachos. It is their help to encourage me to finish my dissertation. I want to thank my parents, It is their selfless support and effort to teach me why I should work hard for my dream. I want to thank my friends, Jiayu Lu, He Yang, Yue Shen, Yinghui Jiang, and many other friends. It is them who encourage me to fight against the giant who wants to destroy my dream.

I want to thank my alma mater, University of Sheffield, it is you who help me find to how to achieve my dream in practical, it is you who help me to become mature, it is you who left my great memory.

Many years ago, Great British explorer George Mallory, who was dead on Mount Everest, was asked why he wants to climb it, he said: because it is there. Nowadays, our world is facing many challenges, I strongly believe AI could change the world, and new hope for knowledge and peace is there. Therefore, this dissertation would be the start of my dream of making the world better place, not because it is easy, but because it is hard.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Natural Language Processing (NLP) is a new trend and hot point in Artificial Intelligence research area. Especially after a great success in Deep Learning, researchers have made a huge breakthrough in NLP. As one of the most important research areas in NLP, Natural Language Generation (NLG) uses non-linguistic data (structured data) to generate texts, which is the research focus. For example, the Los Angels Times generates their journalism of soccer report or weather forecast by non-linguistic data[10]. There are many traditional methods to generate texts, such as templates, hand-coded grammar-based systems, and statistical approach, etc[10].

However, the texts generated by the aforementioned methods may encounter some problems. Because of lacking parallel training data in language style transfer, the achievement in generating language with different styles is less than the other domains in the NLP[7]. Consider such a scenario, where dialog system uses a template to tell you the weather of this week, maybe you want the system to tell you how weather is in the following days. However, these intelligent personal assistants would repeat the answers in the same format, which might lead the conversation to be tedious. So we decide to generate texts based on the E2E dataset with various styles which are full of diversity.

1.1 Aims and Objectives

This project aims to generate texts with various styles by the same structured data. Original generation model which need to be extended will be based on the OpenNMT[14](Open-Source Toolkit for Neural Machine Translation). The model, as it exists now, consists of modeling and generation support. The aim of our project would be to generate different outputs for the same meaning representation. i.e.outputs with different lexical and structured variations.

First of all, We need to cluster the training data with similar styles together. This will be achieved by K-means, and we also need to evaluate whether there are different styles among these clusters. Then different generation models need to be trained at the same

time. We will train encoder model based on bidirectional LSTM and decoder model based on Recurrent Neural Netowrk (RNN) separately. Finally, the trained model could generate different styles of texts (see the following table), and be evaluated with the golden data. We use BLEU score and language modelling to evaluate results.

MR	***name @x@name_0 eattype @x@eattype_0 area @x@area_0***
C0	@x@name_0 is a @x@eattype_0 in the @x@area_0 .
C1	@x@name_0 is a @x@eattype_0 in the @x@area_0 .
C2	@x@name_0 has a @x@customer_rating_0 customer rating and serves @x@food_0 food .
C3	@x@name_0 is a @x@eattype_0 near @x@near_0 .
C4	there is a @x@eattype_0 called @x@name_0 that serves @x@food_0 food .

Table 1.1: examples for generating different styles based on the same meaning representation

1.2 Overview of the Report

Now for practical reasons it is established how this project will advance.

Chapter 2. Literature Survey : Assessment essays that are relevant for Natural Language Generation and Deep Learning, Generating texts with different styles.

Chapter 3. Requirements and Analysis : Explain the aims and objectives of my project and analyse individual parts in detail.

Chapter 4: Design and Implementation : Discuss the whole plan in detail, and elaborate how to put the idea in practice

Chapter 5: Evaluation and Results : Illustrate results and analyse them

Chapter 6: Conclusions : a brief summary of the whole paper, and summarize the conclusion from this paper.

Chapter 2

Literature Survey

In a broad sense, the input of the NLG task is unstructured data, which acts as the basis for the computer to generate output text. In general, concept-to-text Natural Language Generation is the task of expressing the components(attributes and values) of a meaning representation (MR) as a fluent natural language (NL) text. This project concentrates on the various styles of outputs.

2.1 E2E DataSet

E2E data[20] is a new dataset for training end-to-end, data-driven NLG systems in the restaurant domain. E2E data is composed of unordered sets of attributes and values, and it will formulate NL sentences as output. The dataset contains data about the restaurant and more than 50,000 instances of a dialogue-act-based MR with approximately eight references on average.

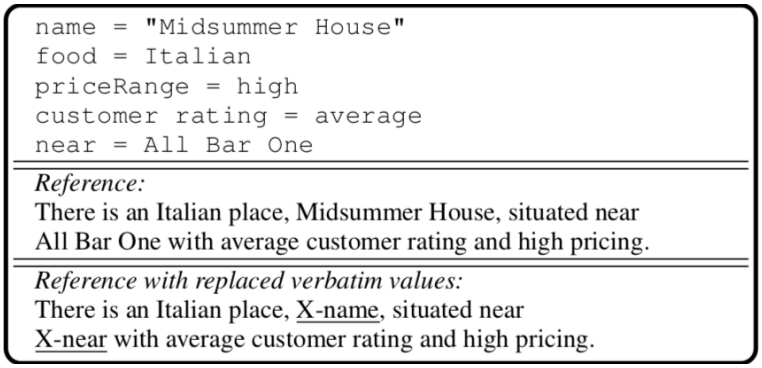


Figure 2.1: Sample MR and corresponding NL reference from the E2E dataset[15]

Figure 2.1 illustrates a sample form of E2E dataset. Each attribute plays the role of the key in the python dictionary, and each of them has a value with a specific data type. For

example, the name in figure 2.1 only takes string values, however, other attributes may take boolean values. Also, we processes the MR and References by replacing verbatim strings with variables ("X-"), which makes the same delexicalized MR has mutiple references.

Novikova et al.[20] compared the E2E dataset with other datasets, such as BAGEL, SF Hotel(SFHOTEL) and SF Restaurants(SFREST), which have different domains from E2E. E2E has more references for MRs than other two datasets have, and E2E contains many bigrams and trigrams which are used only once, where neither SFREST nor BAGEL does. Finally, they evaluate the syntactic and complexity of human references based on the revised D-Level Scale (Lu,2014). The E2E dataset has the highest levels of syntactic complexity, which may be helpful to our main goal — variation.

2.2 Clustering Method

Clustering belongs to unsupervised learning. This kind of algorithm allocates similar samples to one class. They can be divided into top to down and bottom up from the structure. K-means algorithm is the most widely used clustering method, which belongs to bottom up.

Algorithm 1 K_means algorithm

Require: K

function K_MEANS(K)
    Randomly selecting K points as centroids;
    for *each data* $\in$ *Data* **do**
        $calculatethedistancebetweendatapointandcentroids$
        $allocatethisdatapointtothecentroidwhichhasthelowestdistancewithit$
    end for
    for *each cluster* $\in$ $CLusters$ **do**
        $recalculateitsnewcentroid$
    end for
end function

2.3 Natural Language Generation

This Chapter discusses two methods as follows:

2.3.1 Template Based method

Rule-based approach is very common nowadays. Using the template would be an appropriate way under both the scenario where generated texts have low variance and the scenario which needs variation in generating texts. The template could look like :

$ player scored for $ team in the $ minute

This template has three variables, which can be filled with the names of a player, a team, and the minute in which this player scored a goal. It can thus serve to generate a sentence like:

Cristiano Ronaldo scored for Real Madrid in the 63 minutes.

This sentence conveys the clear and exact information, which makes the quality of generating texts reliable. However, there are some disadvantages, for example, the generated text is monotonous. Sometimes we want the generated text to be as rich and colorful as the sports announcers' reports, which needs more complex templates and more useful information.for example:

Real Madrid leads a ball by Cristiano Ronaldo's goal in the 63 minutes.

This expression would make the fans excited. However, such human-authored templates are time-consuming and tedious to construct. Next, we will introduce a corpus-based method which has attracted peoples attention recently.

2.3.2 Recurrent Neural Network

Recurrent Neural Network (RNN) is a special neural network which specializes in sequence data. Sequence data refers to data which was gathered at different times, which could represent changes of the phenomenon by time. Just like the basic neural network which contains an input layer, a hidder layer, an output layer, so does RNN. The main difference between the basic neural network and RNN is that RNN connects not only layers but also neurons. See figure 2.2.

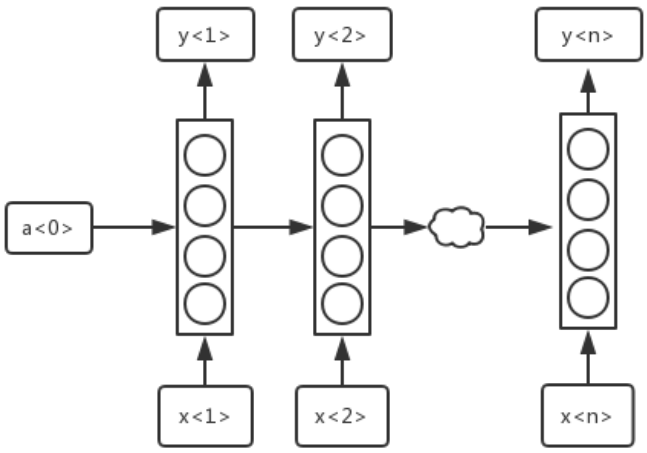


Figure 2.2: RNN Model

Notations:
Superscript $[< t >]$ denotes specific time t in time sequence.
Example: $a^{[<t>]}$ is the t^{th} time. So does $x^{[<t>]}$.
Lowerscript m denotes the what the output is.
Example: W_y denotes that weight W_y is used for outputing y.
After notations, let's define the forward propagation:

$$a^{<t>} = g(W_a[a^{<t-1>}, x^{<t>}] + b_a)$$

$$\widehat{y}^{<t>} = g(W_y a^{<t>} + b_y)$$

Normal RNN only focuses on the previous context, however, the data at the certain time depends not only on the previous data but also the following data. Schuster[22] introduced an extended form of RNN, which is Bidirectional Recurrent Network (BRNN). BRNN concerns both the previous and following context, which provides much more information to make a prediction. BRNN is composed of two RNN with opposite direction, where BRNN not only take the output from the previous time as input but also the output from the following layer. See figure 2.3.

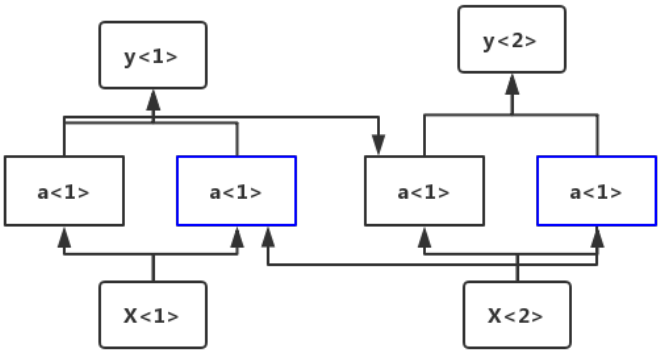


Figure 2.3: BRNN Model

2.3.3 Sequence to Sequence Model

During recent years, with deep learning technology making breakthroughs in the Natural Language Processing field, researchers try to apply the neural network to generate texts. The Sequence to Sequence Model (Seq2Seq) is widely used, which learns a generation model directly from the data, and then generates texts directly.

Kyunghyun Cho et al.[4] put forward a model, and named the model Encoder-Decoder(see figure 2.4) model. This model is applied to the seq2seq model.

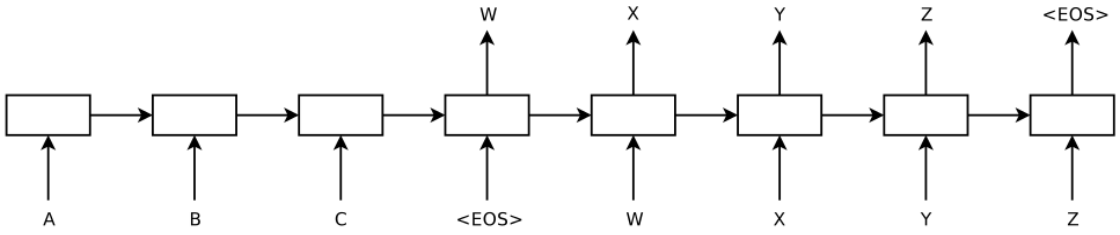


Figure 2.4: Seq2Seq Model

Seq2seq means that generating a sequence of output given a sequence of input. There are many applications in the seq2seq domain, such as machine translation. In this project, the input would be MRs, and the output would be the sentences. Encoder-decoder model was introduced in order to solve the seq2seq problem, where encoder refers to transferring sequential input to vectors with fixed dimension and decoder refers to transferring certain vectors to sequential output.

There are alternative methods to achieve encoder and decoder, such as RNN, GRU, LSTM, etc. We will talk about the model implemented by LSTM as below:

As for LSTM[11], it is much powerful than RNN, because it could memory the information, and resolves the gradient vanish problem in RNN. See figure 2.5.

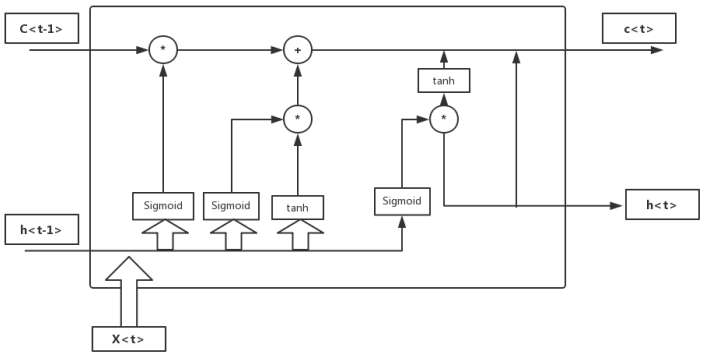


Figure 2.5: LSTM Model

There are five important elements for one LSTM cell. Let's denote them as below:

$$\widehat{C}^{<t>} = tanh(W_c[a^{<t-1>}, x^t] + b_c)$$

The above equation is the candidate output from each LSTM cell.

$$\Gamma_u = \sigma(W_u[a^{<t-1>}, x^t] + b_u)$$

The above equation is the update gate.

$$\Gamma_f = \sigma(W_f[a^{<t-1>}, x^t] + b_f)$$

The above equation is the forget gate.

$$\Gamma_o = \sigma(W_o[a^{<t-1>}, x^t] + b_o)$$

The above equation is the output gate.

$$C^{<t>} = \Gamma_u * \widehat{C}^{<t>} + \Gamma_f * C^{<t-1}$$

The above equation is the final C.

$$a^{<t>} = \Gamma_o * tanh(C^{<t>})$$

The above equation is the final output.

Although the encoder-decoder model is powerful, there are still lots of limitations. The biggest limitation is that the connection between encoder and decoder is only the fixed semantic vector C. This implies encoder would compress the whole sequence into a vector with the fixed length. However, this has two pitfalls, one is that C cannot fully represent the whole sequence, the other one is that the information which is input previously would have less influence than the latter input. These would decrease the accuracy of the output from the decoder.

Many researchers have done research on these models. Wen et al.[23] introduced Semantic Controlled LSTM(Long Short-term Memory) model to generate in the dialogue system. This model reads the structured data by the control gate which is added to the original LSTM model and controls the output in the language model. Kiddon et al.[5] introduced the Neural Checklist Model to solve the reduplicated information generated by the decoder. This model was used for generating recipes by Kiddon, i.e. Inputting the dishes and ingredients to the model, then the model outputs the relevant recipe. The sparse data will occur during generating texts by structured data, which makes the learning model difficult. To solve this problem of sparse data, Lebret et al.[16] introduced the copy-action idea to the Neural Language Model. Lebert applied this model to generate biography of the people in Wikipedia.

Inspired by the Neural machine translation model. Mei et al.[19] regarded generating texts based on structured data as a translation task, i.e. the input is the structured data, the output is the texts. Naturally, Mei proposed the mechanism which needs to model for the importance of the data recording in Neural Machine Translation, i.e. More important data hold higher prior probability, which will let these data be expressed more frequently.

Novikova et al.[20] used TGen (Dusek and Jurcicek,2016a), one of the E2E data-driven systems, where it uses beam search to decode and rerank over the top k outputs, and this way penalizes those outputs that do not verbalize all attributes from the input MR. They have evaluated the development part of the E2E dataset by several automatic metrics. The BLEU score of the TGen system is similar with scores for BAGEL and SFRest datasets.

Shubham Agarwal et al.[1] proposed a character level seq2seq model with attention, which needs no pre-processing step, such as delexicalization, tokenization or lowercasing. Biao Zhang et al.[25] has proposed a new encoder model for encoder-to-decoder architecture. Normal bidirectional RNN simply concatenates the history and future context, however, their encoder, named CAEncoder, could incorporate the future and history contexts. This method performed better than the bidirectional RNN encoder.

2.3.4 Attention Model

From last part, we know that Seq2Seq model tries to encode input sequence as a state with the fixed size, and regard this state as the initial input for the decoder. However, the influence to every state of decoder model from this state is the same. Attention[2] tries to put different influence on the output from the input. Rather than encoding the whole input sequence as the intial state for the decoder, each state in the decoder will rely on a feature $a^{<t,t^{'}>}$, which represents the amount of "attention" that y^t should pay to $a^{t^{'}}$. Context vector takes all cells outputs as input to compute the probability distribution of source language words for each single word decoder wants to generate. The difference between normal LSTM and Attention model can be seen in figure 2.6. The decoder will not only use the last hidden state from the encoder, however, each input in each time step of decoder will be based on a weights called attention weights. The attention weights will explain the amount of "attention" that decoder should pay to every hidden state in encoder.

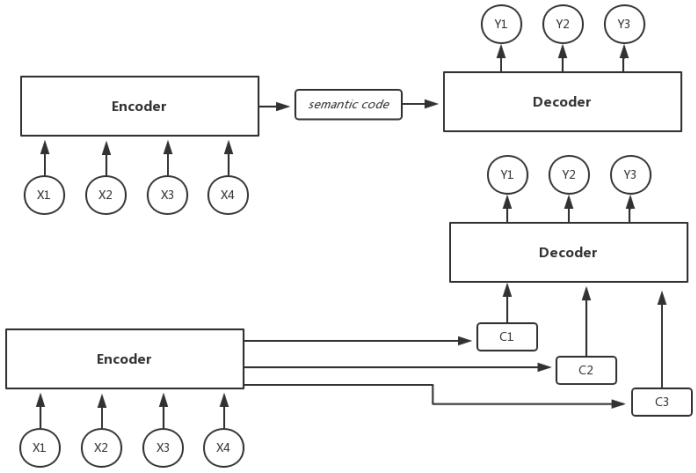


Figure 2.6: Attention Model

2.4 Generating with Style

So far, the existing NLG model which is available for this project can generate only one specific sentence given the same meaning representation as input, so this model needs to be revised in order to generate texts with various styles. This is the key aim of this project.

In the past, people used handcrafted rules to vary the style of the generated linguistic tests which leads to the question of portability and scalability. And then people introduced a new method, which is the statistical natural language generation(SNLG). It uses the annotated data rather than manual writing rules, and achieves the goal.

Recently, Wen et al.[23] mentioned language variational can be easily achieved by sampling from the word distributions of the output from each LSTM cell, however, they do not actually evaluate their model on this capability. Kabbara et al.[13] accessed two corpora of sentences with the same distribution of content albeit rendered in different styles. They learned a style-independent encoder, then passed the output to a style-dependent decoder to change the styles from positive to negative. One important thing in the input of their decoder is that sentences hold the same content. However, this method may face the challenge that it is difficult and expensive to access training data, and also it is hard to determine the style by linguistic features[13].

Juncen Li et al[17]. changed the styles by replacing the phrases relevant with the sentence original stylistic attribute with new phrases relevant to the target style, then combined them into sentences. In their research, there were no aligned sentences with the same content but different styles for training, so they let the system to learn how to separate attributes and content. Zhenxin Fu et al.[8] did the similar thing as Li et al. did. However, they did not try to separate the style from the content. Instead, they developed two models which were Multi-decoder Model and Style-embedding model. Both models used the encoder from the multi-decoder seq2seq (Sutskever, Vinyals, and Le(2014)), where this encoder would separate the content from the input. The first model used multi-decoder to generate outputs with different styles, while the second model used the style embedding to generate the texts in different styles. However, both of their work only focused on the small domain of sentiments, such as positive to negative, etc.

Reiter and Williams[24] discussed generating styles by the examples from SkillSum which was a NLG system. They used three existing methods to generate texts with various styles. One method they used was Explicit Stylistic Control. SKillSum applies a constraint-based approach to make expression choices in Microplanning. It is comprised of a group of hard constraints and a preference function. The constraints try to determine the choice of the linguistic words, and the preference function helps to score the choices. The SkillSum will choose the choice with the highest score. Reiter and Williams[24] thought that the style was important for the preference function. They illustrated some choices that were made in the microplanning process. I will talk about part of the choices which can be used for evaluating the variance among the styles:

1. Lexical difference: This part is to determine which word should be used for the sentence. In the project, we can think about choosing different words to connect the attributes of E2E data, which will let the system generate texts with lexical variation.

2. Ordering difference: This part is about the order of each attribute in the MRs. For example, *John Fitzgerald Kennedy was a great man who served as the 35th President of the*

United States. We can organize the sentence as *John Fitzgerald Kennedy, the 35th President of the United States, was a great man*. The latter sentence does not change the content of the previous sentence while using the same words as the previous. This can be regarded as the order difference in the project.

3. Syntactic choice: This process is about the choice of the syntactic structure. For example, the sentence can be expressed by ordinary sentence(e.g. *He could have a try but he might fail*) or inverted sentence(e.g. *Try as he would, he might fail again*). This feature can help a lot to generate various styles.

The discussion above will be a feasible evaluation choice for evaluating the difference among output sentences.

2.5 Evaluation Method

BLEU Score

BLEU (Bilingual Evaluation Understudy), a method for automatic evaluation of machine translation, is an algorithm for evaluating the quality of text generated by computers from texts generated by humans. This evaluation method is widely used due to its efficiency and convenience. This evaluation can help us to judge the correctness(grammar, word usage, etc.), and the coherence of the generated sentence.

The key point of BLEU is the closer the output sentence is to the human texts, the higher the BLEU score is. BLEU adopts the N-gram matching rule, through which it can find how similar between the candidate texts and the golden texts. However, if there is a generated text like: *the the the the the the the*, and golden text likes: *The cat is on the mat*. If we calculate the naive accuracy by 1-gram matching rule, the denominator will be the count of words in output sentence and the numerator will be the number of words in output sentence which occur in the golden text. For the above example, all the *the* will be matched, both the numerator and the denominator will be 7, i.e. matching score is 7/7. This means the output sentence achieve the 100% accuracy, however, this result is not meaningful. BLEU will choose the maximum between the count of the N-gram in the candidate texts and the count of the N-gram in the golden text, where the numerator will be the sum of counts of all the ngrams and the denominator will be the sum of maximum counts of these ngrams occur in the golden text. The BLEU will penalize the generated sentence which is shorter than the golden text by the equation as below:

$$BP = \begin{cases} 1 & if\ MT\_output\_length > referecnce\_output\_length \\ exp() & otherwise \end{cases}$$

$$exp() = exp(1 - MT\_output\_length/referecnce\_output\_length)$$

where $reference\_output\_length$ is the length of candidate text, and MT_output_length is the length of the golden text, BP refers to the brevity penalty. The final equation is like below:

$$BP \cdot \exp\left(\frac{1}{n}\sum_{n=1}^{N} P_n\right)$$

where BP refers to Brevity Penalty.

Besides BLEU, Zhenxin Fu et al.[8] proposed two methods to evaluate the style transfer, which could be useful for evaluating difference among styles of the texts. One is transferring strength, another one is content preservation. Transfer strength evaluates whether the style is transferred. However, this project does not aim to transfer style but generating various styles. As some words may change during style transfer, the BLEU score may not be suitable here. They proposed to use the cosine distance to measure the similarity between the source sentence and the target sentence. We can use their experience to measure the similarity between the golden references to generate clusters.

ROUGE Score

ROUGE[18] (Recall-Oriented Understudy for Gisting Evaluation) is an evaluation metric based on recall score, whose role is similar with BLEU score. ROUGE includes measures to judge the quality of a summary by comparing it to other golden references, however, it cannot evaluate fluency of the output reference.

$$ROUGE\_N = \frac{\sum_{S \in \{ReferenceSummaries\}} gram_n \in S \sum Count_{match}(gram_n)}{\sum_{S \in \{ReferenceSummaries\}} gram_n \in S \sum Count(gram_n)}$$

In the above formula, n refers to the length of N_gram, $Count_{match}(gram_n)$ refers to how many N_grams co-occur in both testing references and golden references.

NIST Score

NIST (National Institute of Standards and Technology) is another method which is based on the BLEU score to evaluate the quality of sentence generated by the machine translation model[6]. It is a method which calculates not only the overlap of N-grams but also the information of each N-gram, and then divided by the information of the whole text. The formula used to calculate the information is as below:

$$Info(w_1...w_n) = log_2\left[\frac{the\ number\ of\ occurences\ of\ w_1...w_{n-1}}{the\ number\ of\ occurences\ of\ w_1...w_n}\right]$$

Where the denominator and numerator donate the number of N-grams and the corresponding (N-1)-grams occur in the translated texts respectively. After getting the information, we could have the final result as below:

$$Score_1 = \sum_{n=1}^{N} \left\{ \sum_{all\ w_1...w_n\ co-occur} Info(w_1...w_n) / \sum_{all\ w_1...w_n\ co-occur} (1) \right\}$$

$$Score_2 = exp\left\{ \beta log\left[min\left[\frac{L_{sys}}{L_{ref}}, 1 \right] \right] \right\}$$

$$Score = Score_1 * Score_2$$

NIST uses the arithmetic average way, where L_{sys} denotes the length of translated text and L_{ref} denotes the average length of referenced texts.

COVERAGE Score

Before talking about COVERAGE score, let's define ERR score, which calculates how many values in MR are not expressed in NL sentences[15]. COVERAGE score is opposite to the ERR score, it is calculated by $1 - ERR$. Look the formula as below:

$$COVERAGE = \frac{attribute - value\ pairs\ in\ the\ MR\ that\ are\ expressed}{total\ number\ of\ attribute - value\ pairs\ in\ the\ MR}$$

Chapter 3

Requirements and Analysis

This chapter will talk in detail about the aim of our project. Both functional requirement and non-functional requirement will be included in the following discussion. Then we will analyze how to achieve our aim. Finally, an evaluation will be talked, which is as important as the previous ones.

3.1 Aim

Briefly speaking, our aim is to train different generation models which could generate sentences with different styles. Different styles here include orders of the words, lexical variants, etc.

Generally speaking, in order to train different generation models, we need training data to be divided into different clusters, which have specific styles. We also want to train different models in the meantime, so that we need to revise the OpenNMT code. Finally, we want to evaluate output sentences, not only the fluency but also the variants among them from different clusters. Here we just describe the top level of concepts. We will discuss the requirement firstly.

3.2 Requirements

3.2.1 Functional Requirement

1.Pre-processing the training data 2.Clustering the training data 3.Generate sentences

3.2.2 Non-Functional Requirement

1.Could use memory more efficiently 2. Provide a good data structure to store MRs and Refs

3.3 Analysis

In this section, we would like to talk about the details of achieving our aims.

3.3.1 Functional

1. Pre-processing the training data

As we talked in the literature review part, our training data and evaluation data are the E2E dataset. We need to correspond the MR with its relevant Refs, and we also need to get rid of punctuations, etc. The training data will be the E2E dataset. Before using the E2E dataset as training data, we need to preprocess it. As mentioned before, there are on average eight references for each MR. We could try to find whether there is linguistic variance between these NL references, which might help to train a model to learn the styles from these references. To do this, we only consider the MR with more than one NL reference.

Also, to minimize the influence of the noise in the training data, we will delexicalize the values of name and near with another variable in the MRs and NLs. For example, the original NL reference : *There is an Italian place, Midsummer House, situated near will become* will become *There is an Italian place, <X->, situated near, <X->.*

2. Clustering the training data

This is a crucial step for our project. Because our generation model will be trained based on these clusters. First of all, we need to determine what is the number of clusters. In order to find a proper number, we will use Hierarchical clustering first, which could generate a clustering tree for us. Then, we would use the K-means method to cluster the documents.

Secondly, we need to find out a good way to represents Refs, for example, tf-idf. For our baseline model, we just want to cluster the whole training data, and trained generation models based on these clusters. Then we will try some more advanced method. We would cluster the Refs from each MR after we get the clusters for each MR, then we would combine all the Refs together. This clustering process should be based on the data structure we introduced for MRs and Refs. As we want to input one MR , and different generation model could generate different Refs with different styles. We would try a more complicated clustering method finally, we would like to make sure that each MR could appear in as many clusters as possible. In order to achieve this, we would also build a special data structure to store the MRs and Refs.

Last point but most important, we need to correspond the training cluster with the validation cluster together. Because we can not combine the training data and validation data together before clustering, it will be difficult. We need to cluster the validation data in the same way for clustering the training data, and then we will use N-gram overlapping features, in order to allocate validation clusters to their corresponding training clusters.

We have talked about three plans for clustering our data, in order to create good training

data for our generation model. This will be talked in the next section.

3. Generate sentences

We would use OpenNMT toolkit to generate our sentences, however, the existing model could only generate one kind of styles. We want to revise the source code, in order to train different generation models in the meantime.

3.3.2 Non-Functional Requirement

1.Could use memory more efficiently

During our implementation, we always want to save the intermediate data as caches, so we also develop a small but useful package which could store and load these caches. This will help us to reduce the use of memory.

2. Provide a good data structure to store MRs and Refs

As we talked in Functional requirements, there are two clustering plans where we need to create an efficient data structure to store our MRs and Refs data. We have created a class named MR_Ref, and has good extendibility.

3.4 Evaluation

It is important for us to have an evaluation method to evaluate our results. We have used some standard evaluation ways in machine translation, such as BLEU, ROUGE, CONVERAGE, etc.

Despite these ways of evaluation could test the quality of the sentence, we still need to check the variance between the outputs. We have introduced the following criteria:

1. Whether the output has lexical variance.

2. Whether the output has different word order.

3. Whether the output sentence maintains the original meaning.

Chapter 4

Design and Implementation

This chapter will elaborate on the full details of the project. Based on what we talked in the last chapter, the detailed implementation of these requirements will be discussed individually in this chapter. It is crucial to have a solid overall plan before implementing it, so we will talk about the overall design at first, and then breaking into each part of the plan.

4.1 Overall Plan

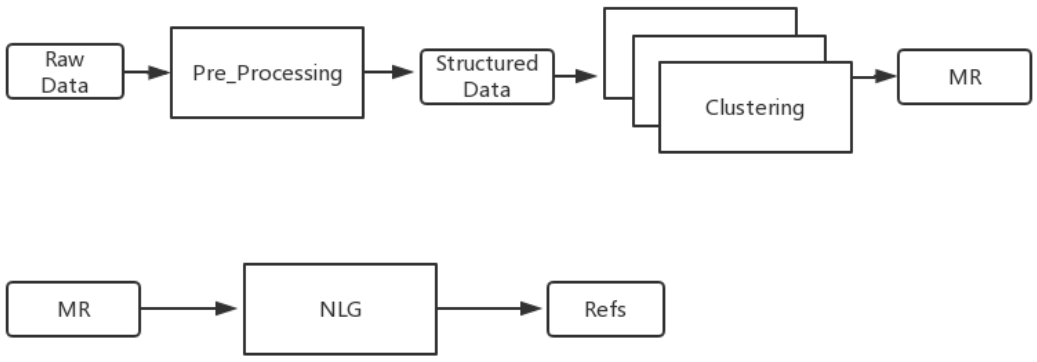


Figure 4.1: Overall Plan

The main steps of our project have been illustrated in figure 4.1. The whole procedure could be divided into three steps,i.e. Pre-Processing, Clustering, Natural Language Generation.

At first, all training data, validation data and testing data which are in CSV format would be preprocessed. This step (Pre-Processing) would output MRs and their corresponding references. The following list shows what the data looks like, and these data would be saved

in the txt files separately(see Tabel 4.1).

MR	name @x@name_0 eattype @x@eattype_0 near @x@near_0 customer_rating @x @customer_rating_0 pricerange @x@pricerange_0
Ref	the eagle is a coffee shop serving japanese food in the less than 20 price range. It is in the riverside near burger king , is family friendly and has a low customer rating .

Table 4.1: MR_Ref

There is one thing needs to be mentioned, although each MR and its corresponding Refs are saved in the same index of the separate files, duplicate MRs are not saved one by one in the index. This problem would be solved by our MR_Ref data structure.

4.2 MR_Ref data structure

We would like to talk about MR_Ref data structure first, by which we saved MRs and corresponding Refs. This data structure is quite important for our implementation, it will play a fundamental role in two of our all clustering methods(see figure 4.2).

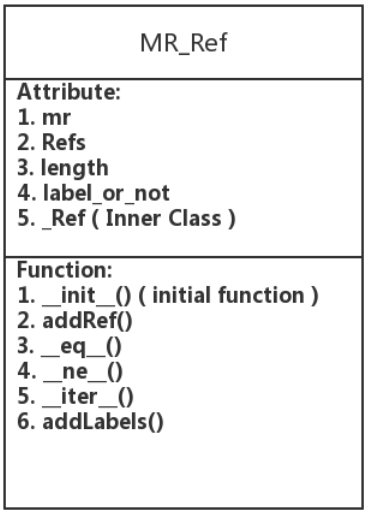


Figure 4.2: Project Plan Part 1

This data structure is created for the clustering step. It contains 5 features and 6 functions and could be extended so that it could fit a new scenario.

4.2.1 Attributes

Tabel 4.2 shows the detail of attribures belonging to the MR_Ref.

Attribute	**datatype**	**function**
mr	string	identify the objects identity
Refs	list	store the Ref object
length	int	record how many references each MR contains
label_or_not	boolean	exclusive attribute for the third clustering method
_Ref	class	class which represents the reference and its features

Table 4.2: Attributes for MR_Ref

The following table 4.3 shows the inner structure of _Ref class which is an inner class of MR_Ref class.

Attribute	**datatype**	**function**
ref	string	store _Ref object name
tf_idf	scipy.sparse.csr.csr_matrix	store references tf_idf feature

Table 4.3: Attributes _Ref

Figure 4.3 illustrates the sturcure of _Ref.

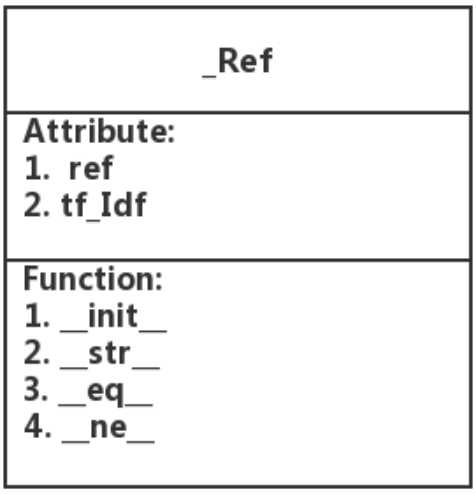


Figure 4.3: strucure of _Ref

4.2.2 Function

Tabel 4.4 shows functions belonging to _Ref.

addRef()	create a Ref object and store it in the Refs list
addLabels()	create a new feature self.labels, and store the clusters for one MRs, references into it

Table 4.4: Functions in _Ref

4.3 Hierarchical Clustering

Although we plan to use K-means algorithm to cluster our references, we still need to find out a proper K (how many number of clusters we want). Hierarchical could visualize dendrogram which describes the clustering result, this gram could help us a lot with choosing a proper K. We choose Scipy[12], which is a very popular Python library used for scientific research, to help us achieve this requirement.

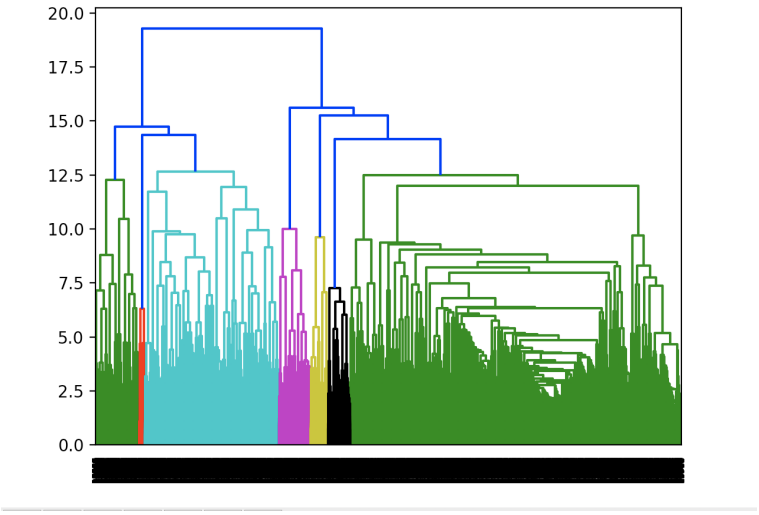


Figure 4.4: Visualize Dendrogram

As we can see from the dendrogram(figure 4.4), choosing k equals to 5 would be a good choice, because 5 clusters contain neither more nor less data.

4.4 Clustering

In this part, we will talk about clustering methods which have been used for clustering our data. They are overall clustering, single clustering, double clustering. Each of them clustered the data by different methods. Overall clustering regards the data as a whole.

Single clustering method only concerns with those MRs which have references in all clusters. Double clustering method would cluster the references among the single MR firstly, then clustering these small clusters together.

4.4.1 Overall Clustering method

The first class clustering method is our baseline model(see figure 4.5). The tf-idf function and K-means function we used in this method are from scikit-learn[21].

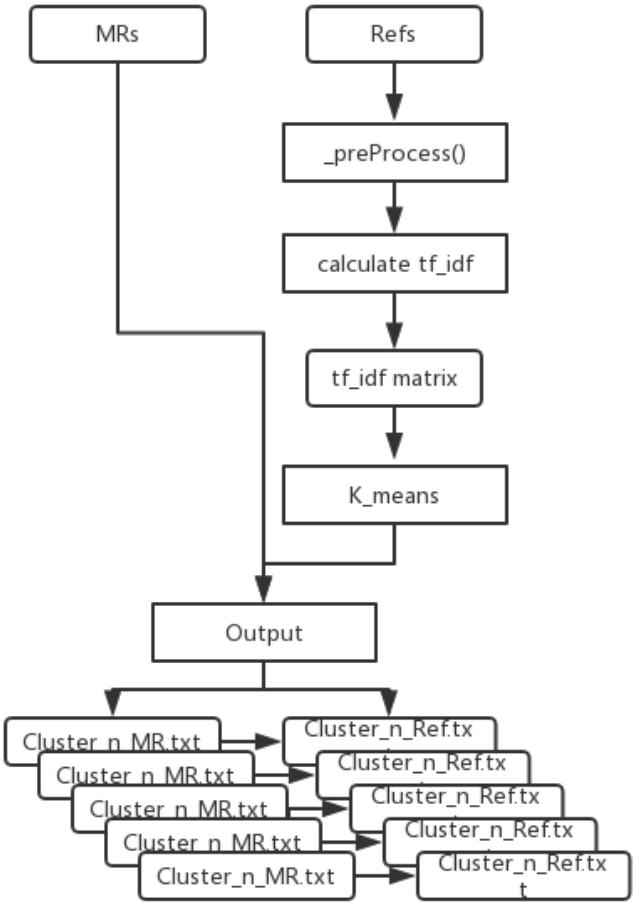


Figure 4.5: Overall Clustering

1. We take all the references and do some preprocesses on them. Punctutaions were be removed and references were added $< BOS >$ padding in front of them, i.e. $< BOS >$ *@x@name_0 is a @x@food_0 place located in the @x@area_0 area. They have @x@pricerange_0 pricing and a customer rating of @x@customer_rating_0.*

2. After step 1, we calculate tf-idf of the given references. Firstly, we only calculate tf-idf for training references in order to cluster them, then we would allocate validation references to the existing clusters by another way.

3. After we get the tf-idf matrix, the matrix will be input into K-means model from sklearn.cluster package which would return us a list of clustering results whose length is 29067(see example from figure 4.6).

```
tf-idf training ...
[4 3 4 ... 3 3 3]
```

Figure 4.6: Clustering Result

4. As indexes of MRs and indexes of Refs are corresponding, we could output our clusters into text files.

By now, we only finish clustering the training data. We need to correspond the validation data with its relevant training clusters. This is achieved by checking the N-gram overlap situations(see figure 4.7).

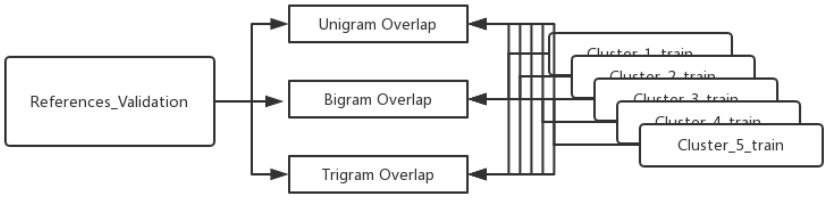


Figure 4.7: N_gram overlap

We check the overlap between the references from the validation data and each cluster and allocate the reference with a cluster who has the highest overlap score him. This could be seen in the First algorithm. For each references in the validation data, we will have three scores for them, which are bigram score, trigram score and fourgram score. Each score is calculated by checking the overlap between the ngrams of validation data and sepatare cluster. Finally, we add these three scores together, and allocate this reference to the cluster which has the

highest score.

Algorithm 2 Correspond validation references with cluserts

Require: $EachClusterN\_gram$ dict, n Cluster number
Ensure: $Cluster\_Validation$

function OVERLAP($N\_gram, validationdata$)
  for *each* $ref \in Refs[]$ **do**
    $score \leftarrow [] * n$
    $create\_bigram()$
    $create\_Trigram()$
    $create\_Fourgram()$
    $score_b \leftarrow bigram\_check()$
    $score_t \leftarrow bigram\_check()$
    $score_f \leftarrow bigram\_check()$
    for $i = 0 \rightarrow n - 1$ **do**
      $score\_[i] \leftarrow score_b + score_t * 2 + score_f * 3$
    end for
    $c \leftarrow argmax\ score$
  end for
  return c
end function

4.4.2 Single Clustering method

In the overall clustering method, the references have been divided into five clusters. Now we want to make sure that each MR could appear in every cluster only once, and for each cluster, one MR could only have one reference. For example, MR_1 contains 8 refernces, two of its references would appear in the same one cluster together by overall clustering method, however, not single clustering method. The reason why we did this way is that we want different generation model could generate different outputs by being given the same MR. This can be understood on intuition by the figure 4.8.

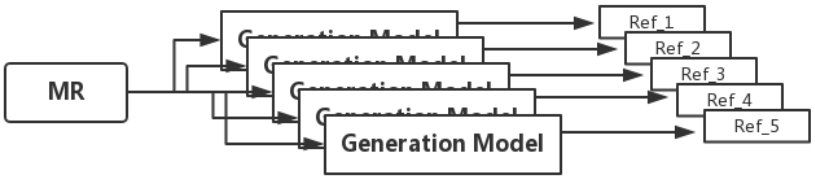


Figure 4.8: Single Clustering

In order to achieve this goal, we need to combine MRs with their corresponding references from the training data. They will be represented by the MR_Ref data structure which was described in part 4.2. This data structure would be saved by the format of pickle, which would be used for the next clustering step.

At the clustering step, K-means would be used to finish this mission.

The new class Cluster is quite important, which could represent a cluster. Its attribute MR_Ref is a list which stores MR_Ref objects. There are two necessary functions, one is addItem(), which takes the responsibility of adding MR_Ref objects to the MR_Ref list, the other one is centroid(), which could create an attribute value which represents the mean tf-idf value of this cluster. The value of one cluster would be changed once a new MR_Ref object is added to the cluster.

Then, K-means would be used for clustering these references. The initial step is a little different, where we randomly take one MR_Ref which contains just K (the number of clusters) references and let each of its references be a single cluster. And this algorithm could make sure that each MR which contains greater or equal 5 references could appear in each cluster only once, and every reference in each cluster are from different MRs. See algorithm 2. We only pick up MRs which contain the K (the number of clusters) references. At beginning, we initialize K clusters by randomly picking one MR. Next we loop all MRs and allocate their references to the most relevant clusters, then we need to change the centroid of clusters.

Algorithm 3 K_means customizing algorithm

Require: *trainingdata* dict, K Cluster number
Ensure: *Cluster*

```
1: K < m
2: function K_MEANS(trainingdata)
3:     randomly initialize K cluster centroids by picking one MR_Ref whose length is K
4:     for each MR ∈ MRs do
5:         Cluster_i ← index (from 1 to K) of cluster centroid closet to X^i
6:     end for
7:     for k = 1 → K do
8:         u_k ← average (mean) of points assigned to cluster k
9:     end for
10:    return Cluster
11: end function
```

After clustering the training data, we also need to allocate validation references to the most relevant cluster. This method used here is just the same as what we did in the first clustering method.

4.4.3 Double Clustering Method

There are two striking characteristics in the previous two methods, one is that the clustering method only refers to the training data, the other one is that we take all the training references together during clustering. However, these have been changed in the third clustering method. First, we select the MRs which contains greater or equal to K (the number of clusters) from the whole MRs, then try to cluster each MRs references first. After doing this, Each MR contains the same number of clusters for their references, which are saved in the exclusive attributes for them in MR_Ref class. Finally, we would run another customised K-means algorithm to associate clusters from one MR with the cluster from other MRs(see figure 4.9).

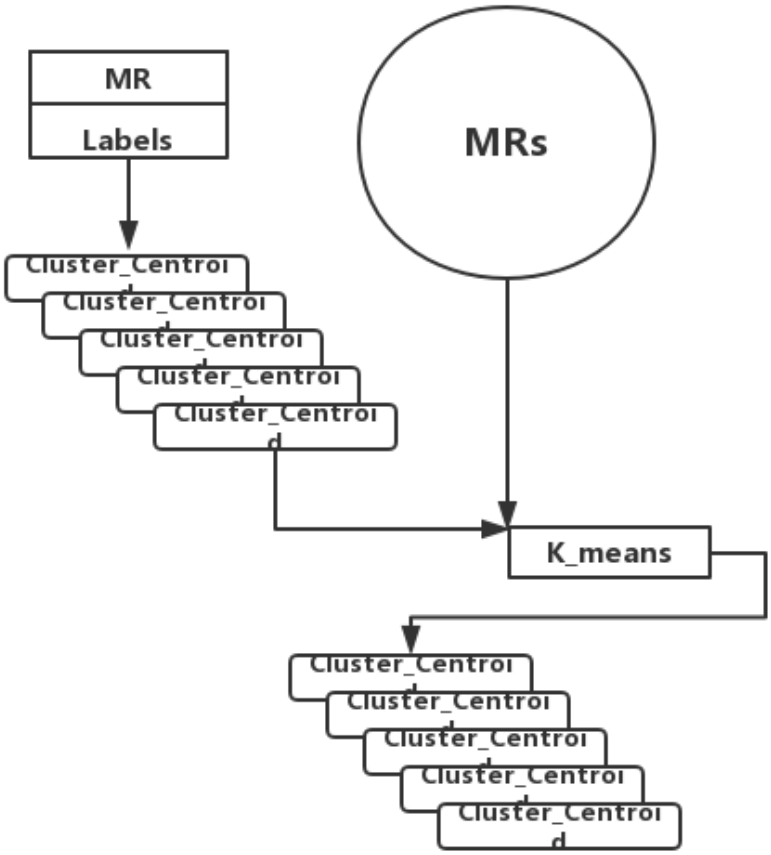


Figure 4.9: Double Clustering

It is different from what we did for validation data in the previous methods. This time we would do the same thing for validation data as we did in training data. We will cluster references of each MR first, and then check the N-gram overlap to allocate these clusters to the most relevant clusters. This step is the same with associating clusters with the most relevant clusters in the training process. See algorithm 3. Firstly, we will cluster references of each cluster, and then allocating these clusters together by checking ngram overlap.

Algorithm 4 K_means customizing algorithm

Require: $MR\_Reflabels[]$, K Cluster number
Ensure: $Cluster$

```
function K_MEANS(labels[])
    for each MR ∈ MRs do
        for each cluster ∈ K do
            check_Ngram()
            c_allocate ← the most relevant cluster
        end for
    end for
    return Cluster
end function
```

4.5 NLG

The generation model we used is based on the model introduced by the Lampouras and Vlachos[15] and improved in Chen et al.[3]. We input MR into the model, after the encoding process, the reference would be output by the decoder model(see figure 4.10).

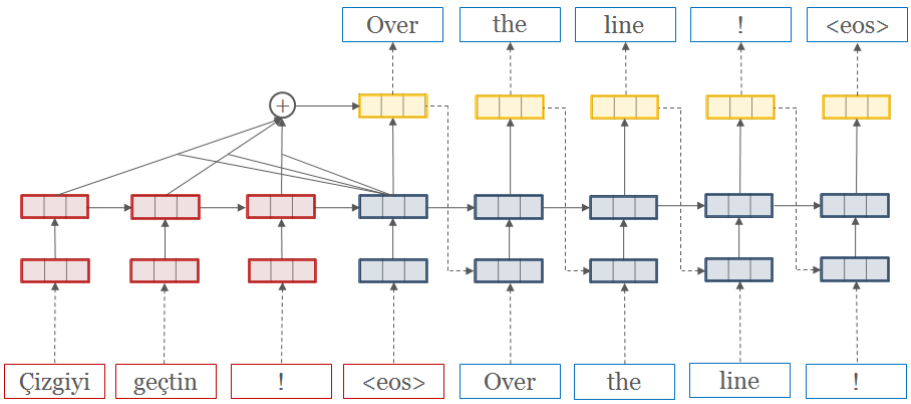


Figure 4.10: OpenNMT[14]

In our implementation, the word-vector size in 50. The encoder model was bidirectional LSTM and the decoder model was LSTM. We chose Adam with dropout to update the

parameters. Heres parameters we use for this model(see table 3.5). The original model could only train one generation model, so some revisions were made to the original code. After some changes, the code could train different models based on different clusters.

Chapter 5

Evaluation and Results

5.1 Evaluation

In this chapter, we will talk about how to evaluate our results, i.e. whether the hypothesis is fluent or not, whether there are different styles between the output sentences. We have implemented our model in the design and implementation chapter, and collected results from that step. We will show what the results looks like and use mutiple methods to evaluate them.

5.1.1 Golden References

In the literature review part, we have discussed about out training data E2E data. Each MR contains multiple references (See table 5.1). These references provides us sufficient object of references which could be used for evaluating our results.

MR	***name @x@name_0 eattype @x@eattype_0 food @x@food_0 pricerange @x@ pricerange_0 customer_rating @x@customer_rating_0 area @x@area_0 near @x@near_0 familyfriendly @x@familyfriendly_0***
Ref 1	the phoenix is a pub serving french food in the riverside area. it is family friendly with a moderate price range , and located near crowne plaza hotel. it has a customer rating of 1 out of 5.
Ref 2	the phoenix is a family - friendly pub near crowne plaza hotel in the riverside area. with a moderate price range , it serves french food and has a customer rating of 1 out of 5.
Ref 3	the phoenix is a one - star rated pub located along the riverside near crowne plaza hotel. they serve medium priced wine and hors d ' oeuvre in a family friendly atmosphere.
Ref 4	the phoenix is a quaint pub serving french food at a moderate price point, specifically friendly to kids. however it has a 1 out of 5 customer rating, but is located near crowne plaza hotel by the riverside .
Ref 5	a family friendly pub , the phoenix , has a moderatey expensive menu that is located next to crowne plaza hotel on the river .

Table 5.1: Example

We could see from the above table, each MR dose contain references with various styles.

5.1.2 Evaluation Method

In the previous discussion, both fluency and variation in styles would be evaluated, which means we need two different evaluation method. One are BLEU Score and ROUGE, another one is language model, which we will talk in detail about how to use language model to evaluate difference.

BLEU Score

We have talked about BLEU score in literature review, it is used widely in machine translation. The BLEU score is just from *Papineni, Kishore, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002.* According to Wikipedia, *BLEU is designed to approximate human judgement at a corpus level, and performs badly if used to evaluate the quality of individual sentences*, we could see this fact in our results.

Language Model

In order to evaluate the difference in styles of our output data, we try to seek help from language model. We build trigram language model for each clusters, and use these language

models to calculate the average probabilities of each cluster. Obviously, the LM that is trained on a particular cluster (e.g. cluster_5) returns a higher average probability for the sentences in that cluster (e.g. cluster_5), than the probability it returns for the other clusters. The same should hold for all language models. See figure 5.1.

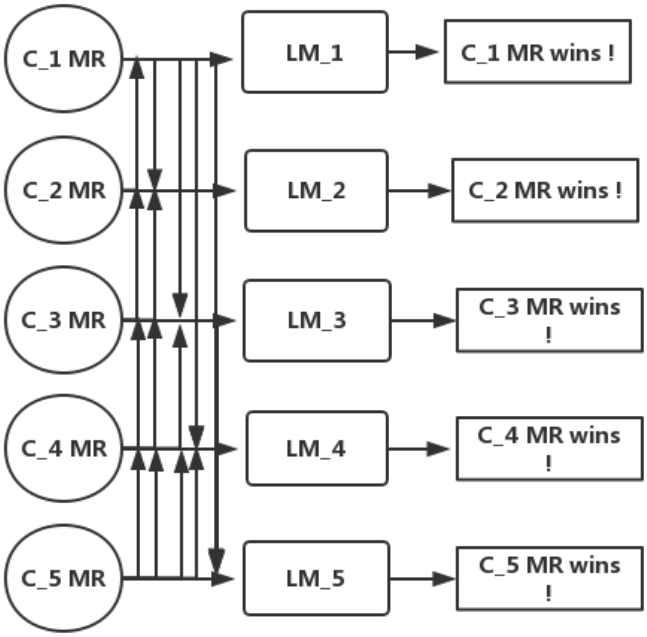


Figure 5.1: Language Model

5.2 Result

As we talked before, training data, validation data, and testing data have been seperated in advance. After our models are trained, we use these trained model to test on our testing data. We try to see the results in two ways, first run each of the "clustered" models on the testing set as a whole, and secondly, try to cluster the the testing set (in the same way we cluster the validation set by checking N_gram overlap) and test each clustered model on the corresponding clustered test set. We will talk about the fluency of testing results first, and then talk about differences between styles.

5.2.1 testing as a whole

Testing as a whole means that we put all the testing data together, rather that clustering them first. We will show the results for each method first and compare them at last.

Overall Clustering Method

Remembering our first clustering method where we just cluster the training data into five clusters, we use these five clustered models to generate sentences from all MRs in the testing set. Here are the results. See table 5.2.

Overall Method (full test set)				
Cluster	**NIST**	**BLEU**	**ROUGE**	**COVERAGE**
C0	7.176	0.5475	0.548	0.99
C1	7.331	0.5504	0.535	0.99
C2	5.829	0.4437	0.49	0.868
C3	6.606	0.505	0.49	0.881
C4	5.180	0.349	0.334	0.612

Table 5.2: Overall Method results

From the above table, we could find that the higher the BLEU score is, the higher the NIST is, although, in NIST, the N-gram which occurs fewer times has more informative weight. The NIST also weakens the influence of the brevity penalty from BLEU. As BLEU score and ROUGE score are very similar, it expresses the idea that the number of N-grams from the generated texts appearing in references are similar with the number of N-grams from references appearing in generated texts. We could find that all clusters perform not bad except for cluster 4. Also, we could find the COVERAGE scores in both Cluster_0 adn Cluster_1 are pretty high, which means that almost all values in the MR are expressed in the generated texts.

If we use the language model method to evaluate the difference between styles of different clusters, we got results in table 5.3.

	LM_1	**LM_2**	**LM_3**	**LM_4**	**LM_5**
C0	0.64838	0.64817	0.497	0.594	0.2314
C1	0.491	0.648	0.470	0.594	0.422
C2	0.519	0.676	0.544	0.634	0.438
C3	0.482	0.639	0.460	0.670	0.464
C4	0.739	0.582	0.589	0.544	0.881

Table 5.3: language model for clusters

We can find that each cluster get the highest score in their corresponding language model, except for cluster_2. From the result above, where BLEU score for cluster_2 is not very high,

it is reasonable that cluster_2 does not get the highest score in its language model. This means that generated sentences in cluster_2 do not have distinct various styles with other clusters.

Now let's compare outputs of the same MR generated from different models. We could find out obvious differences between styles of output sentences based on only one MR. Cluster 4 has the distinct order difference with others.

MR	***name @x@name_0 eattype @x@eattype_0 area @x@area_0***
C0	blue spice is a pub that sells chinese food in the city centre.
C1	blue spice is a pub that sells chinese food in the city centre.
C2	blue spice in the city centre and serves chinese food.
C3	blue spice is a pub that sells chinese food near the city center
C4	there is a pub called blue spice that serves that sells chinese food.

Table 5.4: examples from the first method

Single Clustering Method

In the second clustering method, we build an architecture where the generation models are trained in parallel, given the same input MR to produce multiple texts in different styles. Table 5.3 shows the result.

Single Method (full test set)				
Cluster	**NIST**	**BLEU**	**ROUGE**	**COVERAGE**
C0	6.430	0.4247	0.473	0.905
C1	6.484	0.4396	0.423	0.862
C2	4.670	0.307	0.317	0.597
C3	6.525	0.450	0.416	0.906
C4	4.896	0.349	0.366	0.732

Table 5.5: Single Method results

The above table shows that the Cluster 3 performs better, however, Either ROUGE referring to recall or BLEU referring to precision performs not good. However, COVERAGE in Cluster_2 is not good, which indicates the generated model tries to include more attributes generated texts.

Let's see the language model results. See table 5.6.

	LM_1	LM_2	LM_3	LM_4	LM_5
C0	0.77	0.517	0.517	0.517	0.563
C1	0.775	0.857	0.857	0.857	0.543
C2	0.889	0.76	0.767	0.76	0.79
C3	0.512	1.0	0.843	1.0	0.657
C4	0.570	0.901	0.745	0.901	0.991

Table 5.6: language model for clusters

It looks that cluster_2 does not get the highest score in language model evaluation, however, other 4 clusters perform the best in their own language model.

Still, we want to see some examples of the second method. These are distinct differences between outputs. We could find out that the main difference reflects in the order of sentences.

MR	***name @x@name_0 eattype @x@eattype_0 area @x@area_0name @x@name_0 familyfriendly @x@familyfriendly_0 food @x@food_0 eattype @x@eattype_0 area @x@area_0 near @x@near_0***
C0	blue spice is a not friendly serves chinese food in the riverside area near rainbow vegetarian caf
C1	blue spice is a not friendly serves chinese food in the riverside area near rainbow vegetarian caf
C2	blue spice is a pub serves chinese food near rainbow vegetarian caf it is located in the riverside
C3	there is a non - family friendly pub along the riverside and near rainbow vegetarian caf . its name is blue spice and serves chinese food
C4	blue spice is a pub that serves chinese food near the riverside and the rainbow vegetarian caf . it is not family friendly

Table 5.7: some examples of the second method

Double Clustering Method

In the third clustering method, we cluster references belonging to each MR in training data firstly and allocate clusters in each MR to its most relevant clusters. Let's see the result in table 5.8.

Double Method (full test set)				
Cluster	**NIST**	**BLEU**	**ROUGE**	**COVERAGE**
C0	6.924	0.475	0.462	0.95
C1	6.248	0.412	0.421	0.95
C2	5.281	0.340	0.443	0.91
C3	6.651	0.452	0.431	0.95
C4	6.95	0.469	0.416	0.91

Table 5.8: Double Method results

From the table above, we found that the third method indeed performs slightly better than other methods in COVERAGE score, each cluster gets a pretty high COVERAGE score, which indicates that almost all values in the MR are expressed in the generated texts.

Finally, let's look the language model scores. See table 5.9.

	LM_1	**LM_2**	**LM_3**	**LM_4**	**LM_5**
C0	0.798	0.575	0.798	0.798	0.778
C1	0.337	0.648	0.501	0.501	0.816
C2	0.765	0.524	0.773	0.773	0.751
C3	0.768	0.512	0.765	0.765	0.734
C4	0.284	0.538	0.449	0.449	0.758

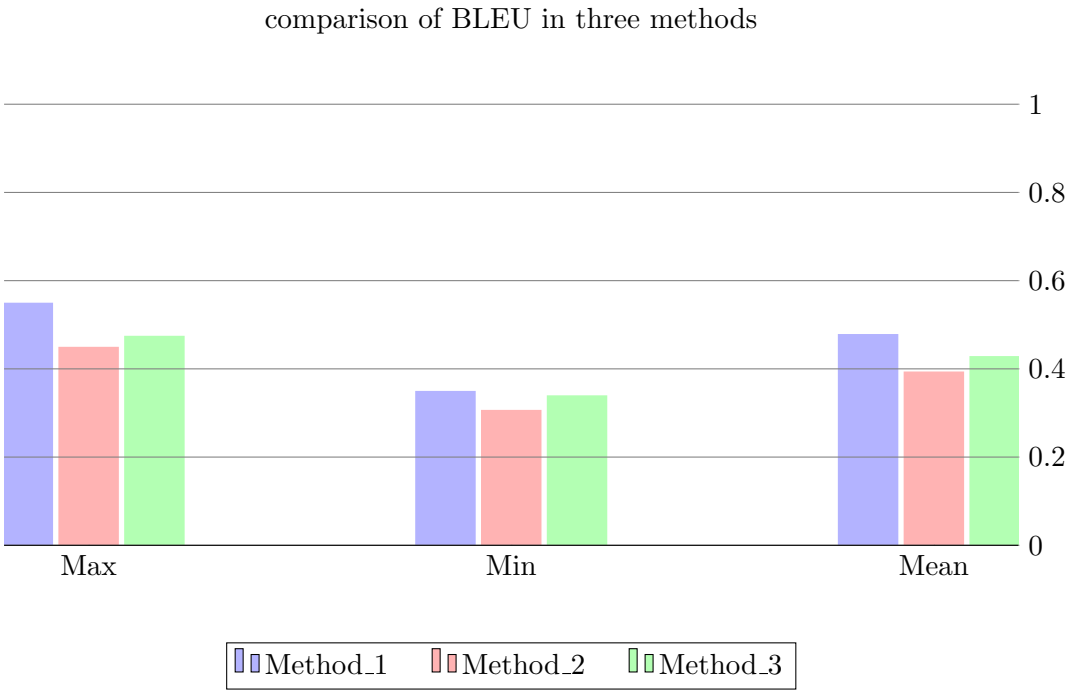
Table 5.9: Double Method results

It is a little strange that only cluster_0 and cluster_4 get the highest scores in their own language models, even if cluser_3 score is a little bit lower than cluster_0 score in its language model, because some of the clusters may not have distinct style. From another point of view, references in some MRs may not have distinct five clusters. We will see what happened when we calculate scores on the clustered testing data.

Before we go to next step, let's compare the BLEU score of each method. See table 5.10.

BLEU Score	Max	Min	Mean
Overall Clustering Method	0.55	0.349	0.479
Single Clustering Method	0.45	0.307	0.394
Double Clustering Method	0.48	0.340	0.429

Table 5.10: comparison of BLEU in three methods

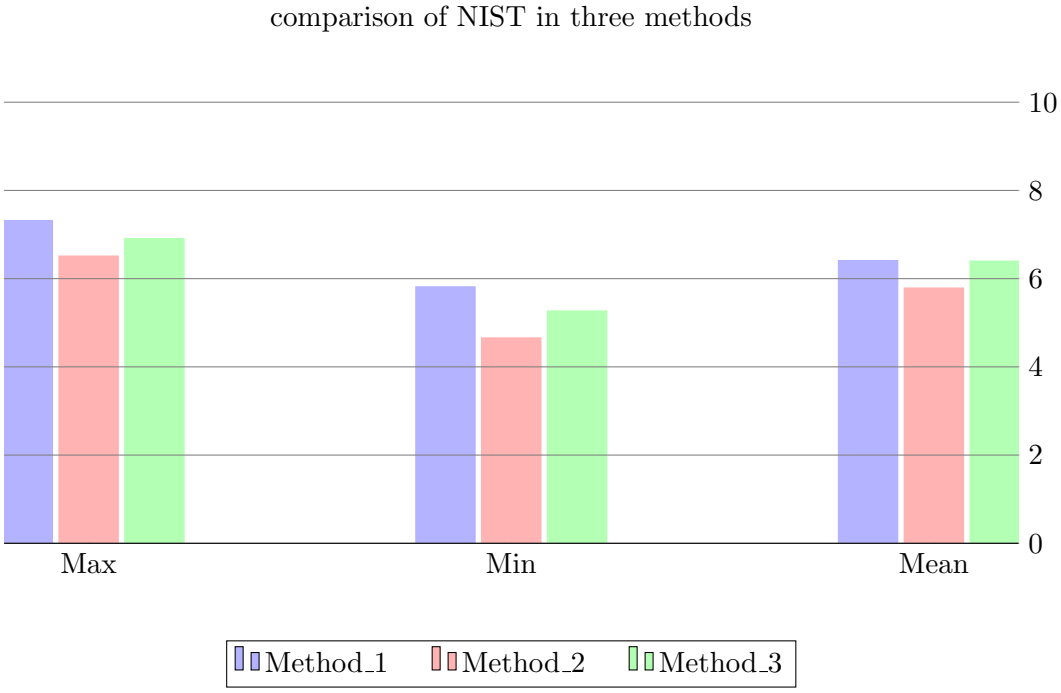


From the chart bar above, we could found that the overall method gets all the highest score in max, min and mean. Some clusters in the single method perform well, some of them do not. This could be explained that results from the overall clustering method have more N-grams from the golden references.

Let's compare the NIST score.

NIST Socres	Max	Min	Mean
Overall Clustering Method	7.331	5.828	6.424
Single Clustering Method	6.525	4.670	5.8
Double Clustering Method	6.923	5.28	6.411

Table 5.11: comparison of NIST in three methods

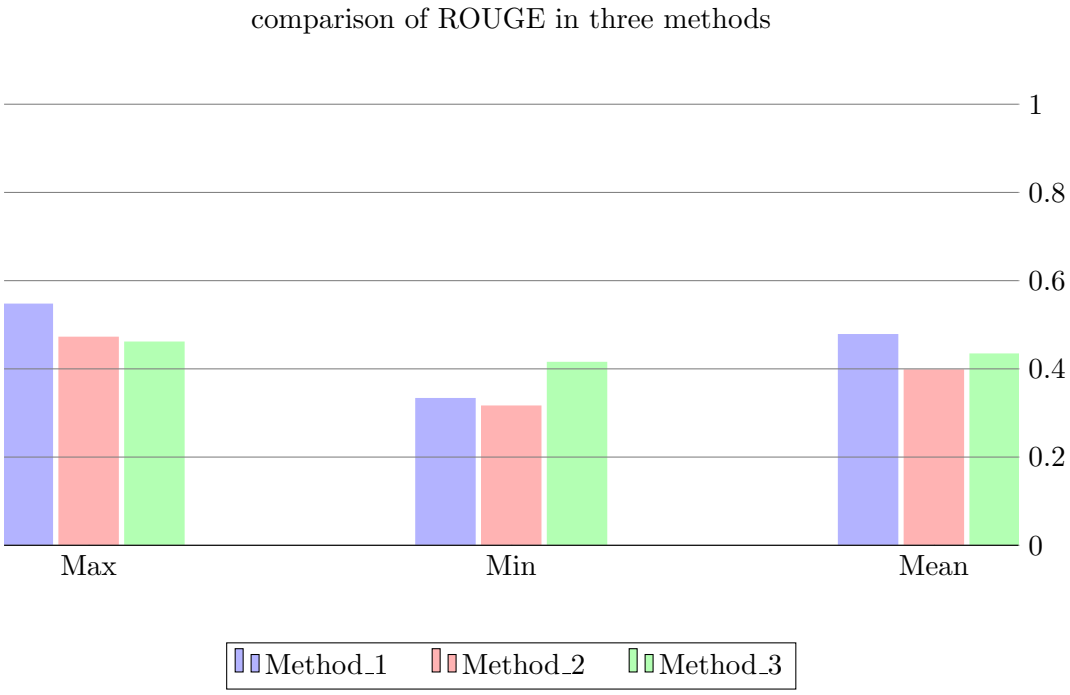


We could find from the above bar chart that the NIST score right reflects the conclusion drawn from the BLEU score. The reason for the single clustering method and double clustering method got lower BLEU scores is not due to the brevity penalty, which penalizes results which are shorter than the length of the golden reference.

Then comes to ROUGE comparison:

ROUGE Score	Max	Min	Mean
Overall Clustering Method	0.548	0.334	0.479
Single Clustering Method	0.473	0.317	0.399
Double Clustering Method	0.462	0.416	0.435

Table 5.12: comparison of ROUGE in three methods



We could find that the minimum ROUGE score for overall clustering method is the highest. This could be explained that results from overall clustering method contain more N-grams from the golden references, however, results also have some N-grams which golden references don't include.

Next, let's look at the comparison in language model score. We could find that method 2 performs the best.

LM_Score	C0	C1	C2	C3	C4
Overall Clustering Method	0.64	0.64	0.544	0.670	0.881
Single Clustering Method	0.77	0.857	0.767	1.0	0.991
Double Clustering Method	0.798	0.648	0.648	0.765	0.758

Table 5.13: LM Comparison

Finally, let's compare the precision score of the language scores for each method. We define the precision score as:

$$Precision = \frac{number\ of\ clusters\ get\ the\ highest\ score\ in\ their\ LMs}{number\ of\ all\ clusters}$$

Method	Overall Clustering	Single Clustering	Double Clustering
Precision	80%	80%	80%

Table 5.14: Precision Score

We could find out that the majority of clusters achieve the highest score in their corresponding language scores. If we look at the language score table above separately, the cluster which does not get the highest score gets the reasonable score.

5.2.2 testing separately

Testing separately means that we do not input all MRs into clustered generation model. Instead, we will cluster testing data first, which was discussed previously. In this part, we will also compare scores between clustered testing data and no-clustered testing data, however, we only focus on the language model score in order to gauge how well the style has been learned. Let's see the result first.

Overall Clustering Method

	LM_1	**LM_2**	**LM_3**	**LM_4**	**LM_5**
C0	0.687	0.626	0.646	0.640	0.173
C1	0.532	0.626	0.618	0.640	0.406
C2	0.553	0.673	0.678	0.686	0.418
C3	0.433	0.626	0.333	0.705	0.423
C4	0.627	0.425	0.604	0.348	0.899

Table 5.15: Overall Method results

We find that each cluster gets a reasonable score in their own language model, which indicates that our generation models were well trained.

Single Clustering Method

Tabel 5.15 shows the result:

	LM_1	LM_2	LM_3	LM_4	LM_5
C0	0.905	0.810	0.810	0.810	0.362
C1	0.843	0.943	0.943	0.943	0.757
C2	0.946	0.764	0.893	0.764	0.643
C3	0.743	1.0	0.871	1.0	0.614
C4	0.614	0.743	0.614	0.743	0.905

Table 5.16: 2nd Method results

From the table above, the score for each cluster is good in their own language model. Scores are pretty high, it is because that we try to let only one reference of each MR appears in a cluster, which leads to there are much less MR-Ref pairs in each cluster.

Double Clustering Method

Let's see the language model score in the following tabel 5.16.

	LM_1	LM_2	LM_3	LM_4	LM_5
C0	0.750	0.591	0.934	0.750	0.705
C1	0.462	0.621	0.463	0.469	0.727
C2	0.603	0.361	0.898	0.609	0.540
C3	0.727	0.551	0.925	0.725	0.674
C4	0.442	0.589	0.457	0.450	0.704

Table 5.17: Double Method results

From the tabel above, there are three clusters which do not get the highest score in their own language models.

Let's take some example outputs to check. We could find that cluser 3 has not only order difference but also lexical difference.

MR	***name @x@name_0 eattype @x@eattype_0 food @x@food_0 pricerange @x@ pricerange_0 customer_rating @x@customer_rating_0 area @x@area_0 near @x@near_0 familyfriendly @x@familyfriendly_0***
C0	the punter is a kid friendly italian restaurant near rainbow vegetarian caf in the riverside area . it is high priced with a customer rating of 1 out of 5 .
C1	the punter is a children friendly italian restaurant located in riverside near rainbow vegetarian caf . the prices are on the high end with a 1 out of 5 customer rating .
C2	the punter is a high priced italian restaurant . it is kid friendly with a customer rating of out of 5 near rainbow vegetarian caf in the riverside area .
C3	the punter , an italian restaurant located in riverside near rainbow vegetarian caf , offers a children friendly atmosphere with prices on the higher end . the punter has a 1 out of 5 customer rating .
C4	the punter , an italian restaurant located in riverside near rainbow vegetarian caf , offers a children friendly atmosphere with prices on the higher end . the punter has a 1 out of 5 customer rating .

Next, let's compare the language model score of different clustering methods.

LM_Score	C0	C1	C2	C3	C4
Overall Clustering Method	0.687	0.626	0.678	0.705	0.899
Single Clustering Method	0.905	0.943	0.893	1.0	0.905
Double Clustering Method	0.750	0.621	0.898	0.725	0.704

From the tabel above, we could find that the method 2 still got the highest score. Because there are much less MRs in each cluster, so clusters could show more difference in styles.

Chapter 6

Conclusions

6.1 Conclusion

This project proposed to generate different styles of sentences based on MRs. We have tried different clustering method to get clustered data with various styles and used encoder-to-decoder architecture to train the model.

Previous work has been done in generating different styles of sentences, such as humorous and romantic styles[9], which paid attention to the semantic differences. Whereas, we focused on the difference between E2E data, and pay more attention to the lexical and structural variety of the outputs.

In the first step of our project, we have introduced three methods to cluster the data and talked about how to relate validation data to clusters. After that, we used our encoder-decoder model to generate sentences based on these clustered models. Finally, we have compared these methods and analyzed the difference of styles among these clusters.

We could find the E2E dataset does contain the various style features, although it mainly reflects on the order and lexical choice. The sentences generated from our different models are good based on the language modeling scores and the average BLEU scores. We have also introduced three clustering methods for the E2E dataset and compared the difference between them. Finally, we find out the overall clustering method performs the best.

6.2 Future Work

We have achieved a good result in our project. However, If we could train models parallelly, where clustered models could generate sentences in the meantime, the result will be much more interesting. We could build only one encoder for all models, and each model has different decoders. More accurately, after backpropagation, each model has the same weight matrics for their encoder models and different weight matrices for their decoder models.

Bibliography

[1] Agarwal, S., Dymetman, M., and Gaussier, E. A char-based seq2seq submission to the e2e nlg challenge.

[2] Bahdanau, D., Cho, K., and Bengio, Y. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* (2014).

[3] Chen, C., Lampouras, G., and Vlachos, A. Sheffield at e2e: structured prediction approaches to end-to-end language generation.

[4] Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078* (2014).

[5] Choi, C. K. L. Z. Y. Globally coherent text generation with neural checklist models.

[6] Doddington, G. Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In *Proceedings of the second international conference on Human Language Technology Research* (2002), Morgan Kaufmann Publishers Inc., pp. 138–145.

[7] Fu, Z., Tan, X., Peng, N., Zhao, D., and Yan, R. Style transfer in text: Exploration and evaluation. *arXiv preprint arXiv:1711.06861* (2017).

[8] Fu, Z., Tan, X., Peng, N., Zhao, D., and Yan, R. Style transfer in text: Exploration and evaluation. *arXiv preprint arXiv:1711.06861* (2017).

[9] Gan, C., Gan, Z., He, X., Gao, J., and Deng, L. Stylenet: Generating attractive visual captions with styles. In *Proc IEEE Conf on Computer Vision and Pattern Recognition* (2017), pp. 3137–3146.

[10] Gatt, A., and Krahmer, E. Survey of the state of the art in natural language generation: Core tasks, applications and evaluation. *Journal of Artificial Intelligence Research 61* (2018), 65–170.

[11] Gers, F. A., Schmidhuber, J., and Cummins, F. Learning to forget: Continual prediction with lstm.

[12] Jones, E., Oliphant, T., Peterson, P., et al. SciPy: Open source scientific tools for Python, 2001–. [Online; accessed ¡today¿].

[13] Kabbara, J., and Chi Kit Cheung, J. Stylistic transfer in natural language generation systems using recurrent neural networks. 43–47.

[14] Klein, G., Kim, Y., Deng, Y., Senellart, J., and Rush, A. M. Opennmt: Open-source toolkit for neural machine translation. In *Proc. ACL* (2017).

[15] Lampouras, G., and Vlachos, A. Imitation learning for language generation from unaligned data. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers* (2016), pp. 1101–1112.

[16] Lebret, R., Grangier, D., and Auli, M. Neural text generation from structured data with application to the biography domain. *arXiv preprint arXiv:1603.07771* (2016).

[17] Li, J., Jia, R., He, H., and Liang, P. Delete, retrieve, generate: A simple approach to sentiment and style transfer. *arXiv preprint arXiv:1804.06437* (2018).

[18] Lin, C.-Y. Rouge: A package for automatic evaluation of summaries. *Text Summarization Branches Out* (2004).

[19] Mei, H., Bansal, M., and Walter, M. R. What to talk about and how? selective generation using lstms with coarse-to-fine alignment. *CoRR abs/1509.00838* (2015).

[20] Novikova, J., Dusek, O., and Rieser, V. The E2E dataset: New challenges for end-to-end generation. *CoRR abs/1706.09254* (2017).

[21] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research 12* (2011), 2825–2830.

[22] Schuster, M., and Paliwal, K. K. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing 45*, 11 (1997), 2673–2681.

[23] Wen, T.-H., Gasic, M., Mrksic, N., Su, P.-H., Vandyke, D., and Young, S. Semantically conditioned lstm-based natural language generation for spoken dialogue systems. *arXiv preprint arXiv:1508.01745* (2015).

[24] Williams, S., and Reiter, E. Generating basic skills reports for low-skilled readers. *Natural Language Engineering 14* (2008), 495–525.

[25] Zhang, B., Xiong, D., Su, J., and Duan, H. A context-aware recurrent encoder for neural machine translation. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP) 25*, 12 (2017), 2424–2432.