Karlsruhe Institute of Technology

A Knowledge-grounded Conversation Model with Word-level Incorporation

Master's Thesis of

Zhang Mingyu

at the Department of Informatics
Institute for Anthropomatics and Robotics

Reviewer:	Prof. Dr. Alexander Waibel
Second reviewer:	Prof. Dr.-Ing. Tamim Asfour
Advisor:	M.Sc. Stefan Constantin

14. July 2019 – 13. January 2020

Karlsruher Institut für Technologie
Fakultät für Informatik
Postfach 6980
76128 Karlsruhe

I declare that I have developed and written the enclosed thesis completely by myself, and have not used sources or means without declaration in the text.

PLACE, DATE

. . . . . .

(Zhang Mingyu)

Abstract

Deep learning techniques contribute significant advances on dialogue systems. Nevertheless, although end-to-end neural approaches help to generate fluent and appropriate response, there are some problems, among which a typical issue is the trend of generating "general", "safe" responses like "it's a good idea" and "I don't know what you mean". This is because these fully data-driven systems lack grounding in the real world and infer responses using only the collection of conversational transcriptions.

For this issue, we proposed a neural conversational model that is able to utilize ground knowledge to produce meaningful and specific responses without traditional pipeline structure by encoding ground knowledge with pre-trained model BERT. We use Transformer as encoder-decoder backbone and put forward 2 schemes: 1. use BERT embedding corresponding to CLS token to represent a fact sentence; 2. use BERT embedding to keywords in a fact sentence. Then BERT embedding are injected to each token in source sequence, so that information from source sequence and external knowledge is combined.

In our experiments, we used the datasets and the evaluation metrics from the sentence generation task in Dialog System Technology Challenges 7 (DSTC7-Track2). According to the evaluation results, the proposed system worked fine at the DSTC7-Track2 and performed better than the baseline RNN model provided by the competition organizers, as well as our own baseline RNN model and baseline Transformer model. In appropriateness metrics, NIST scores are about 0.3 points higher than baselines, METEOR are 0.04 higher, only BLEU are lower than baselines. In informativeness metrics, entropy are about 0.1 higher and diversity are 0.1 higher than baseline.

Zusammenfassung

Deep-Learning-Techniken tragen zu erheblichen Fortschritten bei den Dialogsystemen bei. Obwohl durchgängige neuronale Ansätze dabei helfen, eine natürliche und angemessene Reaktion zu erzielen, gibt es einige Probleme, darunter der Trend zur Erzeugung von ”allgemeinen”, ”sicheren” Reaktionen wie ”it’s a good idea” und ”I don’t know what you mean”. Dies ist darauf zurückzuführen, dass diese vollständig datengesteuerten Systeme in der realen Welt keine Grundlage haben und Antworten nur unter Verwendung der Sammlung von Gesprächstranskriptionen ableiten.

Für diese Ausgabe haben wir ein neuronales Konversationsmodell vorgeschlagen, das in der Lage ist, das Grundwissen zu nutzen, um ohne traditionelle Pipeline-Struktur aussagekräftige und spezifische Antworten zu erhalten, indem das Grundwissen mit dem vorgeübten Modell BERT codiert wird. Wir verwenden Transformer als Encoder-Decoder-Backbone und schlagen zwei Schemata vor: 1. Verwenden Sie die BERT-Embedding, die dem CLS-Token entspricht, um einen Faktensatz darzustellen. 2. Verwenden Sie die BERT-Embedding für Schlüsselwörter in einem Faktensatz. Anschließend wird jedem Token in der Quellsequenz eine BERT-Embedding hinzugefügt, sodass Informationen aus der Quellsequenz und externem Wissen kombiniert werden.

In unseren Experimenten verwendeten wir die Datensätze und Bewertungsmetriken aus der Satzerzeugungsaufgabe in Dialog System Technology Challenges 7 (DSTC7-Track2). Den Bewertungsergebnissen zufolge funktionierte das vorgeschlagene System beim DSTC7-Track2 einwandfrei und schnitt besser ab als das von den Wettbewerbsveranstaltern zur Verfügung gestellte RNN-Basismodell sowie unser eigenes RNN-Basismodell und Transformer-Basismodell. Gemäß der Bewertung liegen die NIST-Werte in Angemessenheitsmetriken etwa 0,3 Punkte über den Basiswerten, METEOR 0,04 und nur BLEU unter den Basiswerten. In der Informationsmetrik ist die Entropie um etwa 0,1 höher und die Diversität um 0,1 höher als die Basislinie.

Contents

List of Figures

List of Tables

1 Introduction

A dialogue system is a computer agent that makes interaction with human via conversation. More precisely, in this thesis, it refers to non-task-oriented systems, which are chatbots that generate proper responses given conversation inputs. Typically both query and response are in text form. Dialogue systems have been widely used on a variety of business scenario such as customer service and personal assistant.

In tradition, a dialogue system is usually built as a pipeline structure, as shown in Figure 1.1. Such a system mainly consists of ASR (automatic speech recognition, convert speech to text), NLU (natural language understanding, represent the semantics of natural language), DM (dialogue modeling, manage flow of conversation), NLG (natural language generation, generate natural language from semantic representation) and TTS (text-to-speech, convert text to speech). In this thesis, we consider texts as the input and output of the system and only focus on NLU, DM and NLG.

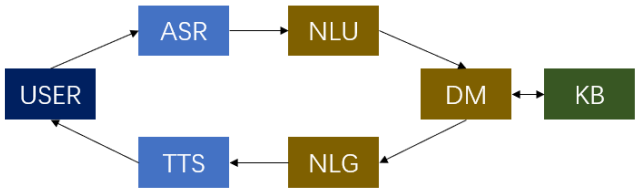

Figure 1.1: Components of traditional pipeline for dialogue systems

Dialogue systems with pipeline require us to predefine structure of the dialogue states, which needs independent components and is complicated and limited to very narrow domain. Moreover, it cannot encode all the features that might be useful. With the development of big data and deep learning techniques, neural network-based dialogue systems trained in an end-to-end and data-driven fashion have attracted more and more attention due to its great potential and application value. On the one hand, deep learning techniques have shown their effectiveness on dealing with unstructured data such as images and sounds. Since natural languages are also unstructured data and share many properties with the others like local correlation, deep learning is expected to be good at dealing with natural languages, including dialogues. On the other hand, deep learning, especially on supervised learning schemes, usually requires large-scaled datasets for capturing complex meaningful feature representations automatically, as well as learning response generation strategies especially for dialogue systems. Nowadays, abundant conversation corpus can be acquired on the Internet, which firmly supports us training complicated neural networks and building open-domain conversation models. A large number of contributions

have been made on promoting the performance of dialogue systems by introducing deep learning and utilizing massive data.

In general, two major approaches have been developed for non-task-oriented systems – (1) generative methods such as sequence-to-sequence models, which generate proper responses during the conversation. Typically, they are more widely used for conversing with human on open domains; and (2) retrieval-based methods, which learn to select responses from the current conversation from a repository and are more usually used on specific scenarios (e.g. online shopping guide).

In this article, we mainly focus on neural network based sequence-to-sequence generative models, which are trained in a completely end-to-end and data-driven fashion, without any hand-coding. Without requirement of manual efforts in rule designing and feature engineering, it is of convenience to develop an extensible open domain conversation system.

The original end-to-end conversation models are inspired by statistical machine translation [18], including neural machine translation [6, 17, 3]. Other works in this direction include Long Short-Term Memory (LSTM) models [37, 19], the Hierarchical Recurrent Encoder-Decoder (HRED) models [30], attention-based models [39, 22, 31], and further the Transformer model [36]. These neural network based approaches produced impressive results on probity and fluency, especially on large-scaled corpus.

Nevertheless, these fully data-driven systems are mainly based on question-response pairs ignoring conversational context and do not have access to any external knowledge (textual or structured), which makes it difficult to respond substantively. While a human focuses on the entities in the questions and thinks a substantive answer about the entities, the neural network-based system only take into account how to make an appropriate response. For example, if the question is "Do you know the best restaurant nearby?", a dialogue system could reply "Yes, I do.", while a human could answer "I would say Yangxiao is the best." Although the former is a correct answer, nobody would be satisfied with it. In contrast, traditional dialogue systems with pipeline can readily inject entities and facts into responses, which indicates that we can combine the advantages of pipeline and neural networks.

The 7th Dialog System Technology Challenge (DSTC7) track 2 proposes an End-to-End conversational modeling task, where the goal is to generate conversational responses that go beyond chitchat in order to produce system responses that are both substantive and “useful” which can contain factual contents, by injecting informational responses. So DSTC7-Track2 provides not only the social conversation corpus but also contextual-related factual texts such as Foursquare, Wikipedia [12, 9]. The results are evaluated by "appropriateness" (e.g. BLEU, METEOR) as well as "informativeness" (e.g. entropy, diversity). Our experiments and evaluations are conducted with this task.

1.1 Goals

The experiment was performed according to the regulations of DSTC7-Track2. We first build own seq2seq model baselines. Besides the baseline model provided by the DSTC organizers, we also used our own primary seq2seq models as control groups.

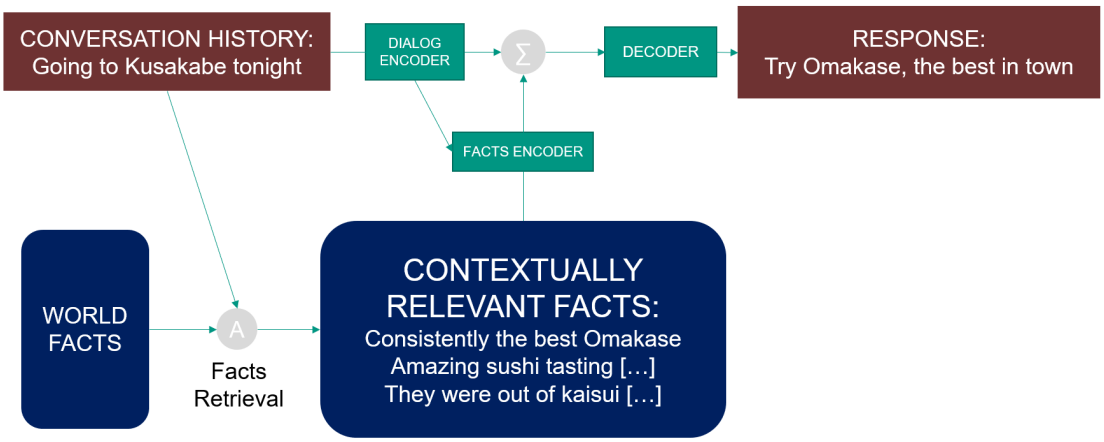

Figure 1.2: Knowledge-grounded model architecture [1]

We can abstract a knowledge-grounded model architecture as Figure 1.2. In many realistic approaches, typically provided by the DSTC competing teams [33, 27, 42, 34], the backbone encoder-decoder architecture are RNN and the facts are represented as sentence-level features. They also made a contribution on promoting the performance by sophisticate data preparations, output inference and candidates reranking. In this article, we focus on fact encoder, which means how the conversation history and the extra knowledge are represented and how they are combined. Since the Transformer model released by Vaswani et al. has shown its high performance [36], we use the Transformer as encoder-decoder backbone. It represents an input sequence as a list of augmented word embeddings. Our proposed method introduces the extra knowledge on word level. Since we make the incorporation on each word, representing the facts and combining it with context using trainable parameters and adjusting the parameters while training would cause massive computation workload. We turned to pre-trained models, BERT [7], which are widely used coupled with fine-tuning on downstream NLP tasks. We put forward 2 schemes: 1. utilize BERT sentence embedding; 2. extract important words from facts and utilize their BERT context sensitive word embedding. Then we combine facts representation and input sequence representation with attention mechanism. Since we aimed to explore the influence by incorporating the facts, we set other condition as invariable, such as decoding strategy and response selection mechanism. The machine systems for comparison are the official RNN baseline, our own RNN and transformer baseline.

To summarize, our work mainly includes:

1. prepare data, make essential data preprocess
2. build stable RNN and Transformer conversation model as baselines, use the Transformer model as backbone of our proposed model
3. select contextually relevant facts from world facts

[1] Source: Galley M, Brockett C, Gao X, et al. End-to-End Conversation Modeling: Moving beyond Chitchat[J]. 2018.

4. convert the contextually relevant facts to proper semantic representation with BERT
5. combine the source sequence representation and the fact representation on word level with attention
6. generate responses and evaluate the results

1.2 Outline

chapter 2 gives an introduction into the fundamentals of this thesis, in particular the Transformer, BERT and TF-IDF. chapter 3 reviews briefly on the development of dialogue system, especially the works that have great significance on this thesis. chapter 4 clearly describe the issue and our target. In chapter 5, we put forward our proposed methods. In chapter 6, we discuss the details of implementation of our experiments and make an evaluation on our proposed methods. In chapter 7, we have a review on our works and discuss the future work.

2 Fundamentals

This chapter provides an essential introduction into the basic terminology, algorithms and models that will be used within this thesis, in particular LSTM, the Transformer, BERT and TF-IDF. This introduction should give a fundamental understanding of the basic concepts and approaches.

2.1 Long short-term memory (LSTM)

Recurrent neural network, RNN, is a type of neural network processing time series data, which is traditionally dominant in NLP tasks with sequences [23]. In theory, RNN can handle sequences of arbitrary length by carefully tuning the parameters, but in practice, RNN does not perform well on long sequence. LSTM is a special type of RNN that can learn long-term dependence information. LSTM was proposed by Hochreiter and Schmidhuber [16] and has been improved and promoted by Felix Gers [11] and Alex Graves [14]. In this thesis, we built a baseline sequence-to-sequence model with LSTM.

LSTM network architecture is composed of a cell (the memory part of the LSTM unit) and three "regulators", usually called "gates", for the information flow inside the LSTM unit: an input gate, an output gate and a forget gate, as shown in Figure 2.1.

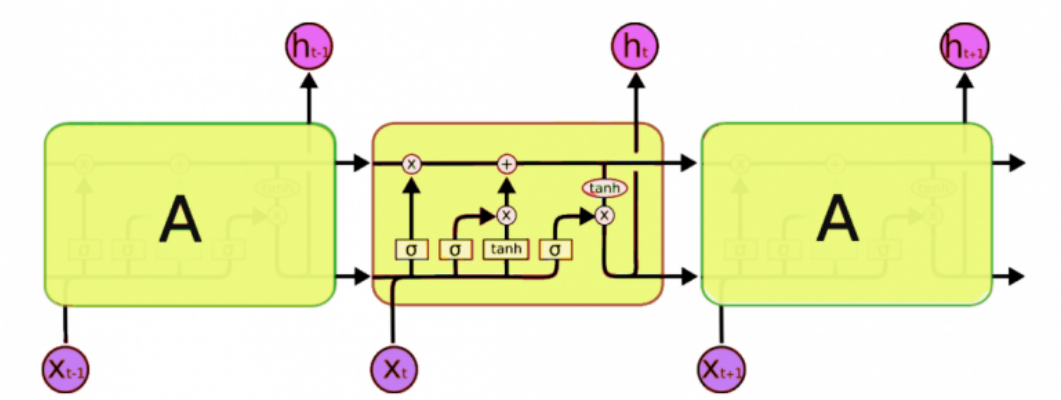

Figure 2.1: LSTM architecture

The network can delete or add information to cell states through gates, which is a combination of a sigmoid layer and a dot product operation, as shown in Figure 2.2.

This form of LSTM can be more precisely defined as follows: h_t is the output of the network at each time step, f_t, i_t and o_t are the forget, input and output gates respectively,

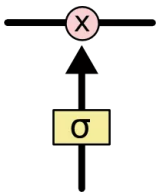

Figure 2.2: Gate for information flow control

c_t is the cell memory state. $\odot$ denotes an element-wise product, [] denotes concatenation.

$$
\begin{aligned}
f_t &= \sigma(W_f[h_{t-1}, x_t] + b_f) \\
i_t &= \sigma(W_i[h_{t-1}, x_t] + b_i) \\
o_t &= \sigma(W_o[h_{t-1}, x_t] + b_o) \\
c_t &= i_t \odot \tanh(W_g[h_{t-1}, x_t] + b_x) + f_t \odot c_{t-1} \\
h_t &= o_t \odot \tanh(c_t)
\end{aligned}
\tag{2.1}
$$

At the price of complexity, LSTM have been shown to be much more resilient to the vanishing gradient problem, as well as being able to model long-term dependencies with much greater reliability than RNN [11, 14].

2.2 Encoder-Decoder Architecture

Encoder-decoder architecture can be regarded as a general framework, under which different algorithms can be used to solve sequence-to-sequence tasks, which means, as literally, input one sequence and output another sequence and the lengths of both input and output sequences are variable. Deep neural networks have achieved good results on issues such as image classification, where input and output can be expressed as a vector of fixed length. If the image size is slightly changed, operations such as zero padding are performed. However, in many NLP tasks, such as machine translation and automatic dialogues, we have only access to sequences without knowing their length in advance. The encoder-decoder framework has emerged as the tasks require.

Encoder - Its role is to transform real world problems into mathematical problems, technically, convert the input sequence into a fixed-length vector.

Decoder - Its role is to solve mathematical problems and turn them into real world solutions, technically, convert the previously generated fixed vector into an output sequence.

Connecting the above two steps, a general diagram is shown as Figure 2.3:

Figure 2.3: Encoder-Decoder

Note that:

1. Regardless of the length of the input and output, the length of the context vector in the middle is fixed.

2. We can choose different encoders and decoders according to the tasks (it can be an RNN, but usually a variant of LSTM or GRU).

We model the scenario as: input sequence in encoder $X = (x_1, x_2, ..., x_m)$, output sequence in decoder $Y = (y_1, y_2, ..., y_m)$, the whole encoder-decoder share an identical context vector $C = G(x_1, x_2, ..., x_m)$. Then we have:

$$\begin{aligned} y_1 &= f(C) \\ y_2 &= f(C, y_1) \\ y_3 &= f(C, y_1, y_2) \\ &... \end{aligned} \tag{2.2}$$

2.2.1 Attention Mechanism

As mentioned in section 2.2, there is only a context vector between encoder and decoder in the traditional form, and the length of the context vector is fixed. It means, encoder compresses all the sequence information into a fixed-length vector. There are two disadvantages: 1. the semantic vector cannot completely represent the information of the entire sequence; 2. the information carried by the content entered first will be diluted or covered by the information entered later.

In order to deal with the above problems, we can use attention mechanism. For example, in machine translation, let the generated words not only focus on the global semantic vector C, but a set of semantic vectors $C_1, C_2, C_3, ...$ that indicates which part of the input sequence should be focused on when outputting words, then based on the area of interest to produce the next output. The model structure is as follows:

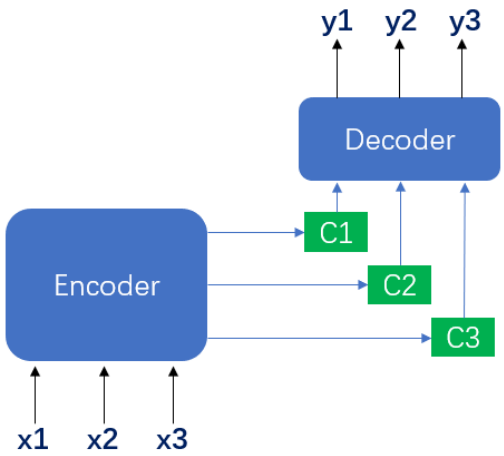

Figure 2.4: Encoder-Decoder with attention

In this form, we generate target sentence Y as follows:

$$\begin{aligned} y_1 &= f(C_1) \\ y_2 &= f(C_2, y_1) \\ y_3 &= f(C_3, y_1, y_2) \\ &\dots \end{aligned} \tag{2.3}$$

More generally, Vaswani et al. [36] gave attention mechanism a clear definition, which plays a significant role on combining different level features. Assume that at the current time step t, we have a query vector and a group of key vectors, a group of value vectors respectively corresponding to the keys. Here query can be understood as a global vector containing more information. We use this query and the keys to weight and sum all value vectors to learn a more suitable new vector, which can be used for tasks such as classification or prediction.

Provided that q_t is the query on time step t. K is the key matrix, among which k_s is a key vector. V is the corresponding value matrix. We first perform a similarity calculation on q_t and each key to get an unnormalized score. For example, Bahdanau attention (additive) [2] and Luong attention (multiplicative) [21].

$$s(q_t, k_s) = Similarity(q_t, k_s) \tag{2.4}$$

The simplest dot product is used here, and the denominator is to adjust the inner product result, so that the inner product is not so large. Then Softmax normalize the score as the attention probability weight, as shown in following formula, where v_s is an element vector in V:

$$a(q_t, k_s) = \frac{\exp(s(q_t, k_s))}{\sum_{i=1}^{N} \exp(s(q_t, k_i))} \tag{2.5}$$

Then we weight and sum the values according to the weight corresponding to each key and get the final output vector:

$$Attention(q_t, K, V) = \sum_{s=1}^{m} a(q_t, k_s) v_s \tag{2.6}$$

We can see that for the same key, different queries will have different output vectors. Conceptually, attention can be understood as selectively filtering a small amount of important information from a large amount of information and focusing on this important information. This focus is reflected by the weight. The larger the weight, the more focused it is on its corresponding value, that is, the weight represents the importance of information, and value is its corresponding information.

2.2.2 The Transformer

The content of following section, including the formulas and the figures, is from the paper of Vaswani et al. [36]. In encoder-decoder framework for general tasks, the input (source) and the output (target) are different. For example, in English-Chinese machine translation,

the source is an English sentence, and the target is a corresponding Chinese sentence. The attention occurs between a query in target and all elements in source, we can consider $key = value \neq query$. Self-attention, as the name implies, refers not to the attention between target and source, but to that occurs between the elements within source or target. It can also be understood as a special attention where $key = value = query$.

With the introduction of self-attention, it is easier to capture long-distance interdependent features in a sentence, because with an RNN or LSTM, we need to calculate the sequence time step by time step. For long-distance interdependent features, it takes several time steps to accumulate information before they can be linked. The longer the distance, the more difficult it is to capture information effectively.

However, self-attention will directly build the relationship between any two words in a sentence through a calculation step, so that the distance between long-distance dependent features is greatly shortened, which is conducive to utilize these features effectively. In addition, self-attention also directly helps to increase the parallelism of the calculation.

Vaswani et al. [36] put forward a new Encoder-Decoder Architecture relying entirely on self-attention to compute representations of its input and output without using RNN, which is called the Transformer [36], as illustrated in Figure 2.5.

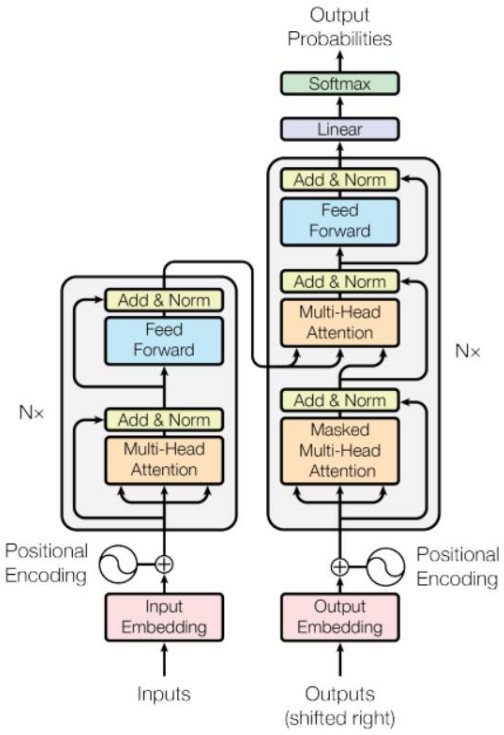

Figure 2.5: The Transformer - model architecture [1]

Multi-head - besides extracting features with self-attention mentioned above, the Transformer contains multi-head attention. The multi-head refers to introduce multiple sets of

[1]Source: Vaswani A, Shazeer N, Parmar N, et al. Attention is all you need[C]//Advances in neural information processing systems. 2017: 5998-6008.

parameter matrices to perform linear transformations on Q, K, and V and obtain the self-attention respectively, then concatenate all the results as the final self-attention output.

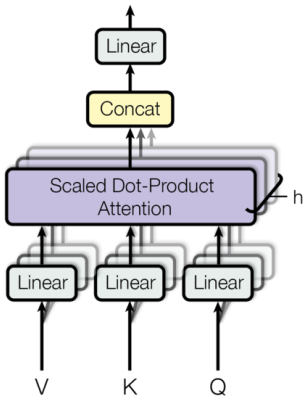

Figure 2.6: Multi-head attention [1]

More precisely, multi-head attention can be calculated as follow:

$$\begin{aligned} MultiHead(Q, K, V) &= Concat(head_1, ..., head_h)W^O \\ head_i &= Attention(QW_i^Q, KW_i^K, VW_i^V) \end{aligned} \tag{2.7}$$

In this way, the model has multiple sets of relatively independent attention parameters, which theoretically can enhance the capabilities of the model.

Positional encoding - without the introduction of position information we cannot model a sequence properly. If we shuffle the word order in the sentence, the result of attention is still the same. The Transformer introduces position information by giving each position a number, where each number corresponds to a vector. By combining the position vector and the word vector, a certain position information is introduced for each word, so that attention can distinguish words in different positions. In the Transformer, sine and cosine functions of different frequencies are used for encoding positions:

$$\begin{aligned} PE_{(pos,2i)} &= \sin(pos/10000^{2i/d_{model}}) \\ PE_{(pos,2i+1)} &= \cos(pos/10000^{2i/d_{model}}) \end{aligned} \tag{2.8}$$

The core idea of the attention mechanism in the Transformer is to calculate the relationship between each word in a sentence and all the words in the sentence. Using these correlations to adjust the importance (weight) of each word can obtain a new representation for each word. This new representation not only contains the word itself, but also the relationship between other words and this word, so it is a more global expression than a simple word embedding vector. Attention reduces the distance between any two positions in the sequence to a constant, so that reduce long term dependence problem. In this thesis, we build our proposed model based on the Transformer architecture.

2.3 Bidirectional Encoder Representation Transformers (BERT)

In NLP tasks, a word or a sentence usually needs to be transform into semantic representation with respect to the context around it. Machine learning models, especially neural networks, cannot directly deal with unstructured information such as texts, images and videos, which have variable size. We must first convert the information into numerous representation, in other word, a group of numbers, which is usually a scalar, a vector or a matrix, so that the machine learning models can recognize them and use them in calculation. This semantic representation is usually called embedding or encoding. Pre-trained models are increasingly popular for obtaining embeddings in order to utilize the massive data from the sophisticated datasets as well as reduce computation workload. In this thesis, we use pre-trained model BERT to generate word embeddings and sentence embeddings for the contextually relevant facts.

BERT, Bidirectional Encoder Representation Transformers [7], can be considered as a general NLU (Natural Language Understanding) model, which provides support for different NLP tasks. It is built based on bi-directional Transformers. In actual use, we only need to add an output layer for fine-tuning according to specific tasks, instead of modifying the model structure for specific tasks. This is an advantage of the pre-trained BERT model.

The BERT model structure is an optimization of the OpenAI GPT model [31], and changed its uni-directional Transformer structure to bi-directional. The paper of Devlin et al. [7] argues that unidirectional language models severely limit the ability to pre-train expressions, especially for fine-tuning methods. Therefore, the BERT model uses a two-way language model, and has obtained a huge capacity improvement. The author of BERT proposed to use MLM (masked language model) to train language models. When entering a sentence, randomly select some words to predict, and then replace them with a special symbol. Although the model eventually see the input information in all positions, but, because the words to be predicted have been replaced by special symbols, the model cannot know in advance what words are in these positions, so that the model can learn which word should to be filled in based on the given labels.

However, the special symbol we use in the pre-training process will not appear in following concrete tasks. In order to be consistent with following tasks, the author enters the original word or a random word at a certain proportion of the positions to be predicted. Because only part of the words in the input text sequence are used for training, the efficiency of BERT is lower than that of ordinary language models. The author also pointed out that the convergence of BERT requires more training steps.

Another innovation of BERT is to add a next sentence prediction (NSP) task based on two-way language model. It is to predict whether the text at both ends of the input BERT is continuous text. The author points out that the introduction of this task let the model better learn the relationship between continuous text fragments. During training, the second segment of the input model will be randomly selected from all the text with a 50% proportion, and the remaining 50% proportion is the subsequent text of the first segment.

BERT training data uses English open source corpus "Book Corpus" and English Wikipedia data, in total of 3.3 billion words. The basic version of BERT model has 100 million parameters, while the large version has more than 300 million parameters.

The basic structure of the BERT model is shown in Figure 2.7. In essence, a multi-layer bi-directional encoder network is constructed using the Transformer structure. We only need to add an output layer after the structure in Figure 2.7 according to the specific task.

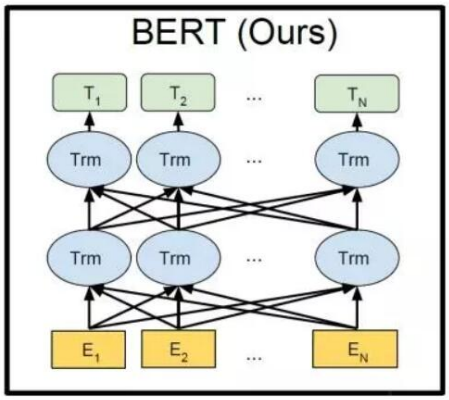

Figure 2.7: BERT encoder [2]

The pre-trained BERT model can be directly applied to a variety of tasks, including sentence-level classification, question answering, sequence labeling. The following figure illustrates single-sentence classification tasks.

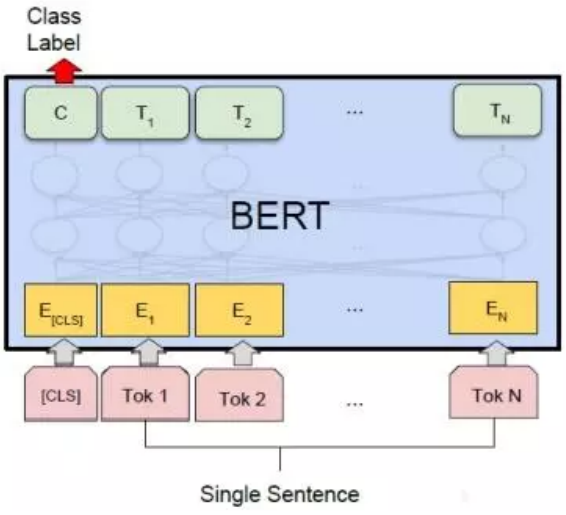

Figure 2.8: Single sentence classification with BERT [3]

[2]Source: Devlin J, Chang M W, Lee K, et al. Bert: Pre-training of deep bidirectional transformers for language understanding[J]. arXiv preprint arXiv:1810.04805, 2018.

[3]Source: https://jalammar.github.io/illustrated-bert/

BERT has refreshed the records in 11 NLP tasks [7] and it played a significant role in our experiments. We use BERT to represent contextually relevant facts and the important words among them.

2.4 TF-IDF

Tf-idf, short for term frequency–inverse document frequency, is a widely used weighting technique for information retrieval. In this thesis, we used it for retrieving contextually relevant facts from world facts and selecting the most important words. Tf-idf measures the importance of a word to a document in a corpus.

On the one hand, if a word is important, it should appear multiple times in this document. Therefore, we make "term frequency" (abbreviated as tf) statistics. On the other hand, if a word is relatively rare in the corpus, but it appears in this document, it probably reflects the characteristics of this document and is the key word we need. In other word, based on the document frequency, each word is assigned a "importance" weight. The common words like "the", "is" and "in" are given less weights, while the less common words like "reinforcement learning" and "recommender systems" are given larger weights. This weight is called "inverse document frequency" (idf).

Given a corpus C containing n_d documents, a document d and a term t, one of the possible definitions of tf-idf is calculated as follow:

$$\begin{aligned}
&\text{tf-idf}(d,t,C) = \text{tf}(d,t) \times \text{idf}(t,C) \\
&\text{tf}(d,t) = \text{number of occurrences of } t \text{ in } d \\
&\text{idf}(t,C) = \log \frac{n_d}{\text{df}(t,C)} \\
&\text{df}(t,C) = \text{number of documents in } C \text{ where } t \text{ occurs}
\end{aligned} \tag{2.9}$$

Add-one smoothing is widely used in idf to avoid zero denominator problem:

$$\text{idf}(t,C) = \log \frac{n_d}{\text{df}(t,C)} + 1 \tag{2.10}$$

2.5 Evaluation Metrics

For measuring appropriateness, BLEU [25], NIST [8] and METEOR [4] are used in our thesis.

2.5.1 BLEU

The idea of BLEU [25] is to compare the overlap of n-grams (from unigram to 4-gram in practice) between candidate responses and reference responses. The higher the overlap, the higher the quality of the translation. For a source sentence, we mark candidate response as c_i and a set of reference response as $S_i = \{s_{i1}, s_{i2}, ..., s_{im}\}$, where s_{im} is a reference response example. ω_k is an n-gram, $h_k(c_i)$ is the number of occurrences of ω_k in c_i,

while $h_k(s_{ij})$ is is the number of occurrences of ω_k in s_{ij}. BLEU calculates the coincidence precision according to the following formula:

$$CP_n(C,S) = \frac{\sum_i \sum_k min(h_k(c_i), max_{j \in m} h_k(s_{ij}))}{\sum_i \sum_k h_k(c_i)}$$

$CP_n(C,S)$ prefers short response. BP (Brevity Penalty) is introduced to compensate this trend, where l_c is the length of c_i, l_s is the length of s_{ij}:

$$BP(C,S) = \begin{cases} 1 & \text{if } l_c > l_s \\ e^{1-\frac{l_s}{l_c}} & \text{if } l_c \leq l_s \end{cases}$$

BLEU is a weighted geometric mean of n-grams precision, as the following formula:

$$BLEU_N(C,S) = BP(C,S) \exp(\sum_{n=1}^{N} \omega_n \log CP_n(C,S))$$

Unigram precision can be used to measure the accuracy of word translation, while higher-order n-gram precision can be used to measure the fluency of sentences. BLEU is more suitable for evaluating machine translation than dialogue. It only regards precision but a lot of high-quality conversation responses have low precision with the references, which has been reflected in [5]. Besides, BLEU tends to shorter candidates despite brevity penalty. The shortages of BLEU were as well reflected in some other works: [24] shows that BLEU does not map well to human judgements in evaluating some natural language generation tasks, [32] shows that BLEU does not reflect meaning preservation very well.

2.5.2 METEOR

Compared to BLEU, METEOR [4] reintroduces recall and uses F-Score as metric combining recall and precision. It looks only at unigram precision and recall and takes into account word inflection variations (via stemming) while matching. When evaluating fluency, the concept of chunk was introduced. The calculation of METEOR requires a set of alignment m in advance, and this calibration is based on WordNet, which is obtained by minimizing the continuous and ordered chunks ch in the corresponding sentence. The smaller the number of chunks, the longer the average length of each chunk, which means the more consistent the word order of the candidate and reference responses.

2.5.3 NIST

NIST [8], short for National Institute of standards and Technology, is an improvement on BLEU. While BLEU simply gives an equal weight to each n-gram and calculates n-gram precision, NIST calculates how informative an n-gram is as well, introducing the concept of $Information$ to every n-gram. When a correct n-gram is matched, it would be given a higher weight if the n-gram is relatively rarer. For example, the words (unigram) "a", "the" are usually given a lower weight while the bigram "carbon dioxide" is more likely to get a higher weight. Besides, NIST does not involve with brevity penalty. Several works [24, 15, 35] made comparison between BLEU, METEOR and NIST and have shown that NIST and METEOR are more reliable for evaluating some NLG tasks.

3 Related Works

This chapter briefly reviews relevant researches of our work, mainly about how other researches utilizing external knowledge.

An intuitive idea to improve response quality is to take into account prior external knowledge relevant to the conversation, which is a major difference between human and conversational agent trained with source-target pairwise corpus. With the development of name entity recognition (NER) and information retrieval (IR), it is of convenience to find and extract the information relevant to the conversation so that build an external knowledge base (KB). The incorporation of knowledge base has been explored for other NLP tasks. [41] used both the words from knowledge base and the common words in vocabulary for question-answering (QA) tasks, instead of tuple retrieval in knowledge base in generation process. [29] proposed Pointer-Generator Networks. It introduced a mode control mechanism to switch between copying tokens from knowledge base or input sequence and generating a new token, which enjoys the advantages of both extractive and abstractive summarization.

3.1 Dialog System Technology Challenges 7

Many efforts have been made to incorporate the non-conversational facts into fully data-driven dialogue systems. In the sentence generation task in Dialog System Technology Challenges 7 (DSTC7-Task2), this issue was discussed in focus.

In [12], [42] and [34], memory network is introduced to represent fact sentences and couple them with sequence-to-sequence backbone. [27] used a hierarchical architecture to encode input sequence and fact sentences as RNN hidden states respectively and combine the hidden state using attention mechanism. All the competition related works represented the input sequence and the external knowledge at sentence-level as a vector and as well manipulated the context vector at sentence-level. They all used RNN Encoder-Decoder architecture as the conversation backbone.

Our experiments and evaluations are conducted with this task and we proposed a model based on the Transformer and introduce fact information into word embeddings in dialogue encoder.

4 Problem Description

In this section, we will clearly elaborate on the issue that this thesis aims at.

Our works target at improving the performance of neural network-based conversational response generation in single turn so that the generated response can be more informative and contentful.

Figure 4.1: Comparison between responses from human and neural model [1]

We illustrate it with the example in Figure 4.1. A normal neural conversation model trained with source-target pairwise conversational corpus has no access to the tips on top of the figure. In the conversation corpus, the sentences like "XXX go to XXX" might occur several times in source sequences, while the corresponding targets might usually include "Have a good time". After training, the neural model remembers the relationship that when the input has similar implication with "XXX go to XXX", the response is highly probable to be "Have a good time". Comparatively, "Kusakabe" is very rare in corpus, and the model does not have enough references to figure out how "Kusakabe" has an influence on the response. When the user inputs "Going to Kusakabe tonight.", the system recognizes the query as "XXX go to XXX" ignoring "Kusakabe", and hence gives the response "Have a great time!". If a human who has completely no impression on "Kusakabe" is asked the same question, he might probably answer "Have a great time!" as well, because

[1]Source: Galley M, Brockett C, Gao X, et al. End-to-End Conversation Modeling: Moving beyond Chitchat[J]. 2018.

he might realize the question is "somebody goes to somewhere" and such answer is proper in this situation. Thus, the response of the neural model deserves to be called "proper". However, if a human know "Kusakabe" is a Japanese restaurant and "omasake" is a famous dish in Kusakabe by reading the tips, he might realize the query is "go to a restaurant" and politely give a suggestion using the knowledge from the tips, sothat the response is comparatively substantive and contentful. In this scenario, the difference between the responses from neural model and human can be mainly ascribed to the association with the tips, in other word, external knowledge. It is to expect that if neural models had access to the external knowledge, their response might contain substantive information and concrete entities as well. This goal is same as that of grounded response generation task at Dialog System Technology Challenge 7 (DSTC7-Track2). Thus, we conducted our experiment on the condition of DSTC7-Track2.

5 Methods for Knowledge-grounded Conversation Models with Word-level Incorporation

In this chapter, we will first illustrate the common architecture for the issue in the first section. Then we will put forward our methods including facts retrieval, generating semantic representation for the contextually relevant facts, incorporating fact information into dialogue encoder and response selection in decoder, while introducing the concrete methods used in each part of our own architecture.

5.1 Common Architecture

A neural conversation model architecture associated with non-conversational external knowledge can be abstracted as Figure 5.1. In the following section, we will describe a comparison between such architecture and a normal sequence-to-sequence backbone.

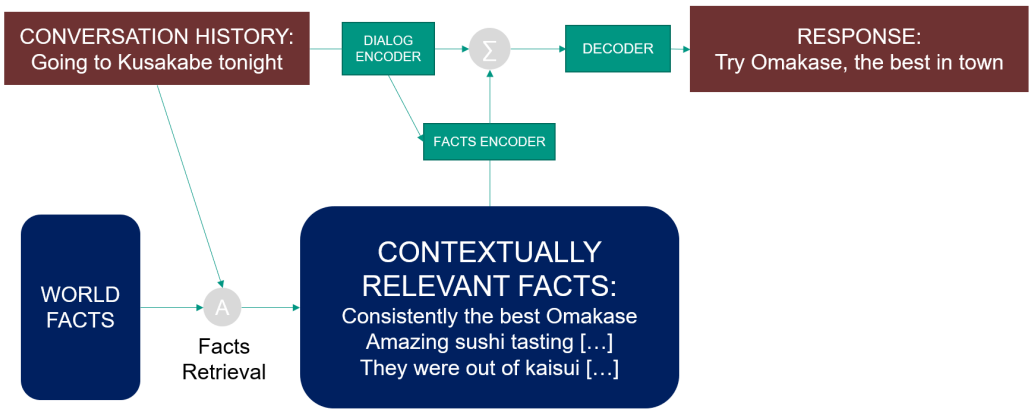

Figure 5.1: Knowledge-grounded model architecture [1]

First, a knowledge base that contains world facts is available. In this thesis, world facts are non-conversational texts that are organized as a group of sentences or short paragraphs. When given a sequence input or conversational history, we have a facts retrieval component for searching the most relevant facts according to the important part

[1]Source: Ghazvininejad M, Brockett C, Chang M W, et al. A knowledge-grounded neural conversation model[C]//Thirty-Second AAAI Conference on Artificial Intelligence. 2018.

in the input. Such important part is usually substantive or concrete things like names, countries and institute. With a dialogue encoder, conversation history is encoded into semantic representation like in normal sequence-to-sequence model. However, instead of feeding the semantic representation into decoder, the architecture first feeds it to a facts encoder. With the facts encoder, the important part of contextually relevant facts are recognized according to the semantic representation of input sequence and encoded into semantic representation, which is usually in the same form with that of input sequence, as well. Then a component combines the semantic representation of input sequence and contextually relevant facts and feeds its output to a decoder. The decoder makes inference like that in normal sequence-to-sequence model and generates the final response. s

5.2 Facts Retrieval

In our experiment, we used one of the most commonly used weighting technique for information retrieval, tf-idf, mentioned in section 2.4, for filtering the contextually relevant facts.

Tf-idf is very flexible in use. In order to find the most relevant facts given a input sequence with tf-idf. There are two approaches used in our experiments:

1. We first filter the important words in the query by blocking the "stop words" like "a", "the" and "is", then calculate the tf-idf score of each important word in every document. For a document, we consider the summation of all the tf-idf scores of the important words as the relevance index of the query to the document. We calculate the relevance between a query and a document from a corpus as the following formula, where q is a query, d_i is a document, C is a corpus, t_j is a "non-stop-word":

$$\text{Relevance Index}(q, d_i) = \sum_{t_j} \text{tf-idf}(d_i, t_j, C) \tag{5.1}$$

We regularized term frequency with the length of the document, because we should prevent the trend from favoring long documents.

2. Since tf-idf reflects the importance of a word to the document, we can use tf-idf to find the most important words in each document. Then we form a term frequency vector with the important words for each document, where the element is the number of the occurrence of each important word in the document. We then calculate the similarity, such as cosine similarity, between the vectors and so on search the most relevant documents to a given query as we consider that higher similarity value two documents have, more relevant they are.

$$\begin{aligned}
&\text{sentence A: I have money}\\
&\text{sentence B: I have no money}\\
&\text{term frequency vector } A\text{: [1,1,1,0]} \quad \text{I:1, have:1, money:1, no:0}\\
&\text{term frequency vector } A\text{: [1,1,1,1]} \quad \text{I:1, have:1, money:1, no:1}\\
&\text{cosine similarity: } \cos\theta = \frac{1\times 1 + 1\times 1 + 1\times 1 + 0\times 1}{\sqrt{1^2+1^2+1^2+0^2}\times\sqrt{1^2+1^2+1^2+1^2}} = 0.866
\end{aligned} \tag{5.2}$$

In our experiments, tf-idf is used for looking for the important words in contextually relevant facts as well. For each contextually relevant fact, we kept 2 most important words among it. If an important word occurs multiple times in the fact sentence, we take into account all of them, using the average of their embeddings to represent this important word. More detail will be mentioned in section 5.4.

5.3 Dialogue Encoder and Decoder

The Transformer, a new type of encoder-decoder architecture published by Vaswani et al. [36], strongly draws public attention. It enjoys a lower computation workload and is good at dealing with long-term dependency problem compared to the RNNs, so that it can use more network layers to capture semantic implication and be applied to longer sequences. Since there are a lot of long sentences (about 60 tokens) in the corpus and the Transformer has shown its high performance, we used the Transformer as the encoder-decoder backbone, unlike other works published for DSTC7-Track2 [33, 27, 42, 34, 10]. That means, our dialogue encoder represents the input sequence as a matrix or more precisely, a list of augmented word embeddings, which is the output of the last self-attention block. The number of columns of the semantic representation matrix is same as the length of the input sequence, while in RNN models an input sequence is represented as a fixed-length vector. That avoids the loss of information by compressing the variable-length input sequence into a fixed-length vector. It is illustrated in Figure 5.2:

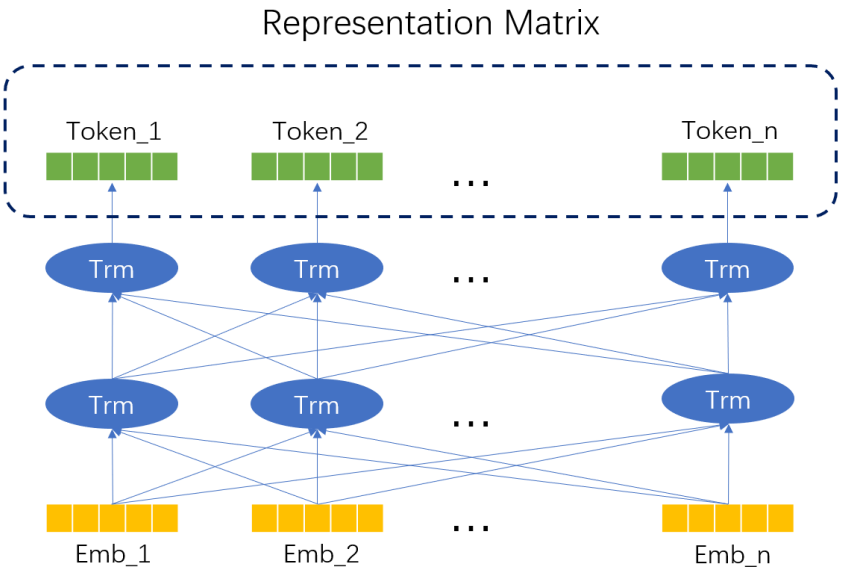

Figure 5.2: Semantic representation of input sequence

In order to make a more comprehensive comparison, besides referring to the official baseline, we build our own baseline models in addition. We build not only a Transformer-

based baseline, but also a traditional RNN encoder-decoder baseline, more precisely, with long short-term memory (LSTM).

The decoder makes inference with the same mechanism of that in normal sequence-to-sequence model, but its input is the combination of semantic representation of input sequence and contextually relevant facts. Beam search and some other tricks is used to improve the quality of the generated response, which will be mentioned in detail in chapter 6.

5.4 Facts Encoder

This part is the focus point of this thesis. In the following 2 subsections, we will mention how we encode the contextually relevant facts into semantic representation and how we combine it with the semantic representation of the conversation history respectively.

5.4.1 Representing Facts with Pre-trained Models

The first task of facts encoder is to filter the important part of the given contextually relevant facts which is non-conversational texts and organized as a set of sentences or short paragraphs with respect to the given semantic representation of input sequence and encode the important part into semantic representation as well. The order can differ as well, first encode the facts and then filter the important part.

In some DSTC7-Track2 relevant works [13, 34, 42], a fact sentence was represented as bag-of-words, masking the less important "stop words" like "a", "the" and "is" in sentences and forming a vector whose elements are the numbers of occurrences of the key words. This method does not introduce trainable parameters, which enjoys a low computation workload. It emphasizes the important part of the fact sentences well, since only the key words that occurs relatively more times are involved in the bag-of-word vectors. However, it loses the implication of word order and ignores the internal dependencies between different parts of the sentences. In other DSTC7-Track2 relevant works [27, 33], a fact sentence was represented as hidden state vectors of RNNs, which are in same form with the input sequence representation. This method utilizes the implication of word order and the internal dependencies, but does not emphasize the important part explicitly and requires a huge amount of computation, since it can be considered as a multi-channel RNN architecture and introduce massive trainable parameters.

In order to better emphasize the important part, utilize the implication within sentences and reduce computation workload, we need a novel word or sentence embedding approach. In this thesis, we turn to a pre-trained natural language understanding (NLU) model, BERT, Bidirectional Encoder Representation Transformers [2]. The methods that encode words or sentences with BERT and then use the embedding for fine-tuning for other downstream tasks have been proven to be effective [7]. We used the single-sentence mode of BERT, adding a "CLS" token at the front of a fact sentence and then feeding it to BERT. With BERT we can convert a sequence to a vector containing the information of the whole sequence, which is aligned with the "CLS" token and a list of vectors whose

[2] See section 2.3

elements are the augmented word embeddings of the corresponding tokens, through the pre-trained parameters in BERT models. We put forward two schemes with BERT:

Scheme 1: Utilize the whole fact sentences. We can use the vector corresponding to the "CLS" token to represent the whole fact sentence. This method have shown its effectiveness [7]. Thanks to the extremely deep network architecture (24 layers) of BERT and the self-attention mechanism, the implication within fact sentences is well exploited and the important part is well emphasized. The extraction of sentence embedding is shown in Figure 5.3.

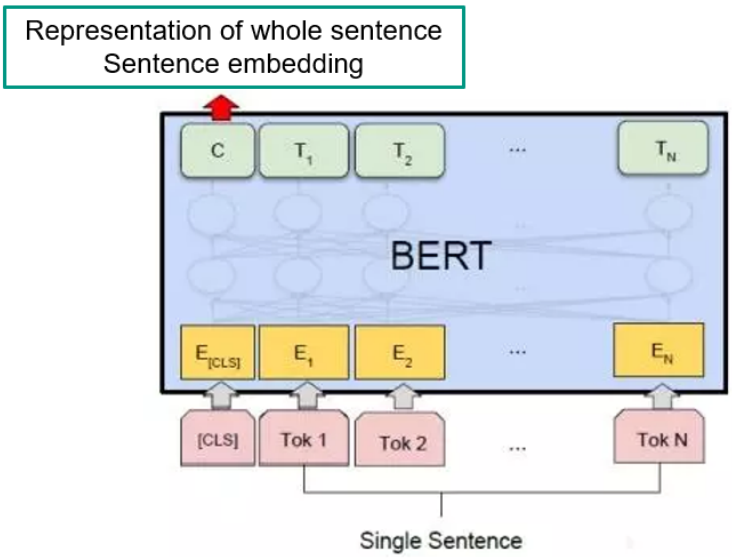

Figure 5.3: Sentence embedding with BERT [3]

Scheme 2: Utilize the important words in fact sentences. In the fact retrieval step, we can find not only the contextually relevant facts, but also the most important and informative words in the facts with tf-idf. In order to better emphasize the important part of the facts, we can use the augmented word embeddings of the important words from BERT to represent the facts. Thanks to the word disambiguation by self-attention and continuous bag-of-word (CBOW) mechanism in BERT, an augmented word embedding contains not only the meaning of the corresponding word itself, but also the information of surrounding words. It does not lose too much sentence-internal dependency and implication if we set a proper sampling window around a word. For each fact, we sample two key words from it. The architecture is shown in Figure 5.4.

[3]Source: https://jalammar.github.io/illustrated-bert/

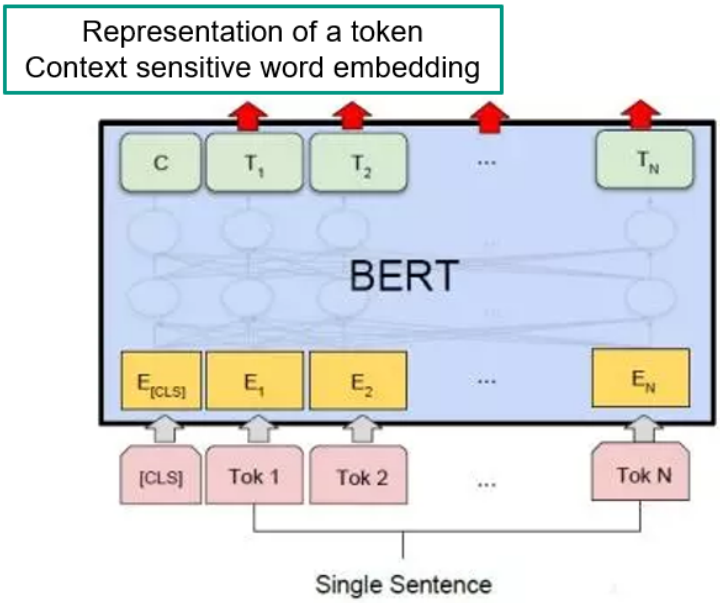

Figure 5.4: Word embedding with BERT [3]

Note that BERT uses WordPiece algorithm to reduce the size of vocabulary, which splits rare words into sub-word characters. It is a trade-off between sequence length and vocabulary size. In order to couple BERT with the selected important words and word-level incorporation in next step, we use the average of the representation vectors of the sub-words to represent the complete word. It is illustrated as Figure 5.5.

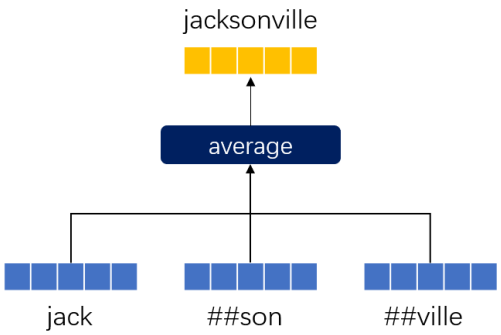

Figure 5.5: Merge pieces of sub-words

If a word is considered as important word and occurs multiple times in the fact sentence[4], we use the average of the BERT embeddings of these words to represent it. Note that BERT takes into account the surrounding context of a word and transforms an identical word into different embeddings so that achieves word disambiguation. It is illustrated as Figure 5.6.

[4]See section 5.2

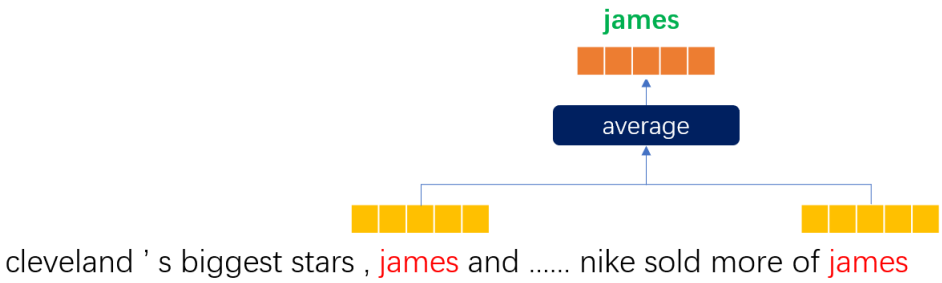

Figure 5.6: Merge multiply occurring important words

The computation of BERT is in fact quite time-consuming, since it makes inference with an extremely complicated model. We can first obtain the encoded fact representations and store them on disk before the end-to-end training with conversation corpus, so that avoid the time consuming for obtaining BERT embeddings. We only need to load the stored representations from disk while training.

5.4.2 Combining embeddings with Attention Mechanism at word-level

Since our dialogue encoder represents the input sequence as a matrix or more precisely, a list of word embeddings, the output of the last self-attention block in dialogue encoder, the semantic representation matrix has the same length with the input sequence instead of a fixed-length vector in RNNs. It is difficult to incorporate the semantic representation of facts into conversation history at sentence-level, because the input sequence is represented as a matrix in which the number of column is same as the length of input sequence, which is variable, and each piece of fact information (word embedding of important word or sentence embedding of fact sentence [5]) is represented as a fixed-length vector. On the one hand, we could not compress the matrix into a vector, because the input of decoder is as well a matrix with same shape. On the other hand, if we intend to map the vector into a same-shaped matrix, a large amount of trainable parameters are required to deal with the variable-length input sequence, which is difficult to implement.

Instead, we proposed a method that incorporates the semantic representation of facts into conversation history at word-level. More precisely, given vectors of encoded fact (word embeddings or sentence embeddings), we manipulate each word vector in the input sequence matrix respectively according to the given vectors. The fusion of dense vectors at same or different feature-level has been explored and proven to be effective. In terms of word or sentence embedding, [36] introduced positional encoding into word embedding without trainable parameters, [7] combine word embedding, segment embedding and positional embedding with trainable parameters. More general in multi-model fusion, [40] introduced video features into encoder-decoder frame with vector fushion. These works inspired us to introduce fact information into each word vector in the input sequence matrix.

[5] See subsection 5.4.1

Typically, addition and concatenation are widely used for combining dense vectors. Assume that there is an n dimensional vector a and an m dimensional vector a. We can simply append b after a, or first map b into an n dimensional vector b' with a matrix W sized $n \times m$ and then perform vector addition $a + b'$. As shown in some experiments [12], with addition we could usually obtain better results. Thus, we focus on combing with addition. In our experiment, vectors from dialogue encoder and from facts encoder are set to have same length. We do not need the matrix W and just perform $a + b$.

Assume that t_i is the word vector on previous time step and $f_1, f_2, ..., f_k$ are the vectors from facts encoder (sentence vector in Scheme 1, word vector in Scheme 2, see subsection 5.4.1). In order to focus more the important information in $f_1, f_2, ..., f_k$ with respect to the relevance between t_i and f_j, $j = 1, 2, ..., k$, we need a set of scalar weights ω to emphasize the facts more relevant to t_i.

When it comes to the determination of weight parameters, it is natural to associate attention mechanism [6]. We consider t_i is a query and f_j serve as both key and value. The method that let key and value have a same source is widely used in NLP tasks [39, 36]. Dot production noted as p_j between key f_j and query t_i is calculated and considered as the relevance score between f_j and t_i. Then Softmax is calculated among p, so that we get a set of weight parameters ω. It is shown as the following formulas:

$$\begin{aligned} p_j &= f_j \cdot t_i \\ \omega &= \text{Softmax}(p) \end{aligned} \tag{5.3}$$

With attention mechanism we get rid of high computation workload introduced by using trainable parameters to determine ω. The process is illustrated in Figure 5.7, where "Kusakabe" is a word in input sequence and now serves as a query q_1. k_1, v_1, k_2 and v_2 are word or sentence embeddings of the facts from facts encoder and serve as key and value. k_1 and v_1 are same and from an identical fact, while k_2 and v_2 are same and from another fact. After attention, fact information is weighted and injected into "Kusakabe", so that an augmented word embedding "Kusakabe'" containing not only information from itself but also information from facts is obtained.

[6] See subsection 2.2.1

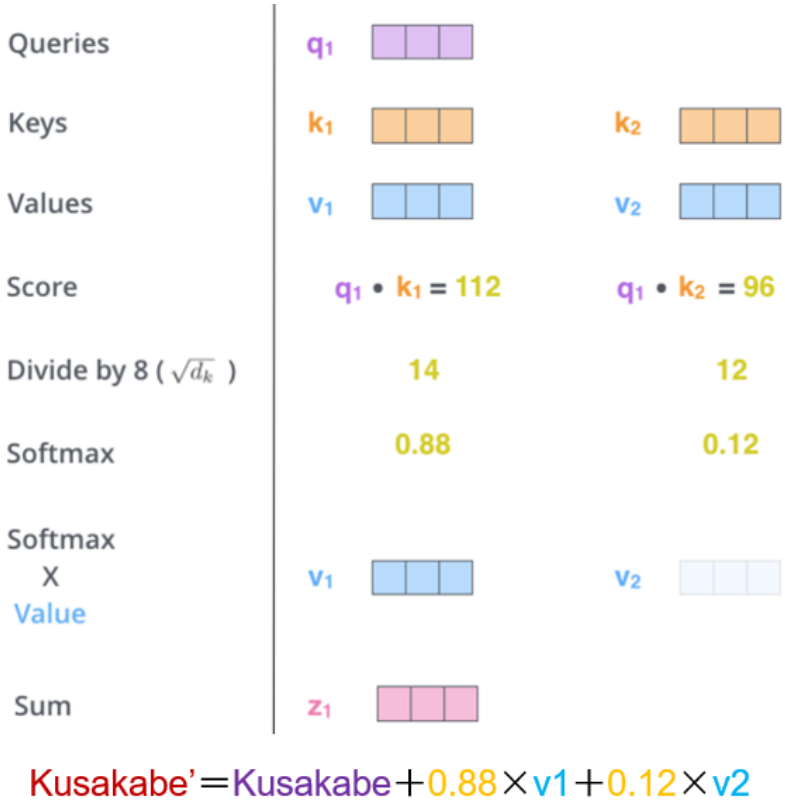

Figure 5.7: Processes of weighting fact information with attention [7]

Such combination via attention mechanism was performed at each word embedding of input sequence from dialogue encoder, so that an augmented semantic representation of conversation history containing information from facts is obtained and then is fed to the next step of the architecture.

5.5 Response Selection

Usually, after the decoder complete beam search, a bunch of candidate responses are generated and we need to choose one among them as the deterministic response for the given input sequence. The DSTC7-Track2 is no exception, which requires only one response for a source sequence. In conversational response generation tasks, we seldom choose the first one from the beam, which has minimum loss and is assigned highest probability, because it is usually a short, trivial sentence like "Yes ." and "OK ." A reason could be, the sequence-to-sequence models typically optimize the likelihood of outputs during training and the general responses do appear more frequently in the conversation corpus, it is natural that the general and trivial responses would be assigned high probability [19, 38]. In order to improve quality of the generated responses, we could use some strategies like selecting the longest candidate, the last candidate or the candidate containing

[7] Source: https://jalammar.github.io/illustrated-transformer/

most various words from the beam. In our experiment, we chose the candidate containing most various words as the deterministic response for the given input sequence, only for our proposed model, but also for our own baseline Transformer model and baseline RNN model.

6 Experiments

This chapter is about the implementation of our experiments: section 6.1 introduces how we obtained the dataset and the efforts to make it suitable for training, section 6.2 mentions our experimental setup, including how we build the concrete framework, the setup of hyper-parameters and the baseline we used and section 6.3 mentions the evaluation metrics. The points worth of notice will be indicated.

6.1 Dataset

Our works target at improving the performance of neural network-based conversational response generation in single turn so that the generated response can be more informative and contentful. This goal is same as that of grounded response generation task at Dialog System Technology Challenge 7 (DSTC7-Track2). Thus, we conducted our experiment on the condition of DSTC7-Track2, which means that we used only the dataset (conversation and external knowledge) provided by the organizers to train our model and evaluated the performance by testing with the official test set. The data collection and the final calculation of evaluation score were executed with the official scripts [1].

Since it was a competition, the organizers did not provide an elaborate dataset like the Ubuntu Dialogue Corpus [20]. Instead of directly releasing the data, the organizers provided the scripts for crawling data. The data preprocess is crucial. Without fine preprocess we could even hardly train a baseline sequence-to-sequence model.

6.1.1 Conversation Data

The conversational data is from a Reddit dump [2]. Reddit [3] is a network of communities, where people can discuss about a topic on posts and the content can be voted up or down by other people. Posts are organized by topics called "subreddits". The facts data is from Common Crawl [4]. The source of facts are information sharing websites like Wikipedia [5].

In our conversation corpus, each turn conversation is equipped with the subreddit name indexed by a conversation ID and other information, as shown in Table 6.1.1. For training our models, we only need the conversation ID (for looking for the relevant facts)

[1] https://github.com/mgalley/DSTC7-End-to-End-Conversation-Modeling, DSTC7: End-to-End Conversation Modeling
[2] http://files.pushshift.io/reddit/comments/
[3] https://www.reddit.com/
[4] http://commoncrawl.org/
[5] https://www.wikipedia.org/

	Training set	Dev. set	Test set
# dialogue turns	2,364,228	119,478	13,440
# facts	15,180,800	1,675,056	582,944
# tagged facts	2,185,893	369,423	139,406

Table 6.1: Data statistics of conversation and facts corpus

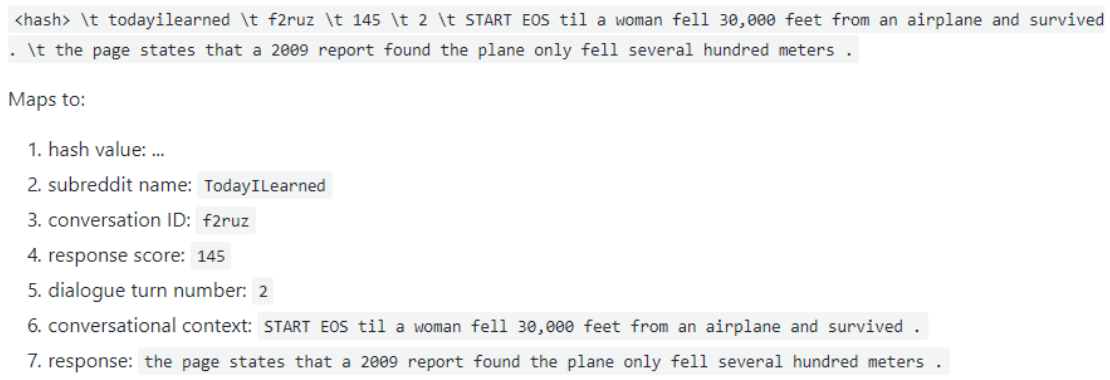

Figure 6.1: Sample conversation turn

and the conversational context and response pair. We did not regard the turns because we aimed at single-turn response generation and considered all the previous turns as conversation history. For an identical conversational context, there might be several responses, which has different response scores. We found that only the responses with best score are not sufficient for training and many responses with lower scores are of high quality in grammar and content, so we collected all the dialogue turns in our rudimentary conversation dataset.

Since the conversation is from a social media source, it contains extremely various words and a lot of noise, which requires data preprocess. We must reduce the vocabulary, because too large vocabulary is not beneficial for fully data-driven neural conversation models and we used pre-trained model BERT, whose vocabulary is limited. First, all the words were lowercased and all the non-letter characters were eliminated except the basic punctuation marks. The words or phrases containing too special information like telephone number, website link and date are replaced with corresponding special tokens like <number>, <link> and <date>. Then some tools like spaCy and our own rules were applied to the dialogues for regularizing the words. For example, "u . s ." is regularized as "u.s." and "don't" is converted to "do n't". The repeating tokens in corpus were then removed, in order to avoid them causing repeating tokens in generated responses. After the manipulation, we chose the most 20000 frequent words to form our vocabulary.

On the one hand, too short sequence might not be informative and contentful and it is not beneficial for training a conversation model aiming at generate long and vivid responses. On the other hand, although our sequence-to-sequence backbone and BERT are based on the Transformer, which is relatively good at dealing with long sequence,

there is still a limit. Too long sequence might be beyond the capacity of the Transformer. Therefore, we kept only the turns whose source and target sequence have both more than 4 tokens. For the sequences with more than 64 tokens, we kept only the last 64 tokens. After the preprocess for conversation data, about 1.3 million pairs were kept, which is still sufficient for training our model.

	Training set	Dev. set	Test set
# dialogue turns	1,354,483	4,542	13,440

Table 6.2: Data statistics for conversation corpus after proprocess

We constructed our vocabulary with words, instead of using the popular WordPiece methods like Byte Pair Encoding (BPE), because our model aimed at making incorporation at word-level. After our fine preprocess, a vocabulary with 20000 words was constructed and is suitable for training.

6.1.2 Facts Data

When it comes to the facts data, since they are crawled through the name entities in conversation corpus (name entity recognition was completed by the organizers), in our facts corpus, each fact is assigned a conversation ID as well, which means the "world facts" of a conversation pair [6] is the group of facts that have same conversation ID with the conversation pair. The crawled content from the website was split into a set of sentences. Most fact sentences are declarative sentences like information from Wikipedia and a few are sentences that express one's opinion like comment from Foursquare [7]. With the official

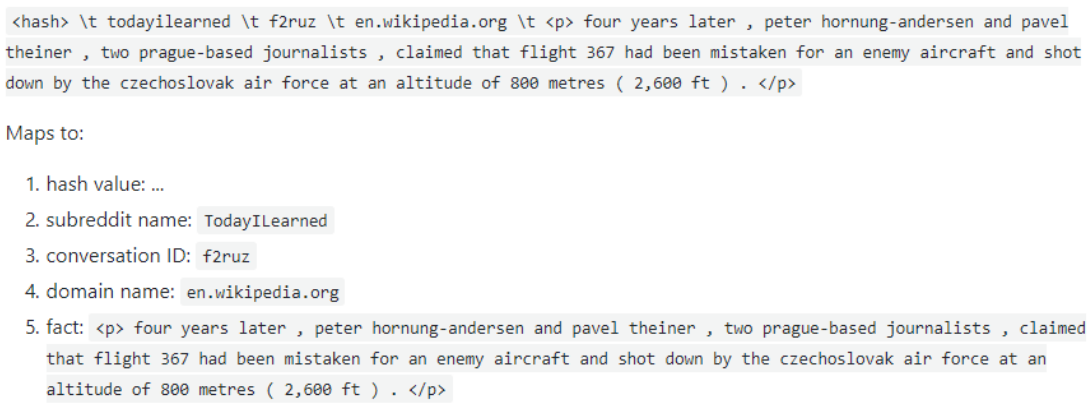

Figure 6.2: Sample fact

scripts, almost all the text of the original pages was kept in the rudimentary facts data, including tags like <title>, <p>, <h1-6>. This first step of our preprocess is to remove

[6] See section 5.1
[7] https://foursquare.com/

the tags making the sentences more like natural declarative sentences. Then the rules for preprocessing the facts data are same with those for conversation data. We removed the facts with no more than 4 tokens.

After preprocessing the facts, we performed fact retrieval with tf-idf [8], so that for each conversation, 10 facts were selected as contextually relevant facts from up to 10000 world facts and kept. Note that if a conversation has less than 10 "world facts", we compensated by repeating the most relevant fact so that 10 relevant facts were obtained. Besides, 20 keywords (each fact 2 keywords) from the contextually relevant facts are collected, which will be used as well for the next step.

6.2 Experimental Setup

After data preprocess and vocabulary construction, we were able to build the end-to-end architecture. In this section, the implementation of training and testing will be discussed, including how we build the concrete framework, the setup of hyper-parameters and the baseline we used.

6.2.1 Framework

The implementation is based on PyTorch [9] and OpenNMT [10].

PyTorch is an open source deep learning framework. It is easy to deploy and flexible in use. A lot of libraries are built based on PyTorch and a large and active community firmly supports the users. For our sequence-relevant task, its advantage of constructing computation process dynamically brings much convenience, which is good at dealing with variable length sequence.

OpenNMT is an open source neural machine translation system, which is built on end-to-end fashion. It provided a stable sequence-to-sequence backbone, including the Transformer and LSTM. We conducted our proposed methods by modifying the basic algorithm and process flow of OpenNMT framework.

6.2.2 Implementation Details

The complete architecture is shown in subsection 6.2.2. Note that Figure 5.1 is only a concept architecture, in experiment, the encoded fact is injected to source sequence word embedding in the first Transformer encoder block and does not go directly to decoder.

We used a PyTorch version BERT open-sourced by HuggingFace to generate our encoded fact representation [11]. BERT uses WordPiece algorithm, split rare words into subword characters for reducing vocabulary size. For example, “jacksonville” is represented as [”*jack*”, ”*##son*”, ”*##ville*”]. If a keyword in our Scheme 2 [12] was split by WordPiece,

[8] See section 5.2
[9] https://pytorch.org/
[10] http://opennmt.net/OpenNMT-py/
[11] https://github.com/huggingface/transformers
[12] See subsection 5.4.1

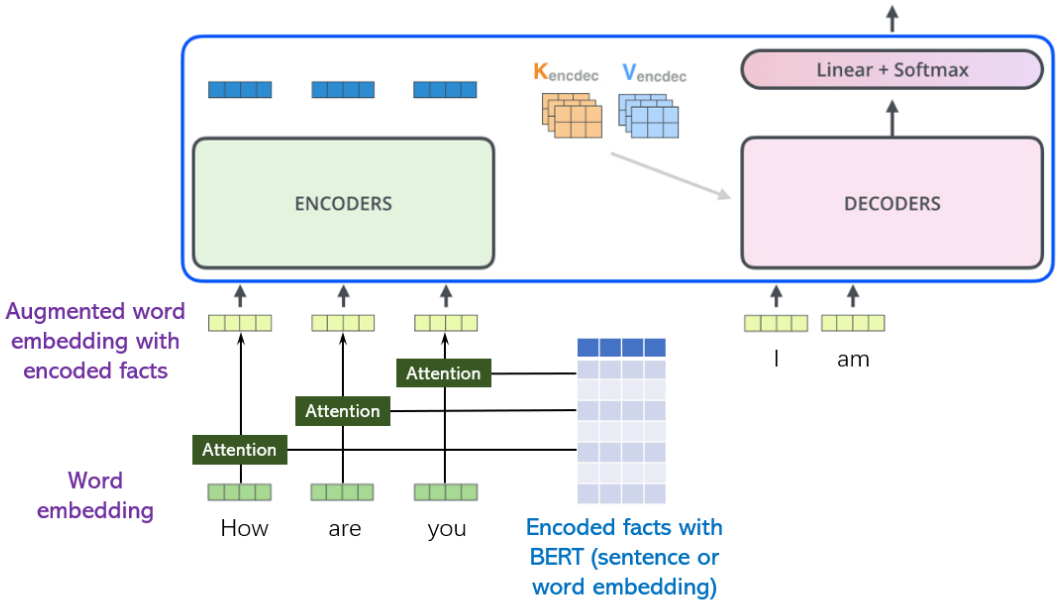

Figure 6.3: Complete architecture

word embedding representing the whole word was calculated as the average of all the pieces. Note that we performed BERT calculation and extracted the required part (vector corresponding to CLS token or vectors corresponding to keywords) as encoded facts. For each fact, we sample two keywords from it. If the keyword appears multiple times in the fact, the average of word embeddings of the same words is considered as the word vector of this word. Note that we store it on disk as text that contains a series of floating point numbers in advance, and load the text while training and testing. In the ideal situation, the system is supposed to generate the encoded fact representation during training or testing, because in practical application like business scenario, the system has no access to the text of conversation history in advance and hence have no way to recognize the required facts. However, in our experimental environment, it is quite time-consuming for calculating BERT, which make the training 2-3 times slower, hence we had to use this trick. We could make improvement on this if we are able to use multi-GPU. For example, at the run-time of current batch, the system generate the encoded facts with BERT for next batch. It is one of the expectations in future work.

We obtained the word embedding with trainable parameters for both encoder and decoder. At first, we attempted to use pre-trained parameters like Glove [26] for reducing the computation workload, but it could hardly train. The reason might be that BERT and Glove are not coupled well in this situation and the too few trainable parameters. Therefore, we introduced trainable parameters for the word embedding, the amount is $vocabulary\ size(20000) \times length\ of\ word\ vector$. It did increase the computation workload, but with that our model was able to train without crash.

6.2.3 Hyperparameters

Table 6.3 shows the hyperparameter we set for generating encoded fact representation with BERT [13].

parameter	description	value
bert_model	Use which BERT model	bert-base-uncase
do_lower_case	Coupled with uncased BERT	true
layers	Capture the output from which layers	-1
max_seq_length	Maximum number of tokens of the input	128

Table 6.3: Hyperparameters for BERT

Table 6.4[14] shows the hyperparameter we set for training our proposed model.

parameter	description	value
word_vec_size	Word embedding size for source and target	768
share_embeddings	Share the word embeddings between encoder and decoder	True
position_encoding	Use a sin to mark relative words positions	True
encoder_type	Type of encoder layer to use	transformer
decoder_type	Type of decoder layer to use	transformer
layers	Number of layers in encoder/decoder	6
rnn_size	Word embedding size in Transformer encoder and decoder (the name is still "rnn")	768
global_attention	The attention type to use	dotprod
self_attn_type	Self attention type in Transformer	scaled-dot
heads	Number of heads for transformer self-attention	8
transformer_ff	Size of hidden transformer feed-forward	512
param_init_glorot	Init parameters with xavier_uniform	True
batch_size	Maximum batch size for training	48
batch_type	Batch grouping for batch_size	sents
normalization	Normalization method of the gradient	sents
accum_count	Accumulate gradient this many times	1
max_generator_batches	Maximum batches of words in a sequence to run the generator on in parallel	2
train_steps	Number of training steps	1000000

[13] The parameters use default value are not mentioned in the table.
Source: https://github.com/huggingface/transformers

[14] The parameters use default value of OpenNMT are not mentioned in the table.
Source: http://opennmt.net/OpenNMT-py/options/train.html

optim	Optimization method	adam
dropout	Dropout probability	0.2
label_smoothing	Label smoothing value epsilon. Probabilities of all non-true labels will be smoothed by epsilon	0.1
learning_rate	Starting learning rate	0.001
decay_method	Use a custom decay rate	noam
warmup_steps	Number of warmup steps for custom decay	4000

Table 6.4: Hyperparameters for our proposed method

Note that in our experiment, a bigger batch size did not mean a shorter computation time, instead a proper and smaller one brings better training efficiency. The reason could be that the GPU we used has a large memory size but a small Flops. Theoretically, expanding batch size k times will result in the computation workload expanding $\sqrt{k}$ times. Our GPU (one side of Nvidia K80) has a relatively big 11GB memory size and is limited by computation performance, no matter how big the batch is, it can only perform the same amount of computation. Since the computation workload increases, the required time increases as well. Only with the GPU limited by memory size can we reduce the run time by increasing batch size, for sure until it reaches the limit of memory size.

6.2.4 Baseline

6.2.4.1 Official Baseline

The baseline models provided by DSTC7-Track2 organizers include constant, random, seq2seq and human response:

- constant: The response is always "i don't know what you mean .", no matter what the input sequence is.
- random: The system randomly select a target sequence in training set as the response.
- seq2seq: a GRU-based seq2seq generation model, under the framework of Keras [15]. It is a pretty rudimentary model since it is a baseline for a competition. It does not involve external knowledge. In terms of sequence-to-sequence backbone, it does not use attention and beam search, instead greedy search. It uses teacher forcing strategy for decoding, the input of decoder at current time step is not the output at last time step but the token from reference sequence. The hyperparameter is shown in Table 6.5[16].

[15] http://keras.io/

[16] https://github.com/mgalley/DSTC7-End-to-End-Conversation-Modeling/tree/master/baseline

- human: The system gives responses from human, which are picked up from the Reddit posts. The responses that have highest score are selected as the response for a query.

parameter	description	value
token_embed_dim	length of word embedding vector	100
rnn_units	number of hidden units of each GRU cell	512
encoder_depth	number of GRU cells stacked in the encoder	2
decoder_depth	number of GRU cells stacked in the decoder	2
dropout_rate	dropout probability	0.5
max_num_token	if not None, only use top max_num_token most frequent tokens	20000
max_seq_len	tokens after the first max_seq_len tokens will be discarded	32

Table 6.5: Hyperparameters for official seq2seq baseline [17]

6.2.4.2 LSTM Baseline and Transformer Baseline

Since our focus point is to explore the influence by incorporation of fact information in word-level on top of Transformer based model, we put forward two baseline as well. First is an RNN (LSTM) model without facts incorporation, second is a Transformer model without facts incorporation. Both are build based on PyTorch and OpenNMT and use the same preprocessed conversation data and the same vocabulary with our proposed facts-involved model. Besides, we used as many the same hyperparameters for the proposd methods as possible so that to the utmost extent, provided identical condition for comparison. The comparison reflects not only the influence by replacing RNN with Transformer in response generation task, but also the influence by injecting facts information into basic Transformer conversation model. It is significant because no other DSTC7-Track2 relevant work uses Transformer as the sequence-to-sequence backbone.

6.3 Evaluation and Discussion

We used the Evaluation Metrics provided by the DSTC7-Track2 organizers and calculated the scores with the official scripts [18]. For measuring appropriateness, BLEU [25], NIST [8] and METEOR [4] are used, which is mentioned in section 2.5.

For measuring informativeness, Entropy and Diversity are introduced. Diversity is calculated as Equation 6.1:

$$n\_Diversity = \frac{\text{number of types of occurring } n\text{-grams}}{\text{number of } n\text{-grams}} \quad n = 1, 2 \tag{6.1}$$

[17] Source: https://github.com/mgalley/DSTC7-End-to-End-Conversation-Modeling/tree/master/baseline
[18] https://github.com/mgalley/DSTC7-End-to-End-Conversation-Modeling/tree/master/evaluation/src

Entropy is calculated as Equation 6.2:

$$
\begin{aligned}
&n\_Entropy = -\sum_{n-gram} \frac{m}{N} \log \frac{m}{N} \\
&n = 1, 2, 3, 4 \\
&m \text{ is the number of occurrences of an n-gram} \\
&N \text{ is the total number of n-gram}
\end{aligned} \tag{6.2}
$$

Naturally, higher Entropy and Diversity can be considered as high response diversity and containing more vocabulary, so on they can reflect high informativeness of responses.

6.3.1 Results

The evaluation was performed with the official script. 2208 samples from the test set were used for calculating the scores. The result was shown in Table 6.6 and Table 6.7, in which we list the scores from two groups that won the prize as well (provided by the organizers), in order to more comprehensively learn from the results.

Models	nist1	nist2	nist3	nist4	bleu1	bleu2	bleu3	bleu4	Meteor
constant	0.175	0.183	0.184	0.184	39.7	12.8	6.06	2.87	7.48
random	1.573	1.633	1.637	1.637	26.4	6.7	2.24	0.86	5.91
seq2seq	0.849	0.910	0.915	0.916	45.2	14.8	5.23	1.82	6.96
human	2.424	2.624	2.647	2.650	34.1	12.4	5.72	3.13	8.31
LSTM	0.668	0.712	0.721	0.722	41.7	14.4	5.11	1.88	6.96
Transformer	0.610	0.656	0.659	0.661	46.3	15.5	5.59	2.13	7.00
scheme1	1.002	1.059	1.065	1.067	40.9	13.7	4.95	1.98	7.04
scheme2	1.114	1.177	1.184	1.185	34.8	11.2	4.65	1.95	7.07
team X	1.925	2.039	2.047	2.047	37.1	11.3	3.66	1.35	6.71
team Y	1.419	1.509	1.515	1.515	36.8	10.9	3.70	1.32	6.43

Table 6.6: Appropriateness metrics. Official baselines include constant, random, seq2seq and human. LSTM and Transformer are our own encoder-decoder backbone baseline. Scheme1 and scheme2 are our proposed model incorporating external knowledge via sentence embedding and word embedding respectively, see section 5.4. Team X [19] and team Y [20] are results by 2 other competing groups.

[19] Team X: Tanaka R, Ozeki A, Kato S, et al. An Ensemble Dialogue System for Facts-Based Sentence Generation[J]. arXiv preprint arXiv:1902.01529, 2019.

[20] Team Y: Zheng J, Kasturi S, Mason Lin X C, et al. The OneConn-MemNN System for Knowledge-Grounded Conversation Modeling[J]. 2019.

Models	entropy1	entropy2	entropy3	entropy4	div1	div2	avg_len
constant	2.079	1.946	1.792	1.609	0.000	0.000	8.000
random	6.493	9.670	10.403	10.467	0.160	0.647	19.192
seq2seq	3.783	5.017	5.595	5.962	0.014	0.048	10.604
human	6.589	9.742	10.410	10.445	0.167	0.670	
LSTM	3.813	5.237	5.643	6.085	0.022	0.073	9.729
Transformer	3.676	5.119	5.666	6.072	0.023	0.077	10.624
scheme1	3.931	5.399	5.887	6.183	0.026	0.079	10.928
scheme2	4.152	5.401	5.846	6.143	0.036	0.113	11.807
team X	5.395	7.925	9.084	9.596	0.094	0.334	14.444
team Y	4.406	6.238	7.117	7.639	0.053	0.171	12.674

Table 6.7: Informativeness metrics

6.3.2 Discussion

In terms of appropriateness evaluations, we can find that human responses got obviously higher scores than other groups in NIST, compared to BLEU and METEOR. This trend is same as that in [35]. We can consider NIST as the most reliable metric for measuring appropriateness of the responses generated by a machine system, among NIST, BLEU and METEOR. According to the results, both our proposed models got higher scores in NIST, compared to the machine baselines. This can reflect that, our proposed methods for introducing external knowledge have benefit on improving the quality and appropriateness of the generated responses. However, the winning teams had a higher score than ours. They didn't only focus on how to introduce facts, but make progress on other parts like data preprocessing, facts retrieval and response reranking. This effort are also significant for improving quality of responses. In BLEU test which reflects the precision of n-gram matching, we can find that the Transformer baseline had highest scores, followed by the official seq2seq baseline. Other groups, including human responses, our proposed methods and the winning teams, didn't have outstanding scores. On the one hand, we can consider that the Transformer made progress on the performance of encoder-decoder architectures since the n-gram precision increased. On the other hand, we can infer that BLEU is not so reliable for evaluating response generation task since human responses and the sophisticated models have lower scores than the simple models. In terms of METEOR, human responses had highest score, followed by constant response. The scores of other machine responses fall on a small range. We can infer that METEOR can reflect the difference between human responses and responses from machine systems, but not so good at judging the performance of different systems.

In terms of informativeness evaluations, we can find that human responses got highest scores in entropy and diversity, followed by random responses, which are sampled from human responses on the Reddit posts. Human responses contain naturally most various word and have highest average lengths. Compared to the baseline encoder-decoder models (official seq2seq, our own LSTM and Transformer baselines), our proposed methods have higher scores in entropy and diversity, especially in diversity. This reflects that our proposed methods that introduce contextually relevant facts made progress on making

neural conversation model generate more vivid and contentful responses. This trend can be found in "average length" term as well. The scores of our proposed models are higher than baselines, but lower than human responses and the sophisticated models by other winning teams. The efforts on other steps like response reranking did have great contribution on improving response diversity.

conversation history	lebron , kyrie have the best-selling signature sneakers among nba players
facts	lebron , kyrie have the best-selling signature sneakers among nba players
	lebron james and kyrie irving are chasing their third straight trip to the nba finals together . (photo by gregory shamus / getty images)
	cleveland ' s biggest stars , james and kyrie irving , have already captured one title this year over golden state ' s own dynamic duo of stephen curry and kevin durant . nike sold more of james ' and irving ' s signature sneakers in the u . s . than those of any other active nba player during the 12 months ending in march 2017 , as measured by dollar volume , according to matt powell , an analyst at market research firm npd group .
	james has been one of the nba ' s elite shoe salesman since he entered the nba in 2003 armed with a seven-year , $ 90 million contract from nike . early editions of lebron ' s signature shoes were not that well received , but sales took off with the lebron vi . nike was selling more than $ 300 million worth of lebron sneakers annually by 2013 . the company locked up james to a " lifetime " deal at the end of 2015 worth as much as $ 1 billion , according to james ' business manager maverick carter . nike released the 14 th version of lebron ' s signature shoe this season .
	the current crop of nba stars compete on the court , but they can't touch a basketball legend who has not laced up his hightops for an nba game when it comes to shoe sales .
	the nba is a hot property globally right now with franchise values soaring , but the performance basketball shoe market is struggling and even king james is not immune . retailers and brands have reported softness in the signature basketball market and powell says he sees the same thing in his numbers with shoe sales for james , curry and durant all down over the past 12 months .

	irving is the only player among the four best sellers without an mvp trophy on his mantle (the other three have combined for seven mvps) , but his shoe sales are on fire . nike brand president trevor edwards said during a nike earnings call in march that the kyrie 3 launched in december was the best-selling performance basketball shoe in the market . credit the more affordable price of $ 120 and the newness of the irving shoe . nike has also backed irving ' s shoes with its marketing muscle .
	the current crop of nba stars compete on the court , but they can't touch a basketball legend who has not laced up his hightops for an nba game when it comes to shoe sales . michael jordan is still the king of nba sneakers with the nike subsidiary jordan brand commanding more than 50 % of the basketball shoe market . the jordan performance has been soft , but the retro business is " quite strong " says powell . nike reported revenue for jordan of $ 2.8 billion last year , or more than 10 times the sales of any active nba hoopster .
	curry ' s under armour kicks ranked third among the best-sellers with durant (nike) in fourth . houston rockets point guard james harden and his first signature shoe with adidas rounded out the top five , although there was a big sales gap between durant and harden says powell .
	michael jordan is still the king of nba sneakers with the nike subsidiary jordan brand commanding more than 50 % of the basketball shoe market . the jordan performance has been soft , but the retro business is " quite strong " says powell . nike reported revenue for jordan of $ 2.8 billion last year , or more than 10 times the sales of any active nba hoopster .
keywords	lebron, kyrie; lebron,irving; cleveland,james; james, lebron; nba, legend; powell, durant; irving, kyrie; jordan, nba; powell, curry; jordan, powell
LSTM Transformer scheme1 scheme2	i do n't think he 's going to be the best player . he was the best player in the league . to be fair , lebron james 's also the best player in the world . and that 's why irving 's the best player in the nba .

Table 6.8: Response example 1 for our proposed methods and baselines

In Table 6.8, a Response example from our proposed methods and baselines is shown. We can find that both our LSTM baseline and Transformer baseline are able to generate appropriate and fluent response. They all learnt from the conversation corpus and generated responses with a backbone "somebody is the best player". In comparison, both of our proposed make progress on making the response more informative and contentful

without losing the appropriateness and fluency. Besides the backbone "somebody is the best player", by injecting the contextually relevant facts, the systems take into account the relationship between the NBA players' names and other named entities more than the normal encoder-decoder models. In fact retrieval step, tf-idf helped us select the facts with more special nouns or named entities, which were highly likely to be sampled as keywords as well. With the contribution of the pre-trained parameters from BERT, the systems represented the filtered fact information properly and successfully made association with the words, especially nouns in conversation history, so that it is more likely to make the responses contain more special nouns or named entities. [21]

conversation history	A hiroshima policeman went to nagasaki to teach other police officers to duck and cover in the days between the bombings . not a single officer died in the nagasaki blast .
LSTM	i do n't know if this is true , but i do n't think it 's true .
Transformer	not sure if you 're joking or not , but i think you 're right .
scheme1	i wonder how many people died in hiroshima and nagasaki .
scheme2	hiroshima and nagasaki were more likely to be released in japan .

Table 6.9: Response example 2

conversation history	Arsenal becomes the most winning team in the fa cup with 13 wins .
LSTM	i 'm not sure what you 're talking about .
Transformer	i do n't know what you 're talking about .
scheme1	do n't worry , arsenal can n't afford it .
scheme2	this guy is a arsenal fan .

Table 6.10: Response example 3

In Table 6.9 and Table 6.10, two examples are shown, where both LSTM and Transformer baselines without external knowledge gave a fluent, but meaningless response, while both our proposed models generated relatively contentful and informative responses associating with the contextually relevant facts and containing more specific entities like "arsenal" and "hiroshima" without losing the appropriateness and fluency.

[21] Note that we do not care if the generated response is consistent in meaning with objective fact.

7 Conclusions and Future Work

7.1 Review

In this thesis, we targeted at making improvement on neural network-based conversational response generation in single turn so that the generated response can be more informative and contentful. The proposed model is supposed to be trained in a fully data-driven fashion with minimal hand-coding. This goal is same as that of grounded response generation task at Dialog System Technology Challenge 7 (DSTC7-Track2). Thus, we conducted our experiment on the condition of the competition. The problem can be abstracted as, given an input sequence and world facts, the proposed system is supposed to be able to retrieve the contextually relevant facts from the world facts, then generate a response with respect to both the contextually relevant facts and the input sequence.

The intuitive idea for this issue is to make some modification on a traditional encoder-decoder framework. After obtaining contextually relevant facts with tf-idf, we could encode them and feed them into dialogue encoder as well, making dialogue encoder take into account both conversation history and the facts. Since Transformer was developed and has been proven to be effective in many NLP tasks, enjoying the advantages of being good at dealing with long sequence and requiring less computation so that more layers could be involved, we paid more attention to Transformer instead of traditional RNN models. We built an LSTM encoder-decoder baseline to explore the improvement resulted by using Transformer and meanwhile built a Transformer baseline to explore the influence of the incorporation of fact information.

We implemented our experiment with deep learning framework PyTorch and made modification on the architecture of encoder-decoder model from OpenNMT. At first the conversation data was too dirty to be used for training. After a series of data preprocess, including tokenization, rule-based regularization and some transformation, we built a vocabulary with 20000 words in word-level and managed to train reliable baseline LSTM and Transformer models with the cleaned data.

Then we put forward a method for injecting fact information into dialogue encoder. Since we used Transformer, there is no more a fixed length vector containing the information of the whole input sequence, instead, a list of word embeddings, whose length is same as the input sequence. We first transformed the facts into semantic representation with BERT in 2 schemes: 1. use the vector corresponding to CLS token to represent the sentence; 2. use the vectors corresponding to two key words in the sentence to represent it. The key words were found at the fact retrieval step as well. The size of vectors from BERT and word embedding in dialogue encoder was set to be same, so that we could conveniently perform attention calculation. With attention we got the relevance between each fact and the word, and then weighed the fact according to the relevance by softmax.

After the summation of original word embedding and the weighted facts, an augmented word embedding was obtained, which contains information from both the word itself and the facts. It was followed by normal encoder and decoder functions and a set of candidate responses was then generated. In order to improve response diversity so that improve the quality of final responses, we selected the responses containing most various words as the final response. The selection was applied to both our proposed model and the baselines.

Finally the automatic evaluation about both appropriateness and informativeness was conducted with the official script. According to the result, our proposed method not only remains well-performed on generating appropriate and fluent response, but also makes progress on making the responses more informative and contentful. In appropriateness metrics, NIST scores are about 0.3 points higher than baselines, METEOR are 0.04 higher, only BLEU are lower than baselines. In informativeness metrics, entropy are about 0.1 higher and diversity are 0.1 higher than baseline. However, our scores are still obviously lower than the winning teams and human.

7.2 Future Work

Reviewing on the works that we have done, there is still a long way to go to perfectly realize our target. In this thesis, we mainly focused on the mechanism of obtaining representation of facts and the method of injecting it into dialogue encoder. In some joint parts of our whole architecture[1], some methods are quite rudimentary and rule-based. For example, we could find a better way to find contextually relevant facts than tf-idf, a statistical or neural-based method to determine the final response instead of the response containing most various words. In terms of obtaining semantic representation of facts, we can introduce a more sophisticated method instead of simply using embeddings corresponding to CLS token or words (average of word pieces). For example, combining BERT with Power Mean[28] and Smooth Inverse Frequency[1], or introduce a small amount of trainable parameters.

In term of the experiment, there is also some detail worth improving. For example, we store the vectors from BERT on disk as text that contains a series of floating point numbers in advance, and load the text while training and testing to avoid consuming too much time for calculating BERT. However, the system is supposed to generate the encoded fact representation meanwhile training or testing, because in practical application the system has no access to the conversation history in advance and hence have no way to recognize the required facts. We could make improvement on this if we are able to use multi-GPU, like that at the run-time of current batch, the system generates the encoded facts with BERT for next batch.

[1] See Figure 5.1

Bibliography

[1] Sanjeev Arora, Yingyu Liang, and Tengyu Ma. "A simple but tough-to-beat baseline for sentence embeddings". In: (2016).

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. *Neural Machine Translation by Jointly Learning to Align and Translate*. 2014. arXiv: 1409.0473 [cs.CL].

[3] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate". English (US). In: *arXiv* (2014).

[4] Satanjeev Banerjee and Alon Lavie. "METEOR: An automatic metric for MT evaluation with improved correlation with human judgments". In: *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*. 2005, pp. 65–72.

[5] Chris Callison-Burch, Miles Osborne, and Philipp Koehn. "Re-evaluating the Role of Bleu in Machine Translation Research". In: *11th Conference of the European Chapter of the Association for Computational Linguistics*. Trento, Italy: Association for Computational Linguistics, Apr. 2006. URL: https://www.aclweb.org/anthology/E06-1032.

[6] Kyunghyun Cho et al. "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation". In: *CoRR* abs/1406.1078 (2014). arXiv: 1406.1078. URL: http://arxiv.org/abs/1406.1078.

[7] Jacob Devlin et al. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. 2018. arXiv: 1810.04805 [cs.CL].

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

[9] Michel Galley et al. "End-to-End Conversation Modeling : Moving beyond Chitchat DSTC 7 Task 2 Description (v 1 . 0)". In: 2018.

[10] Michel Galley et al. "Grounded Response Generation Task at DSTC7". In: *AAAI Dialog System Technology Challenges Workshop*. 2019.

[11] Felix A Gers and Jürgen Schmidhuber. "Long short-term memory learns context free and context sensitive languages". In: *Artificial Neural Nets and Genetic Algorithms*. Springer. 2001, pp. 134–137.

[12] Marjan Ghazvininejad et al. *A Knowledge-Grounded Neural Conversation Model*. 2017. arXiv: 1702.01932 [cs.CL].

[13] Marjan Ghazvininejad et al. "A knowledge-grounded neural conversation model". In: *Thirty-Second AAAI Conference on Artificial Intelligence*. 2018.

[14] Alex Graves et al. "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks". In: *Proceedings of the 23rd international conference on Machine learning*. ACM. 2006, pp. 369–376.

[15] SL Hadla, MT Hailat, and NM Al-Kabi. "Comparative Study Between METEOR and BLEU Methods of MT: Arabic into English Translation as aCase Study". In: *International Journal of Advanced Computer Science and Applications* (2015).

[16] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory". In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[17] Nal Kalchbrenner and Phil Blunsom. "Recurrent Continuous Translation Models". In: *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Seattle, Washington, USA: Association for Computational Linguistics, Oct. 2013, pp. 1700–1709. URL: https://www.aclweb.org/anthology/D13-1176.

[18] Philipp Koehn, Franz J. Och, and Daniel Marcu. "Statistical Phrase-Based Translation". In: *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*. 2003, pp. 127–133. URL: https://www.aclweb.org/anthology/N03-1017.

[19] Jiwei Li et al. *A Diversity-Promoting Objective Function for Neural Conversation Models*. 2015. arXiv: 1510.03055 [cs.CL].

[20] Ryan Lowe et al. *The Ubuntu Dialogue Corpus: A Large Dataset for Research in Unstructured Multi-Turn Dialogue Systems*. 2015. arXiv: 1506.08909 [cs.CL].

[21] Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. *Effective Approaches to Attention-based Neural Machine Translation*. 2015. arXiv: 1508.04025 [cs.CL].

[22] Hongyuan Mei, Mohit Bansal, and Matthew R. Walter. *Coherent Dialogue with Attention-based Language Models*. 2016. arXiv: 1611.06997 [cs.CL].

[23] Tomáš Mikolov et al. "Recurrent neural network based language model". In: *Eleventh annual conference of the international speech communication association*. 2010.

[24] Jekaterina Novikova et al. "Why We Need New Evaluation Metrics for NLG". In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Copenhagen, Denmark: Association for Computational Linguistics, Sept. 2017, pp. 2241–2252. DOI: 10.18653/v1/D17-1238. URL: https://www.aclweb.org/anthology/D17-1238.

[25] Kishore Papineni et al. "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.

[26] Jeffrey Pennington, Richard Socher, and Christopher Manning. "Glove: Global vectors for word representation". In: *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 2014, pp. 1532–1543.

[27] Yu-Ping Ruan et al. “Promoting Diversity for End-to-End Conversation Response Generation”. In: *arXiv preprint arXiv:1901.09444* (2019).

[28] Andreas Rücklé et al. “Concatenated power mean word embeddings as universal cross-lingual sentence representations”. In: *arXiv preprint arXiv:1803.01400* (2018).

[29] Abigail See, Peter J Liu, and Christopher D Manning. “Get to the point: Summarization with pointer-generator networks”. In: *arXiv preprint arXiv:1704.04368* (2017).

[30] Iulian V Serban et al. “Building end-to-end dialogue systems using generative hierarchical neural network models”. In: *Thirtieth AAAI Conference on Artificial Intelligence*. 2016.

[31] Louis Shao et al. *Generating High-Quality and Informative Conversation Responses with Sequence-to-Sequence Models*. 2017. arXiv: 1701.03185 [cs.CL].

[32] Elior Sulem, Omri Abend, and Ari Rappoport. “BLEU is Not Suitable for the Evaluation of Text Simplification”. In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium: Association for Computational Linguistics, Oct. 2018, pp. 738–744. DOI: 10.18653/v1/D18-1081. URL: https://www.aclweb.org/anthology/D18-1081.

[33] Yik-Cheung Tam et al. “Cluster-based beam search for pointer-generator chatbot grounded by knowledge”. In: *Proceedings of the Thirty-Three AAAI Conference on Artificial Intelligence (AAAI 2019)*. 2019.

[34] Ryota Tanaka et al. “An Ensemble Dialogue System for Facts-Based Sentence Generation”. In: *arXiv preprint arXiv:1902.01529* (2019).

[35] Joseph P Turian, Luke Shea, and I Dan Melamed. *Evaluation of machine translation and its evaluation*. Tech. rep. NEW YORK UNIV NY, 2006.

[36] Ashish Vaswani et al. *Attention Is All You Need*. 2017. arXiv: 1706.03762 [cs.CL].

[37] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. *Pointer Networks*. 2015. arXiv: 1506.03134 [stat.ML].

[38] Bolin Wei et al. “Why do neural dialog systems generate short and meaningless replies? a comparison between dialog and translation”. In: *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2019, pp. 7290–7294.

[39] Kaisheng Yao, Geoffrey Zweig, and Baolin Peng. *Attention with Intention for a Neural Network Conversation Model*. 2015. arXiv: 1510.08565 [cs.NE].

[40] Li Yao et al. “Describing videos by exploiting temporal structure”. In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 4507–4515.

[41] Jun Yin et al. “Neural generative question answering”. In: *arXiv preprint arXiv:1512.01337* (2015).

[42] Junyuan Zheng et al. “The OneConn-MemNN System for Knowledge-Grounded Conversation Modeling”. In: (2019).