An Equal-Size Hard EM Algorithm for Multi-Decoder Dialogue Generation

by

Yuqiao Wen

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computing Science
University of Alberta

Abstract

Building intelligent open-domain dialogue systems is a long-standing goal of artificial intelligence. These systems, also known as chatbots, aim to hold conversations with humans in an open-ended fashion. However, it is well known that standard encoder–decoder dialogue systems tend to generate generic responses. A previous study hypothesizes that this phenomenon is due to the one-to-many mapping in the open-domain dialogue task, where the target distribution is multi-modal. As a result, standard cross-entropy training fails as it learns an overly smoothed function that causes the mode averaging problem.

In this work, we address the mode averaging issue with a multi-decoder model, where each decoder can cover a subset of the modes. We treat the choice of the decoder as a latent variable and apply EM-like algorithms. However, we observe that traditional Hard-EM and Soft-EM may not perform well due to the collapse issue: the decoders fail to specialize and the multi-decoder model degenerates to a single-decoder model. To this end, we propose EqHard-EM, which is an EM variant that assigns an equal number of samples to every decoder to alleviate the collapse issue. Results show that our EqHard-EM algorithm achieves significant improvements over single-decoder models in terms of both response quality and diversity. In addition, extensive analyses show that our EqHard-EM algorithm indeed alleviates the collapse issue: different decoders are specialized and generate diverse responses.

Acknowledgments

First and foremost, I would like to thank my supervisor, Dr. Lili Mou. Applying the EM algorithm in a multi-decoder setup has always been his original idea, and we started exploring it soon after I became his student. Together, we faced and tackled lots of problems along the way, leading to a key observation: assigning samples in an equal manner, even without applying the EM algorithm, achieves a good baseline performance. This motivated our Equal-Size Hard Expectation–Maximization algorithm, which is the heart of this thesis.

As a supervisor, Lili is unique in that he spends a lot of time with his students, which I appreciate immensely. Over the past year, we spent countless hours together discussing experiments, math, and to my surprise, scientific writing. Lili taught me how to write papers essentially from the ground up, and he even helped me edit this thesis to a large extent. It is no exaggeration that this thesis would not be complete without his help. Overall, Lili is a brilliant mentor who is passionate about research and also cares about his students.

I would like to additionally thank my co-author, Guoqing Luo, for the paper "An Empirical Study on the Overlapping Problem of Open-Domain Dialogue Datasets", which was published at LREC this year. We found out that many open-domain dialogue datasets have the overlapping problem when we were exploring our EM algorithms, so we decided to submit a separate paper and set up a proper benchmark to continue our research. Guoqing is a first-year Master's student who just joined our group, and he helped me to run experiments mainly on the OpenSubtitles dataset, whereas I developed the code and focused on the DailyDialog dataset. It was a unique experience for me as it was my first collaboration with another student, and I am very thankful for his hard work.

Last but not least, I would like to thank my parents for their continuous support. As an immigrant, I started my journey in Canada when I was 14, at which point I barely spoke any English. Needless to say, my parents also struggled a lot to find their way in Canada, and they pass their hard-earned experience to me. They are always there when I need them, and they strongly supported my decision to pursue further education. I am forever indebted to their selfless support.

The research is supported in part by the Natural Sciences and Engineering Research Council of Canada (NSERC) under grant No. RGPIN2020-04465, the Amii Fellow Program, the Canada CIFAR AI Chair Program, a UAHJIC project, a donation from DeepMind, and Compute Canada (www.computecanada.ca).

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Background

Artificial intelligence (AI) has been gaining increasing interest in both industry and academia due to its rapid progress in recent years. This can be attributed to many factors such as more computing power, larger datasets, and deep learning architectures. As a result, AI has reached a tipping point where it can potentially outperform humans in many areas such as computer vision, natural language processing, and decision making.

Natural language processing is one of the most important sub-fields of AI, and it has applications in our day-to-day life. In general, natural language processing can be further divided into two main branches: natural language understanding and natural language generation. The former focuses on the comprehension of human language; its applications include syntactic parsing [10, 18], sentiment analysis [3, 56], and information extraction [58, 55]. The latter focuses on producing text outputs, and has wide applications such as machine translation [4, 35], paraphrase generation [27, 57], and dialogue generation [45, 20].

In this thesis, we focus on the task of open-domain dialogue generation [45, 47, 77]. Dialogue generation can be categorized into two main paradigms: open-domain and task-oriented. Task-oriented dialogue systems focus on achieving specific goals such as booking a hotel [81, 9, 92]. On the other hand, open-domain dialogue systems focus on conversing with humans in an open-ended fashion with no clear goals [73, 36].

Open-domain dialogue systems play an important role in our modern society as they provide a way for lonely people to feel connected. For example, Microsoft's chatbot XiaoIce[1] has gained 660 million active users across five countries since its release in May 2014. The chatbot is also deployed to over 40 platforms such as Facebook, WeChat, and Line. The success and popularity of XiaoIce emphasize the importance of open-domain dialogue systems and call for the need of further improvements.

1.2 Main Challenges

It is well known that traditional Seq2Seq models do not work well with the dialogue task [45, 47, 38]. In particular, Seq2Seq models tend to generate generic responses such as *–I don't know* when trained with the standard maximum likelihood estimation. Wei et al. [79] study this issue and hypothesize that the generic responses are due to the one-to-many mapping phenomenon in the dialogue generation task. Here, one-to-many mapping suggests that, for a given input context, there could be many plausible responses. For example, consider the dialogue context *–How was your weekend?*. A potentially infinite number of responses are valid, depending on what actually happened over the weekend. Therefore, the goal of a dialogue system is to learn a multi-modal distribution, where the modes can be intuitively thought of as different topics of the responses. In the previous example, a mode or a topic could be "visiting a friend" or "playing video games". Unfortunately, standard maximum likelihood estimation fails to capture different modes: they cause the model to learn an overly smoothed function in an attempt to cover all the modes. Therefore, dialogue systems tend to have the mode averaging issue: they fail to select a mode and resort to generating generic responses such as *–I don't know.*

[1]http://www.xiaoice.com

Previous studies address the generic responses during either training time or inference time. Training-time approaches alleviate generic responses by modifying the problematic maximum likelihood estimation. Li et al. [46] use reinforcement learning to maximize a manually crafted scoring function. Khan et al. [38] apply the generative adversarial network in the latent space to address the same issue. These methods address the diversity issue at training time. The lack of diversity can also be addressed at inference time. For example, diverse beam search [76] encourages diverse beam results by penalizing similar sentences. Also, nucleus sampling [33] increases the diversity by allowing less frequent words to be chosen.

1.3 Contribution of this Thesis

We propose to address the diversity issue with a multi-decoder model. In particular, we adopt the standard encoder–decoder framework but enhance it with multiple decoders. Our multi-decoder model alleviates the mode averaging problem by allowing different decoders to cover different modes. We further propose to implement the decoders as adapter modules [34], i.e., a few additional layers, rather than a fully parameterized deep neural network. This improves the memory efficiency and enables knowledge sharing between different decoders.

To train our multi-decoder model, we adopt the expectation–maximization (EM) framework. In our setting, the choice of decoder is considered as an unobserved latent variable, and therefore the EM algorithm is a principled way of training our latent variable model. However, we observe that classic EM algorithms fail to train our multi-decoder model meaningfully. In particular, the Soft-EM algorithm assigns samples to decoders in a soft manner, but it tends to assign an equal fraction of each sample to every decoder. This is because decoders are similar at initialization, leading to similar posterior distributions during the assignment of Soft-EM. As a result, each decoder is trained synchronously, and the full multi-decoder model degenerates to a single-decoder model. We refer to this as the synchronous-training collapse.

We additionally observe that the Hard-EM algorithm also fails in our multi-decoder training. Unlike Soft-EM, Hard-EM assigns samples to the best-fit decoder. This causes the "rich-gets-richer" phenomenon, where samples will favor the more frequently assigned decoders. Empirically, we find that most decoders are untrained when we apply the Hard-EM algorithm. We refer to this as the non-training collapse.

To this end, we propose an Equal-Size Hard Expectation–Maximization (EqHard-EM) algorithm. EqHard-EM avoids the synchronous-training collapse by adopting hard assignment as in Hard-EM. In addition, the assignments are constrained such that every decoder is assigned with the same number of samples. The equal constraint ensures that every decoder is adequately trained, avoiding the non-training collapse. We further formulate the equal-assignment problem as a balanced assignment problem, which can be solved by the classical Hungarian algorithm [39].

To evaluate our approach, we conducted extensive experiments on the Weibo and OpenSubtitles datasets. Our results show that EqHard-EM achieves significant improvements in terms of both response quality and diversity. Moreover, our case study and analysis confirm that our model and algorithm indeed alleviate the mode averaging problem.

To sum up, the main contributions of this paper include:

- We propose to use the multi-decoder architecture to address the one-to-many mapping in dialogue systems.
- We propose the EqHard-EM algorithm to address the deficiency of traditional EM variants.
- We conducted extensive experiments and analyses on Weibo and OpenSubtitles datasets to verify our model and algorithm.

1.4 Thesis Structure

In this chapter, I introduced the background and motivation of this thesis, and stated thesis contributions.

Chapter 2 reviews the previous work related to this thesis. In particular, it starts with an introduction to sequential models for text generation, ranging from a basic recurrent neural network to the widely used Transformer model. The chapter then provides the background for dialogue systems with a focus on open-domain dialogue generation.

Chapter 3 describes the details of our proposed approach, including the neural architecture, model training, and model inference. I will first explain the motivation of our multi-decoder model and its implementation. Then, I will introduce the expectation–maximization (EM) algorithm, which is adopted for our latent variable training. The deficiencies of traditional EM training will be discussed, which motivates our proposed training approach. The chapter ends with the inference procedure.

Chapter 4 presents our experimental results. It starts with an introduction to the datasets and the evaluation metrics. Then, the main results are shown, followed by both quantitative and qualitative analyses.

Finally, Chapter 5 concludes the findings and contributions of this thesis. In addition, it provides an outlook on potential future work to further improve neural dialogue generation.

Chapter 2

Background and Related Work

2.1 Overview

In this chapter, I will first explain the text generation models in Section 2.2. We start with vanilla recurrent neural networks and discuss the vanishing and exploding gradient problem. This naturally leads to the long short-term memory units, which alleviate the vanishing and exploding gradients through carefully designed internal gates. I will then explain the information bottleneck problem in Seq2Seq models, which motivates the attention mechanism. Finally, we will finish with the widely used Transformer model: they replace the recurrent connections with attention and achieve state-of-the-art performance in most NLP tasks.

In Section 2.3, we will move on to dialogue systems. I will first explain the two main paradigms of dialogue research: open-domain and task-oriented. Although they seem to be polar opposites of each other, we will see how a conversational question answering system addresses both settings simultaneously. We will then move on to open-domain dialogue evaluation, which mainly focuses on the quality and diversity of dialogue systems.

2.2 Text Generation Models

Text generation is most commonly accomplished by encoder–decoder models, sometimes also known as sequence-to-sequence (Seq2Seq) models [72]. Such a framework

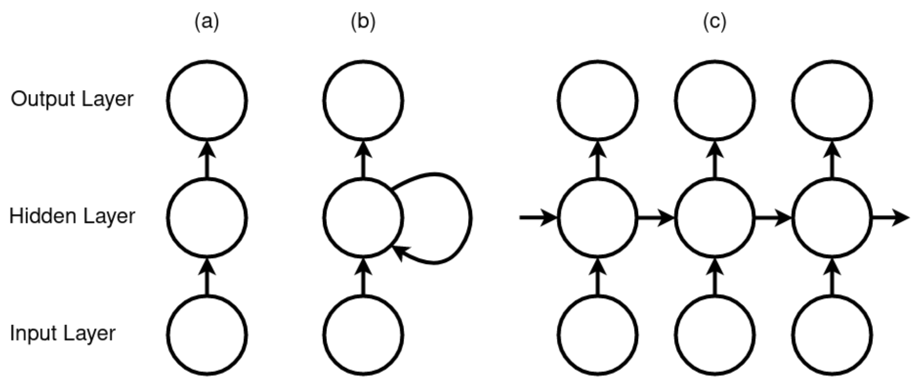

Figure 2.1: (a) Feed-forward neural network, (b) Compact representation of a recurrent neural network, and (c) Unrolled representation of a recurrent neural network.

consists of two main components: an encoder and a decoder. The encoder is responsible for encoding the input sequence to generate hidden representations, whereas the decoder decodes the hidden representation into an output sequence. This framework has been later successfully applied to various text generation tasks, such as paraphrase generation [48, 5], dialogue generation [45, 20], and machine translation [4, 35].

More specifically, the Seq2Seq framework can be realized by different neural architectures. For example, recurrent neural networks (RNNs) are common in early studies [53, 37, 62], and later the attention mechanism is developed to enhance the alignment the input and output in RNNs [4]. The Transformer model, which is fully based on attention mechanisms, becomes a prevailing architecture in the past several years. The following subsections give a brief overview of these models.

2.2.1 Recurrent Neural Network

A recurrent neural network (RNN) is a type of artificial neural network that differs from a standard feed-forward neural network (FNN) [6]. In FNN, information is processed in a feed-forward fashion from an input layer, to one or a few hidden layers, and finally to the output layer (Figure 2.1a). On the contrary, RNN has a loop in the hidden layer, and thus is *recurrent* (Figure 2.1b). During computation,

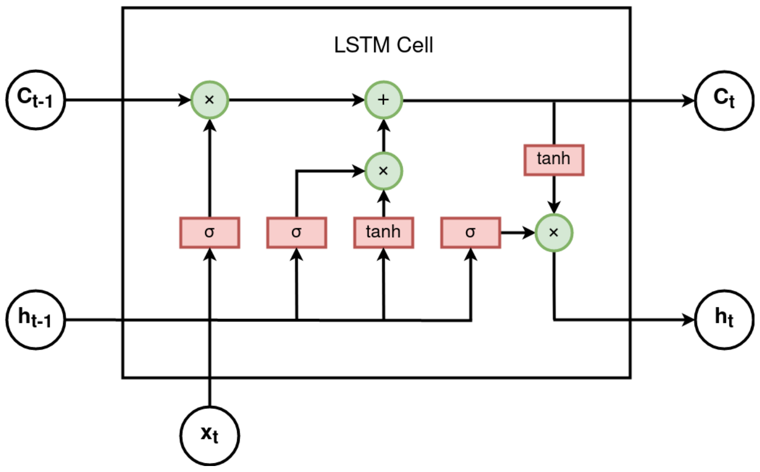

Figure 2.2: The internal design of an LSTM unit.

the loop is expanded along the time step, as shown in Figure 2.1c. In this way, RNNs are suitable for modeling time-series data, such as music generation [8, 54, 14], temperature prediction [23, 51, 86], and text generation [46, 15, 16].

Formally, let $\mathbf{x}_t \in \mathbb{R}^d$ be the input vector at time step t. Then the hidden layer $\mathbf{h}_t \in \mathbb{R}^h$ of a vanilla RNN is computed as

$$\mathbf{h}_t = \phi(\mathbf{W}_{\text{in}}\mathbf{x}_t + \mathbf{W}_{\text{recurrent}}\mathbf{h}_{t-1} + \mathbf{b}) \tag{2.1}$$

where $\mathbf{W}_{\text{in}} \in \mathbb{R}^{h \times d}$ is the input weight matrix, $\mathbf{W}_{\text{recurrent}} \in \mathbb{R}^{h \times h}$ is the recurrent weight matrix, $\mathbf{b} \in \mathbb{R}^h$ is the bias term, $\mathbf{h}_{t-1}$ is the hidden state from the previous time step, and ϕ is a non-linear activation function such as tanh and ReLU [2].

A key problem of the vanilla RNN is the vanishing and exploding gradient [60]. Recurrent neural networks are usually trained with Back-Propagation Through Time [83], which applies the chain rule of derivatives recursively through time steps. Therefore, the gradients keep getting multiplied by the chain rule, and the norm of the gradients can either explode to infinity or shrink to zero as the number of time steps increases.

The Long Short-Term Memory (LSTM) [32] network is proposed to address the vanishing and exploding gradient problem. The LSTM unit contains carefully designed gates that choose which information to keep or forget, allowing the model to build long-term dependencies that cannot be learned by vanilla RNNs. Figure 2.2

shows the internal design of an LSTM unit. In general, the LSTM unit at time step t takes the input vector $\mathbf{x}_t$, the previous hidden state $\mathbf{h}_{t-1}$, and the cell state $\mathbf{c}_{t-1}$ as the input to produce the updated cell state $\mathbf{c}_t$ and hidden state $\mathbf{h}_t$. The detailed computation is as follows

$$\mathbf{i}_t = \sigma(\mathbf{W}_{\text{ii}}\mathbf{x}_t + \mathbf{W}_{\text{hi}}\mathbf{h}_{t-1} + \mathbf{b}_{\text{i}}) \tag{2.2}$$

$$\mathbf{f}_t = \sigma(\mathbf{W}_{\text{if}}\mathbf{x}_t + \mathbf{W}_{\text{hf}}\mathbf{h}_{t-1} + \mathbf{b}_{\text{f}}) \tag{2.3}$$

$$\mathbf{g}_t = \tanh(\mathbf{W}_{\text{ig}}\mathbf{x}_t + \mathbf{W}_{\text{hg}}\mathbf{h}_{t-1} + \mathbf{b}_{\text{g}}) \tag{2.4}$$

$$\mathbf{o}_t = \sigma(\mathbf{W}_{\text{io}}\mathbf{x}_t + \mathbf{W}_{\text{ho}}\mathbf{h}_{t-1} + \mathbf{b}_{\text{o}}) \tag{2.5}$$

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \mathbf{g}_t \tag{2.6}$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t) \tag{2.7}$$

where the $\mathbf{W}$s and $\mathbf{b}$s are the model parameters, and $\odot$ is the element-wise multiplication operator. The LSTM model has shown great improvement in a variety of sequential modeling tasks [54, 23, 86] by alleviating the vanishing and exploding gradient problem.

However, the LSTM model still struggles with long sequential data due to the information bottleneck. A Seq2Seq model based on a plain RNN (including LSTM) encodes the input sequence into a hidden representation and decodes it to the output. A key issue of this approach is that the hidden representation is a fixed-length vector, essentially serving as an information bottleneck between the encoder and decoder. Therefore, the performance of RNN models drops as the sequence gets longer, even if they do not have the vanishing and exploding gradient problem.

To address the information bottleneck, Bahdanau et al. [4] propose the attention mechanism, which greatly enhances the plain RNN. The attention model predicts the certain parts of the input sequence that are important to a specific decoding step. Since such attention is dynamic across different decoder steps, the model does not suffer from the information bottleneck.

Figure 2.3 shows the diagram of an attention mechanism. As seen, the attention

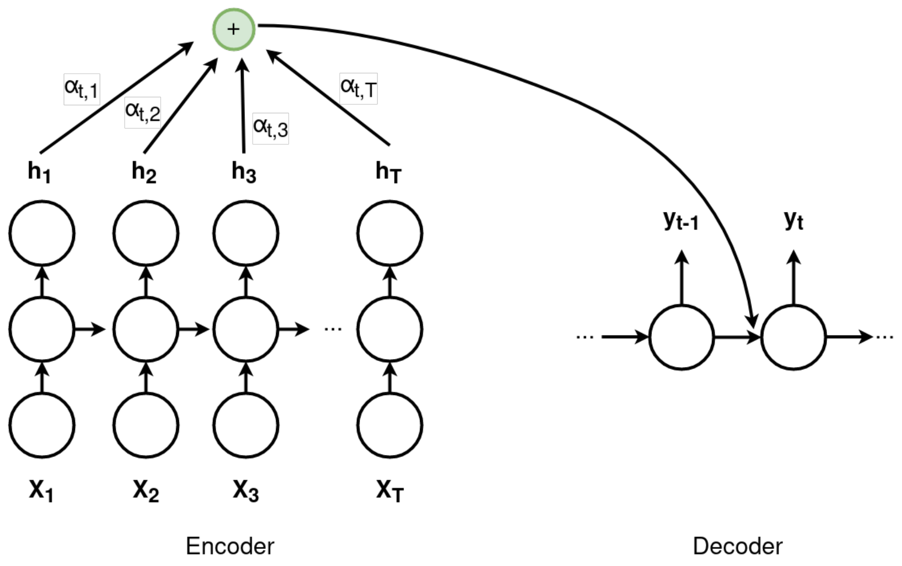

Figure 2.3: The attention mechanism in a encoder–decoder architecture, adapted from [4].

weights $\alpha_{t,1}, \alpha_{t,2}, \cdots, \alpha_{t,T}$, are first predicted to indicate the importance of each input x_i at the decoding time step t. Typically, it predicts higher attention weights for tokens that are more relevant to the decoding step. The encoder hidden states $\mathbf{h}_1, \mathbf{h}_2, \cdots, \mathbf{h}_T$ then form a context vector based on the attention weights, which is passed to the decoder to generate y_t.

Concretely, at each decoding time step i, the attention mechanism computes the context vector $\mathbf{c}_i$ as a weighted vector of the encoder hidden states, given by

$$\mathbf{c}_i = \sum_{j=1}^{T} \alpha_{ij} \mathbf{h}_j \tag{2.8}$$

$$\alpha_{ij} = \frac{\exp(\mathbf{e}_{ij})}{\sum_{k=1}^{T_x} \exp(\mathbf{e}_{ik})} \tag{2.9}$$

$$\mathbf{e}_{ij} = f(\mathbf{s}_{i-1}, \mathbf{h}_j) \tag{2.10}$$

where α_{ij} is the attention weight of the jth token in the input sequence, $\mathbf{h}_j$ is the jth encoder hidden vector, and f is an alignment function.

The decoder RNN then generates the next hidden state $\mathbf{s}_i$ based on the previous hidden state s_{i-1}, the previous prediction y_{i-1}, and the context vector $\mathbf{c}_i$

$$\mathbf{s}_i = \mathrm{RNN}(\mathbf{s}_{i-1}, y_{i-1}, \mathbf{c}_i) \tag{2.11}$$

The encoder, decoder, and attention model can all be trained in an end-to-end fashion because all the operations are differentiable.

2.2.2 The Transformer Architecture

The Transformer architecture replaces recurrent connections in RNNs with the attention mechanism and achieves state-of-the-art performance in most NLP tasks [75, 65, 66]. The general idea is that each layer performs multiple attentions (known as *multi-head attention*) to fuse information; there are also residual connections and layer normalizations, allowing deeper architectures. Details are deferred to Section 3.2. In addition to high performance, the Transformer models are also faster to train, because RNNs can only compute different time steps sequentially due to their recurrent connections, whereas the Transformer model can compute all time steps in parallel as it is fully based on attention. The faster training and superior performance make the Transformer one of the most commonly used architectures in NLP research.

The Transformer model can be used in different ways. For example, an encoder-only Transformer is widely used as a representation model: the famous BERT model is a bi-directional Transformer encoder [19] pretrained by masked language modeling (MLM) and next sentence prediction (NSP) in an unsupervised way on massive corpora. Here, the MLM task requires the model to predict tokens that are masked out, and the task encourages the model to build meaningful sentence representations. The NSP task, on the other hand, requires the model to predict whether a sentence is next to another given sentence. This further encourages the model to understand the relationship between sentences. Together, the two tasks allow the Transformer encoder to achieve state-of-the-art performance in various natural language understanding tasks.

The Transformer model can also be used in a decoder-only fashion. At every step, the decoder predicts a word and feeds the predicted word back in the next step, and thus, the model is able to accomplish text generation tasks. An example is the classic GPT-2 model [65], a uni-directional Transformer for language modeling. It is pretrained to predict the next token conditioned on the context (all previous tokens).

The Transformer architecture has also shown great success in the encoder–decoder model, which features a cross-attention mechanism, compared with encoder-only and decoder-only architectures. One example is the T5 model [66] pretrained on a combination of supervised tasks such as machine translation, summarization, and natural language inference.

2.3 Dialogue Systems

Dialogue generation is an important research direction of NLP that aims to converse with humans through text. In general, the dialogue task can be categorized into two main paradigms: task-oriented and open-domain. In the following subsections, I will first introduce these paradigms. Then, I will explain the metrics of open-domain dialogue systems, which are used in this thesis to evaluate our method.

2.3.1 Task-Oriented Dialogue Systems

Task-oriented dialogue systems aim to converse with humans to achieve specific goals such as hotel booking and restaurant finding [81, 9, 92]. Traditionally, these systems are implemented in a pipeline approach with four major components: a natural language understanding (NLU) unit, a dialogue state tracker, a dialogue policy, and a natural language generation (NLU) unit [13, 7, 80].

The NLU unit is responsible for understanding the users' goals. It involves two sub-tasks: intent detection and slot filling. Intent detection [78, 85] is a classification problem that predicts the intent of the user, for example, to "find a restaurant" and "specify a price range". On the other hand, slot filling [89, 52] is a sequential labeling problem, where the goal is to predict the word-level label for every input token to match a certain slot, for example, the restaurant name and the price range. They are used to achieve the goal of the dialogue. Overall, the NLU unit is responsible for capturing user inputs.

A dialogue state tracker is responsible for representing the current state of the dialogue. At every dialogue turn, the tracker considers the user's input and the history of the dialogue to update its state. Early systems [24, 42] use hand-crafted rules to choose the most likely state; they may not perform well due to the inherent ambiguity in natural language. Later benchmark datasets, such as the Dialogue State Tracking Challenges [84, 29, 30], represent states as a distribution instead. This allows the state to be tracked in a probabilistic manner, greatly enhancing the robustness of the systems [87].

The dialogue policy is the next stage of the pipeline: they generate system actions (e.g., to "request price range") based on the state from dialogue state trackers, and are usually trained through reinforcement learning [71, 63, 61], where the reward is based on the success or failure of achieving the goal. In general, the dialogue policy decides the type of the output utterance.

Finally, the NLG unit is responsible for generating natural language responses based on the actions of the dialogue policy. For example, Tran et al. [74] handle the action through a gating mechanism in the GRU unit, whereas Bordes et al. [7] treat the action as an additional input to an LSTM unit.

2.3.2 Open-Domain Dialogue Systems

Open-domain dialogue generation is the task of generating coherent and human-like utterances; it works in an open-ended fashion without a specific goal. These systems in general follow the encoder–decoder or the decoder-only architecture, and are trained in an end-to-end fashion with massive corpora. Unfortunately, they tend to generate generic responses such as *–I don't know* in the open-domain dialogue setting [45, 47, 79].

Wei et al. [79] hypothesize that the generic responses are due to the one-to-many mapping in dialogue datasets. Here, one-to-many mapping suggests that given an input context, there are a large number of plausible responses. Therefore, the target distribution is multi-modal, where each mode can be thought of as a potential topic for the responses. Maximum likelihood estimation, however, learns an overly smoothed function, which fails to capture different modes.

There are two main approaches to addressing the issue of the generic responses: training-time and inference-time. Usually, training-time approaches avoid the problematic maximum likelihood estimation. For example, Li et al. [45] apply reinforcement learning and optimize the ease of answering, which discourages generic responses that end a conversation abruptly. Khan et al. [38] apply adversarial learning to the latent space to address the same issue. In their setup, generic responses are discouraged because they can be easily detected by the discriminator. Wang et al. [77] modify the standard cross-entropy training with soft labels to encourage less frequent words. All these methods modify the training objective to address the generic responses in dialogue systems.

Inference-time approaches address the issue at the decoding stage. Vijayakumar et al. [76] propose the diverse beam search, which encourages distinct generations by adding a penalty to similar beam results. Holtzman et al. [33] propose nucleus sampling, which increases the diversity of dialogue responses by sampling less frequent words. Overall, the diversity issue in open-domain dialogue systems remains an active area of research.

There are also dialogue systems that combine the open-domain and task-oriented dialogue settings, such as the conversational Question Answering (QA) systems [68, 64, 40]. Traditionally, the QA task is formulated as either answer-span prediction [1, 67] or multiple-choice selection [31, 41]. These systems are not user-friendly due to the lack of interaction. Conversational QA systems alleviate this issue by allowing users to directly query the systems through natural language. They work in both open-domain and task-oriented styles: they need to converse with humans in an open-ended fashion while accomplishing the task of question answering.

Overall, both open-domain and task-oriented dialogue systems have great industrial applications and pose unique research challenges. This thesis focuses on open-domain dialogue systems, and I will explain the evaluation metrics in the next part.

2.3.3 Open-Domain Dialogue Evaluation

Open-domain dialogue evaluation mainly focuses on two aspects: response quality and diversity. I will explain these metrics in the rest of this subsection.

BLEU [59] is the main metric for measuring the quality of dialogue responses. At its core, the BLEU score considers the n-gram precision of a response given a list of reference responses. The precision is clipped by the number of n-gram hits to the maximum number of its appearances in different references; this discourages the system to repeat common words.

To evaluate the generated text at the corpus level, the n-gram precision p_n is computed as

$$p_n = \frac{\sum_{C\in\text{Candidates}} \sum_{n\text{-gram}\in C} \text{Count}_{\text{clip}}(n\text{-gram})}{\sum_{C'\in\text{Candidates}} \sum_{n\text{-gram}'\in C'} \text{Count}(n\text{-gram}')} \tag{2.12}$$

where the $\text{Count}_{\text{clip}}$ operator counts the number of matched n-grams clipped to the maximum occurrences in the references, and the Count operator counts the total number of the n-grams in the candidate input.

The precision is further penalized by its brevity. Such a brevity penalty (BP) ensures that overly short responses cannot achieve high performance. Finally, the BLEU score is computed as the geometric mean of n-gram precisions for $n = 1, \cdots, 4$. Formally, we have

$$\text{BP} = \begin{cases} 1, & \text{if} \quad c > r \\ e^{1-r/c}, & \text{if} \quad c \le r \end{cases} \tag{2.13}$$

$$\text{BLEU} = \text{BP} \cdot \exp\left(\frac{1}{4}\sum_{n=1}^{4} \log p_n\right) \tag{2.14}$$

where c is the sum of the lengths of candidate sentences in the corpus, and r is the sum of the lengths of the references.

The BLEU metric is not ideal for open-domain dialogue evaluation due to the uncertainty in the reference responses. Consider the golden input–reference pair: –*What is your hobby? –I'm an amateur tennis player.* A dialogue system may generate *I love food, so I enjoy cooking a lot.* This is a completely natural response but receives a low BLEU score because it shares few n-gram overlaps with the reference. Such a phenomenon does not occur in other NLP tasks such as machine translation, because the reference is semantically grounded. By contrast, dialogue responses are more diverse and often not semantically grounded. Therefore, open-domain dialogue evaluation remains an active area of research.

Learnable metrics are one way to address the issue of overlap-based dialogue evaluations. Instead of trying to match the words, learnable metrics attempt to learn the correspondence between context and response. For example, the RUBER metric [73] considers embedding-based similarity instead of n-gram overlapping. Further, the metric involves an unreferenced score, where a machine learning model predicts the fitness of a response without references. By combining both referenced and unreferenced metrics, the RUBER metric obtains a high correlation with human satisfaction.

Another aspect of dialogue evaluation is diversity. Open-domain dialogue systems tend to generate generic responses [45, 47, 38], such as *–I don't know.* Therefore, it is crucial to evaluate the diversity of dialogue responses in addition to the quality of a single response (which is often measured by the BLEU score).

The Dist metric [44] is one of the most widely used metrics to measure response diversity. The Dist-n score is computed as the percentage of unique n-grams across all the generations. A low Dist score indicates that the dialogue system tends to generate generic words. It is important to note that a higher Dist score does not necessarily imply a better dialogue system, as random generation achieves a very high Dist score. Therefore, it is common to consider multiple metrics such as BLEU and Dist to evaluate dialogue systems.

2.4 Summary

In this section, I first introduced different text generation models. I first started with the basic RNN model, which has the vanishing and exploding gradient problem. Then, I moved on to the LSTM model which addresses the issue. However, a plain LSTM encoder–decoder model is still problematic for modeling long sequential data due to the information bottleneck. Therefore, the attention mechanism was introduced. I also described state-of-the-art Transformer models: they completely replace RNNs' recurrent connections with attention mechanisms to achieve superior performance and faster training.

I then briefly discussed different types of dialogue systems: open-domain and task-oriented. Open-domain dialogue systems focus on open-ended conversations, whereas task-oriented dialogue systems focus on achieving specific tasks. However, these seemingly opposite paradigms are not mutually exclusive, as conversational question answering works in both styles. Finally, I discussed the main evaluation metrics that measure the quality and diversity of dialogue systems.

Chapter 3

Approach

3.1 Overview

In this chapter, I will provide details about our proposed Equal-Size Hard Expectation–Maximization (EqHard-EM) algorithm. In Section 3.2, I will first introduce the model, where we propose to enhance a standard sequence-to-sequence (Seq2Seq) framework with multiple decoders to address the one-to-many problem in dialogue systems; we further propose to leverage the adapter module [34] to reduce the number of parameters and to share knowledge among different decoders. In Section 3.3, we explain the training method for our multi-decoder model. Specifically, we follow the standard expectation–maximization approach, but modify the E-step in order to enforce an equal and hard assignment. We formulate it as an assignment problem, and solve it by the Hungarian algorithm [39]. Finally, we briefly explain how to apply our model to the dialogue generation task in Section 3.4.

3.2 Neural Architecture

Our model follows the general encoder–decoder architecture, shown in Figure 3.1a: the encoder encodes the context of the dialogue into real-valued hidden representations, whereas the decoder then decodes the hidden representations into a dialogue response. In particular, we use the Transformer architecture [75] for both the encoder and decoder.

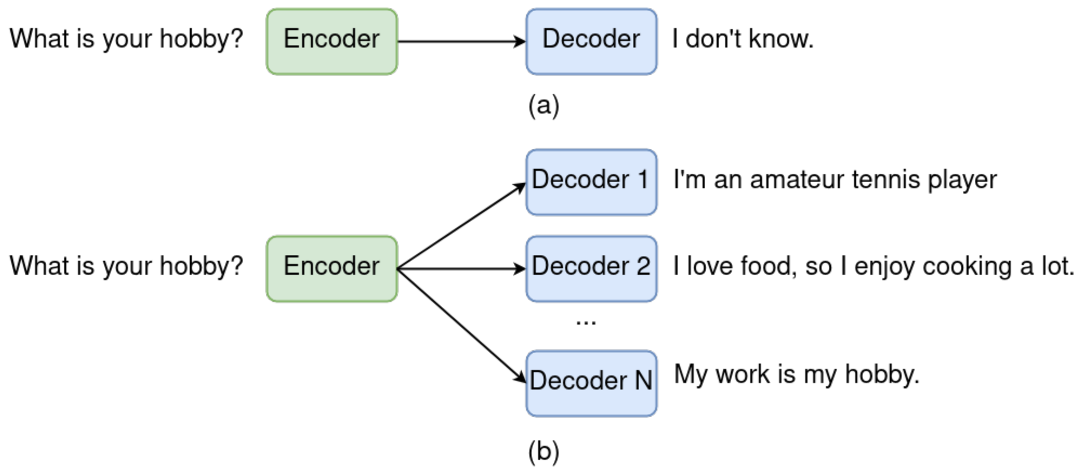

Figure 3.1: Model architecture for (a) single-decoder setup and (b) multi-decoder setup.

The key component of the Transformer model is the attention mechanism, which is used to replace recurrent connections in traditional recurrent neural networks [4]. The Transformer attention is computed using three matrices: $\mathbf{Q}$, $\mathbf{K}$, and $\mathbf{V}$, which represent the queries, keys, and values, respectively. Then, attention is computed with the following formula:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}}\mathbf{V}) \tag{3.1}$$

where d_k is the dimension of the keys.

Vaswani et al. [75] further improve the attention mechanism with multiple heads, allowing different attention heads to focus on different parts of the sequence. Concretely, the multi-headed attention is computed as:

$$\text{MultiHead}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{Concat}(\text{head}_1, \text{head}_2, \cdots, \text{head}_h)\mathbf{W}^O \tag{3.2}$$

where

$$\text{head}_i = \text{Attention}(\mathbf{Q}\mathbf{W}_i^Q, \mathbf{K}\mathbf{W}_i^K, \mathbf{V}\mathbf{W}_i^V) \tag{3.3}$$

Here, $\mathbf{W}_i^Q$, $\mathbf{W}_i^K$, $\mathbf{W}_i^V$, and $\mathbf{W}^O$ are all projection parameter matrices.

Figure 3.2 shows the Transformer encoder–decoder architecture that uses the attention mechanism extensively. As shown in Figure 3.2a, the Transformer layer in

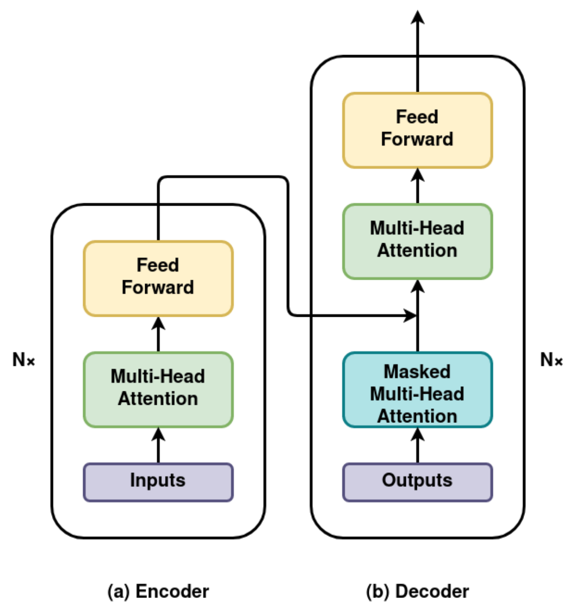

Figure 3.2: The Transformer architecture for the (a) encoder and (b) decoder. This diagram is adapted from [75]. Note that every block in the diagram contains an implicit residual connection and a follow-up layer normalization, and they are omitted for simplicity.

the encoder is composed of two sub-layers: the multi-head attention layer and the feed-forward layer. The Transformer decoder in Figure 3.2b has a similar architecture except for two major differences: first, the self-attention layer (i.e., attention within the decoder) is masked to ensure the decoder cannot attend to unseen tokens; second, an additional multi-head attention layer is added for the cross attention between the encoder and decoder. Such a Transformer encoder–decoder framework has shown great success in various NLP tasks, such as paraphrase generation [21], machine translation [75], and summarization [50].

While the Transformer encoder–decoder architecture has also largely advanced the dialogue generation task, it has been shown to generate generic responses [45, 47, 79]. Wei et al. [79] study this phenomenon and hypothesize that generic responses are due to the one-to-many mapping nature of the dialogue task. While such mapping may also exist in other text generation tasks, it is especially severe in dialogue generation, because a dialogue response may be open-ended and can vary both syntactically and

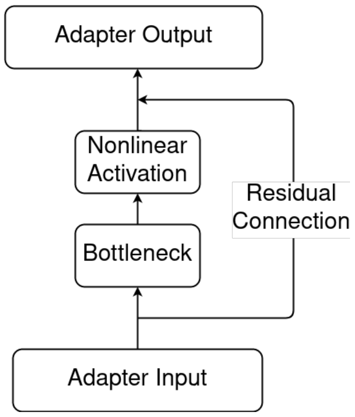

Figure 3.3: Adapter module. The diagram is adapted from [34].

semantically. Without proper treatment, the simple encoder–decoder architecture fails to "select" a mode; rather, the widely applied cross-entropy loss does mode averaging, which often results in generic responses such as *–I don't know.*

3.2.1 Multi-Decoder Model

To address the aforementioned one-to-many mapping, we propose to enhance the standard encoder–decoder architecture with multiple decoders. Figure 3.1b shows our multi-decoder model, with one encoder and multiple decoders. First, an encoder encodes the context into a hidden representation as before. Then, multiple decoders use the same hidden representation to generate diverse responses. Overall, the common encoder provides a shared understanding of the context, and the different decoders can learn to respond differently, similar to how different people respond differently. Given the input *–What is your hobby?*, for example, various utterances may fit as a response, such as *–I'm an amateur tennis player*, *–I love food, so I enjoy cooking a lot*, and *–My work is my hobby.* Ideally, each of our multiple decoders may capture one topic (e.g., sport, food, and work) for dialogue generation.

While it is intuitive and straightforward to instantiate a decoder with the full Transformer architecture, it has two important drawbacks. First, multiple Transformer decoders are not memory efficient: each decoder is usually a full Transformer

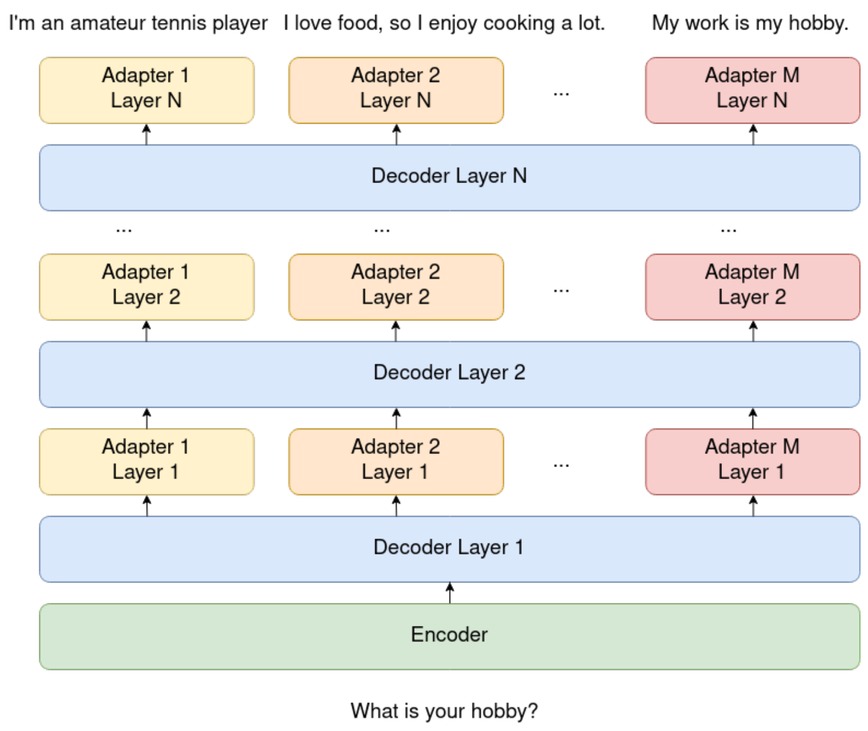

Figure 3.4: Multi-decoder model implemented with adapter modules.

model and takes substantial amounts of GPU memory. Second, complete separation of the decoder parameters means that there is no knowledge sharing among the decoders. However, all decoders solve the same task of response generation, which is similar even though they aim to cover different modes. Therefore, it is beneficial to have some degree of parameter sharing among these decoders.

To this end, we propose to use adapter modules [34] for both efficient memory usage and knowledge sharing among multiple decoders. As shown in Figure 3.3, the adapter module first projects its input to a bottleneck layer. Then it projects it back to the original input dimension after a nonlinear layer. Additionally, the module features a residual connection [28].

Formally, given an input n-dimensional input $\mathbf{x}$, the adapter module performs the following transformation:

$$\mathrm{AdapterModule}(\mathbf{x}) = \mathbf{x} + \mathbf{W}_{\mathrm{up}}(\phi(\mathbf{W}_{\mathrm{down}}\mathbf{x})) \tag{3.4}$$

where $\mathbf{W}_{\mathrm{up}}$ and $\mathbf{W}_{\mathrm{down}}$ are the weights for the linear up-projection and down-projection, respectively, and ϕ is an arbitrary nonlinear activation function.

In summary, we keep one complete Transformer decoder model, but insert multiple adapter modules, each representing a separate "decoder". While our proposed multi-adapter setup conceptually contains multiple decoders as shown in Figure 3.1b, each decoder only contains a few additional parameters for the adapter module. Thus, our model is parameter efficient. Figure 3.4 illustrates our proposed multi-adapter setup.

3.2.2 Model Pretraining

We propose to finetune a standard encoder–decoder model using the entire dataset to maximize knowledge sharing between different decoders. Specifically, we use a pretrained T5 model [66], because pretraining has been shown effective for improving performance [19, 65, 43].

Then, we warm-start the multi-adapter model by finetuning the T5 model with the standard cross-entropy loss. Concretely, consider a sample $\mathbf{x} = (\mathbf{c}, \mathbf{r})$, where $\mathbf{c} = \{c_1, c_2, \cdots, c_N\}$ is the context sequence, and $\mathbf{r} = \{r_1, r_2, \cdots, r_T\}$ is the response sequence. Then, the cross-entropy loss is defined as

$$\mathcal{L}(\mathbf{x}) = \sum_{t=1}^{T} -\log p_\theta(r_t | r_1, \cdots, r_{t-1}, \mathbf{c}) \tag{3.5}$$

where the p_θ is the predicted probability of generating the next token r_t, and θ represents all the parameters for the encoder–decoder model.

Notice that the fine-tuned Transformer parameters are frozen during the training of the multi-adapter model. This allows each decoder to specialize by only adjusting their respective adapter parameters and further maximizes the knowledge sharing among different decoders.

3.3 Equal-Size Hard EM Algorithm

In this part, I will first formulate the multi-decoder training by treating the choice of the decoder as an unobserved latent variable. Then, I will introduce the standard

Expectation–Maximization (EM) algorithm, including both Soft-EM and Hard-EM, with the highlight of the deficiencies of both algorithms, namely, the multiple decoders collapsing to a single decoder. To address the collapse issue, I will then introduce our Equal-Size Hard EM (EqHard-EM) algorithm, which extends Hard-EM by assigning an equal number of samples to each decoder. Finally, I will briefly explain the Hungarian algorithm, which is used to solve equal assignments.

3.3.1 Expectation–Maximization Training

We formulate the multi-decoder training as follows. Suppose there are K decoders in total. For some input context $\mathbf{c}$, we would choose one decoder $z \in \{1, \cdots, K\}$, which in turn generates a reply $\mathbf{r}$. The kth decoder predicts the probability as $p(\mathbf{r}|z = k, \mathbf{c}, \theta)$, where θ is the parameters of our entire model.

Our training follows the standard maximum likelihood estimation, except that z is a latent variable that is not observed in the dataset. Therefore, we need to marginalize it out. Formally, let a training sample be $(\mathbf{c}, \mathbf{r})$. The goal of our multi-decoder training is to maximize $p(\mathbf{r}|\mathbf{c}, \theta)$, the conditional log-likelihood of $\mathbf{r}$ given $\mathbf{c}$. The marginalization of the choice of the decoder z is given by

$$\log p(\mathbf{r}|\mathbf{c}, \theta) = \log \sum_{k=1}^{K} p(\mathbf{r}, z = k|\mathbf{c}, \theta) \tag{3.6}$$

The Expectation–Maximization (EM) algorithm [17] is a typical approach to the training of latent variable models. In particular, the algorithm breaks the optimization into two steps:

- E-step: we estimate the posterior distribution of the latent variable

$$p(z = k|\mathbf{r}, \mathbf{c}, \theta) \tag{3.7}$$

- M-step: we maximize the expected log-likelihood

$$\begin{aligned} & \mathbb{E}_{z\sim p(z|\mathbf{r},\mathbf{c},\theta)}[\log p(\mathbf{r}, z|\mathbf{c}, \theta)] && (3.8) \\ = & \sum_{k=1}^{K} p(z=k|\mathbf{r},\mathbf{c},\theta) \log p(\mathbf{r}, z=k|\mathbf{c},\theta) && (3.9) \end{aligned}$$

In other words, the EM algorithm resembles supervised maximum likelihood estimation, except that z is estimated by its posterior distribution instead of being given in the training set.

To compute the E-step, we apply Bayes' rule as

$$\begin{aligned} & p(z=k|\mathbf{r},\mathbf{c},\theta) && (3.10) \\ = & \frac{p(\mathbf{r}|z=k,\mathbf{c},\theta)p(z=k|\mathbf{c},\theta)}{p(\mathbf{r}|\mathbf{c},\theta)} && (3.11) \\ = & \frac{p(\mathbf{r}|z=k,\mathbf{c},\theta)p(z=k|\mathbf{c},\theta)}{\sum_{k'=1}^{K} p(\mathbf{r}, z=k'|\mathbf{c},\theta)} && (3.12) \\ = & \frac{p(\mathbf{r}|z=k,\mathbf{c},\theta)p(z=k|\mathbf{c},\theta)}{\sum_{k'=1}^{K} p(\mathbf{r}|z=k',\mathbf{c},\theta)p(z=k'|\mathbf{c},\theta)} && (3.13) \end{aligned}$$

We additionally make a uniform prior assumption

$$p(z=k|\mathbf{c},\theta) = \frac{1}{K} \tag{3.14}$$

which asserts that, without seeing the response, the probability of each decoder is the same. This differs from, say, the Gaussian Mixture Model (GMM) [6], where the prior is parameterized and learned. While the prior may also be learned in our multi-decoder model, we assume the uniform prior to fully exploit the capacity of all the decoders.

This further simplifies Equation (3.13) as

$$p(z=k|\mathbf{r},\mathbf{c},\theta) = \frac{p(\mathbf{r}|z=k,\mathbf{c},\theta)}{\sum_{k'=1}^{K} p(\mathbf{r}|z=k',\mathbf{c},\theta)} \tag{3.15}$$

Putting in (3.15) to (3.9) gives us the standard EM algorithm. It is also known as the **Soft-EM** because each sample is assigned to each decoder k in a soft manner.

However, the Soft-EM algorithm suffers from the collapse problem in our multi-decoder model for dialogue generation. In particular, all decoders have very similar parameters before training, because they are initialized with the same fine-tuned model (Section 3.2.2). The posterior (3.15), therefore, will also be similar during the training process, resulting in similar gradients for different decoders. As a result, different decoders tend to generate very similar responses, failing to specialize and capture different modes. We refer to this case as **synchronous-training collapse**.

Alternatively, (3.9) can be approximated by the single best decoder z, given by

$$k^* = \operatorname{argmax}_k p(z = k|\mathbf{r}, \mathbf{c}, \theta) \tag{3.16}$$

Then, the log-likelihood (3.9) becomes

$$\mathbb{E}_{z\sim p(z|\mathbf{r},\mathbf{c},\theta)}[\log p(\mathbf{r}, z|\mathbf{c}, \theta)] \approx \log p(\mathbf{r}, z = k^*|\mathbf{c}, \theta) \tag{3.17}$$

This is known as the **Hard-EM**, as each sample is assigned to one decoder in a hard manner. Hard-EM does not suffer from synchronous-training collapse, because different decoders are assigned with strictly different samples.

Unfortunately, Hard-EM suffers from another type of collapse: **non-training collapse**, where most decoders are untrained except for one or very few. This is because, while the decoders are similar at the beginning, one will nevertheless be selected by (3.16) and trained with some samples. Then, that decoder will give a higher probability to future samples and will be selected by (3.16) again. This is known as the "rich-gets-richer" phenomenon, making Hard-EM ineffective in our scenario.

3.3.2 Equal-Size Hard Assignment

To this end, we propose the Equal-Size Hard Expectation–Maximization (EqHard-EM) algorithm. Our approach adopts the idea of hard assignment, following the Hard-EM algorithm, to avoid the synchronous-training collapse in Soft-EM. However, we constrain the assignment such that each decoder gets the same number of samples.

This equal-assignment constraint helps our algorithm to avoid non-training collapse in Hard-EM, because all decoders are trained with adequate samples. Overall, our algorithm avoids both types of collapse seen in hard and soft EM.

Formally, we treat the E-step as a constrained optimization problem. Consider K decoders and N samples. The optimization variable is an assignment matrix $\mathbf{A} \in \{0,1\}^{K\times N}$ such that A_{kn} indicates whether the nth sample is assigned to the kth decoder. Since each sample must be assigned to some decoder, we must have $\sum_{k=1}^{K} A_{kn} = 1$ for every n. Also, our equal assignment requires $\sum_{n=1}^{N} A_{kn} = N/K$ for every k. Here, we assume N is a multiple of K, which can be easily achieved in batch implementation.

The optimization objective is to maximize the "fitness" of the decoder–sample assignment, which is to minimize $\sum_{k=1}^{K}\sum_{n=1}^{N} A_{nk}C_{nk}$, where $C_{nk} = -\log p(z = k|\mathbf{r}^{(n)}, \mathbf{c}^{(n)}, \theta)$ is the negative log-probability for the nth sample being assigned to the kth decoder.

In general, our optimization can be formulated as

$$\underset{\mathbf{A}}{\text{minimize}} \quad \sum_{k=1}^{K}\sum_{n=1}^{N} A_{nk}C_{nk} \tag{3.18}$$

$$\text{subject to} \quad \sum_{k=1}^{K} A_{nk} = 1, \quad \text{for } n = 1, \cdots, N \tag{3.19}$$

$$\sum_{n=1}^{N} A_{nk} = N/K, \quad \text{for } k = 1, \cdots, K \tag{3.20}$$

$$A_{nk} \in \{0,1\} \tag{3.21}$$

Such an optimization problem may be solved by standard integer linear programming (ILP) algorithms [69], if we reformulate the last constraint as $A_{nk} \geq 0$ and $A_{nk} \in \mathbb{Z}$. However, this may be inefficient due to the large number of constraints in our application.

In our thesis, we reformulate the optimization as a balanced assignment problem, where the numbers of samples and decoders are the same. Then the problem can be

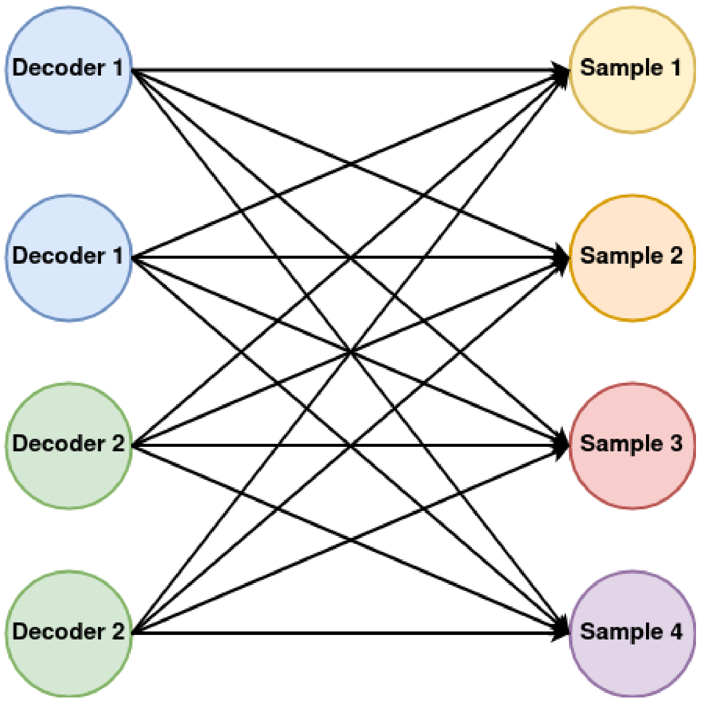

Figure 3.5: Formulating an equal-assignment problem with two decoders and four samples as a balanced assignment problem.

solved by the Hungarian algorithm, explained in the next part.

3.3.3 Hungarian Algorithm

The Hungarian algorithm [39] is a classic algorithm for solving balanced assignment problems between jobs and workers; in our case, they are samples and decoders, respectively. In a balanced assignment problem, the numbers of workers and jobs are the same, which can be formulated as

$$\begin{aligned} \underset{\mathbf{A}}{\text{minimize}} \quad & \sum_{k=1}^{N}\sum_{n=1}^{N} A_{nk}C_{nk} && (3.22)\\ \text{subject to} \quad & \sum_{k=1}^{N} A_{kn} = 1, \quad \text{for } n = 1, \cdots, N && (3.23)\\ & \sum_{n=1}^{N} A_{kn} = 1, \quad \text{for } k = 1, \cdots, N && (3.24)\\ & A_{nk} \in \{0, 1\} && (3.25) \end{aligned}$$

This is essentially the same as our previous equal-assignment formulation, except that every worker k should be assigned with exactly one job, as shown by constraint (3.24).

Our equal-assignment problem can be easily reformulated as a balanced assignment problem by duplicating the decoders to match the number of samples. Recall that we assume the number of samples N is a multiple of the number of decoders K, which can be easily achieved by batch implementation. Thus, we make N/K copies of each decoder to match the number of samples. The cost for the duplicated decoder remains the same as the original one, i.e., $C_{nk'} = C_{nk}$ for some decoder k' duplicated from decoder k.

Figure 3.5 demonstrates how our reformulation works with two decoders and four samples. The assignment problem is shown as a bipartite graph, where the workers (i.e., decoders) are on the left-hand side, and the jobs (i.e., samples) are on the right-hand side. The edges represent the cost for each assignment. As shown, both decoders have two copies, effectively transforming the equal-assignment problem into a balanced assignment problem. This reformulation allows us to apply the Hungarian algorithm to solve the equal-assignment problem in our multi-decoder setup.

The Hungarian algorithm [39] works by manipulating the cost matrix by adding and subtracting constants in the rows and columns, which does not change the solution of the assignment problem. By doing so, the Hungarian algorithm reduces the matrix into an equivalent cost matrix that contains a large number of zero entries. The zeros in the matrix are then used to find the optimal assignment.

Let $\mathbf{C} \in \mathbb{R}_{+}^{N\times N}$ be a non-negative cost matrix, where C_{ij} represents the cost of assigning job i to worker j. The Hungarian algorithm performs the following steps:

Step 1. Reduce the rows by subtracting the minimum entry in each row.

$$C_{ij} = C_{ij} - \min(C_{i1}, C_{i2}, \cdots, C_{iN}) \tag{3.26}$$

Step 2. Reduce the columns by subtracting the minimum entry in each column.

$$C_{ij} = C_{ij} - \min(C_{1j}, C_{2j}, \cdots, C_{Nj}) \tag{3.27}$$

Step 3. Draw horizontal or vertical lines on the matrix to cover all the zeros. If at least N lines are required to cover all zeros, then skip to Step 5. If less than N lines are required to cover the zeros, proceed to Step 4.

Step 4. Find the minimum $C_{\min}$ of the entries that are not covered by any lines. Then subtract it from the uncovered entries and add it to the entries covered by two lines (intersections). For entries not covered, apply:

$$C_{ij} = C_{ij} - C_{\min} \tag{3.28}$$

whereas for entries covered by two lines, apply:

$$C_{ij} = C_{ij} + C_{\min} \tag{3.29}$$

After these manipulations, go back to Step 3.

Step 5. Form the optimal assignment by choosing the zero cost entries in the updated cost matrix, where every worker is assigned with one and only one job.

Intuitively, the Hungarian algorithm manipulates the cost matrix while maintaining the relative costs between workers and jobs. These manipulations additionally keep the cost strictly non-negative, making zero-cost assignments the optimal local choices. Then, the algorithm terminates when there are enough zeros in the matrix such that every job is assigned to a worker at zero relative cost. The Hungarian algorithm can effectively solve our equal-assignment problem when reformulated as a balanced assignment problem.

3.4 Inference

Our multi-decoder model may be used in different ways during inference for response generation. In general, there are two main settings that are common in the existing literature: diverse multi-response generation [90, 25] and single response generation [11, 77].

Diverse multi-response generation aims to generate multiple responses for one given context. Our multi-decoder model can accomplish this naturally by using each decoder to generate a response suppose we have enough decoders. If we would like to generate more responses, then we may apply traditional sampling techniques [33].

Single response generation, on the other hand, requires only one response for every input context. In this case, we may obtain a candidate response with each decoder and choose the "best" candidate as the output. Specifically, we propose a length-normalized confidence score for candidate selection, which is the average of the predicted log-probabilities.

Given context $\mathbf{c}$, the kth decoder predicts the response $\mathbf{r}^{(k)} = \{r_1^{(k)}, \cdots, r_{T_k}^{(k)}\}$ in a greedy manner. At time step t, the predicted token is given by

$$r_t^{(k)} = \operatorname*{argmax}_{r \in V} p_k(r|\mathbf{r}_{<t}, \mathbf{c}, \theta) \tag{3.30}$$

where V is the vocabulary, and $p_k(r|\mathbf{r}_{<t}, \mathbf{c}, \theta)$ is the kth decoder's predicted probability conditioned on the context and previously predicted tokens.

We further define the confidence score of the kth decoder as

$$\text{log-confidence}(\mathbf{r}^{(k)}, \mathbf{c}) = \frac{1}{T_k} \sum_{t=1}^{T_k} \log p_k(r_t^{(k)}|\mathbf{r}_{<t}, \mathbf{c}, \theta) \tag{3.31}$$

Eventually, we would choose the most confident decoder as

$$k^* = \text{argmax}_k \, \text{log-confidence}(\mathbf{r}^{(k)}, \mathbf{c}, \theta) \tag{3.32}$$

and yield $\mathbf{r}^{(k^*)}$ as the generated output in the single-response setting.

3.5 Summary

In this chapter, we went through three important aspects of our proposed dialogue system: the neural architecture, the training, and the inference.

We started by understanding the mode averaging issue in single-decoder models, which often generate generic responses. To address the issue, we proposed a multi-decoder model, allowing different decoders to specialize and cover separate subsets of modes.

We then focused on the training of our multi-decoder model by formulating it as a latent variable model. In our case, the selection of the decoder is treated as the unobserved latent variable. Although the EM algorithm is a principled approach to train latent variables. We found that the classic Hard-EM and Soft-EM algorithm variants both have collapse issues: Soft-EM suffers from synchronous-training collapse, whereas Hard-EM suffers from non-training collapse. We then proposed our EqHard-EM algorithm to address both kinds of collapse by applying an equal and hard assignment. To achieve that, we formulated the equal-assignment problem as a balanced assignment problem, which can be solved with the Hungarian algorithm.

Our last focus was on multi-decoder inference. Since we have multiple decoders, the model can naturally generate multiple diverse responses, which is required in the multi-response generation setting. On the other hand, single response generation requires the model to select one response among the candidate responses as the model output. To this end, we proposed to select by model confidence, allowing the most confident decoder to be used for the given context.

Chapter 4

Experiments

4.1 Overview

In this chapter[1], I will first provide details in Section 4.2 about the two datasets that we use for evaluation: Weibo and OpenSubtitles. We observe that the OpenSubtitles dataset has the overlapping problem, so we clean the dataset before using it to benchmark our models. In Section 4.3, I will explain our evaluation, which features both quality and diversity metrics. Sections 4.4 and 4.5 will focus on our main results for Weibo and OpenSubtitles, respectively. Then, Section 4.6 will compare individual decoders quantitatively in terms of their individual performance as well as their generation lengths. Finally, I will provide case studies for decoder outputs in Section 4.7.

4.2 Datasets

We evaluated our proposed EqHard-EM algorithm in two popular open-domain dialogue datasets: Weibo [22] and OpenSubtitles [49]. In our previous work [82], we find that the original OpenSubtitles dataset contains overlapping samples, so we used the cleaned dataset instead. The two datasets are explained in detail as follows.

The Weibo dataset is constructed by [22]. It contains around 4 million training

[1]Figures 4.1 and 4.2 and part of the text related to the OpenSubtitles dataset are reused from our published paper [82]. ©European Language Resources Association (ELRA), licensed under CC-BY-NC-4.0.

samples, crawled from the Chinese social micro-blogging website, Weibo; each sample consists of a post (context) and a reply (response). The Weibo dataset is unique as most posts have multiple replies, so the dataset has explicit one-to-many mappings.

We additionally evaluated our model on the OpenSubtitles dataset [49]. The dataset is a collection of movie subtitles that have been processed initially for machine translation research. However, the dataset has also gained popularity for dialogue research, where consecutive utterances in the movies are treated as context–response pairs. Unlike the posts and responses in Weibo, the utterances in movies do not naturally have the explicit one-to-many mapping. Therefore, the OpenSubtitles dataset helps us to understand how well our model works with implicit one-to-many mapping. As mentioned, the original OpenSubtitles dataset contains overlapping samples. The details are explained in the rest of this section.

4.2.1 The Overlapping Issue of Dialogue Datasets

We find that open-domain dialogue datasets are prone to the overlapping problem, where duplicate samples exist across the training, validation, and test splits. In this part, I will present the statistics and consequences with the OpenSubtitles dataset as an example.

The overlapping problem is characterized by similar samples across the training, validation, and test splits. We propose a metric, called the overlap ratio, to quantify the amount of word overlapping between a test sample and the training set. Specifically, we represent an utterance as a bag of words, and compute the overlap ratio between two utterances $\mathbf{u} = \{u_1, \cdots, u_m\}$ and $\mathbf{v} = \{v_1, \cdots, v_n\}$ as $R(\mathbf{u}, \mathbf{v}) = \frac{2|\mathbf{u} \cap \mathbf{v}|}{|\mathbf{u}|+|\mathbf{v}|}$. A data sample is a tuple $\mathbf{x} = (\mathbf{c}, \mathbf{r})$, where $\mathbf{c}$ is the context and $\mathbf{r}$ is the response. We then compute the overlap ratio of two samples $\mathbf{x} = (\mathbf{c}, \mathbf{r})$ and $\mathbf{x}' = (\mathbf{c}', \mathbf{r}')$ as $R(\mathbf{x}, \mathbf{x}') = \min\{R(\mathbf{c}, \mathbf{c}'), R(\mathbf{r}, \mathbf{r}')\}$; the min operator rules out false overlapping caused by generic utterances (e.g., *hello*) for either the context or the response. Finally, the overlap ratio of a test sample $\mathbf{x}$ against the training dataset $\mathcal{D}_{\text{train}}$ is given by

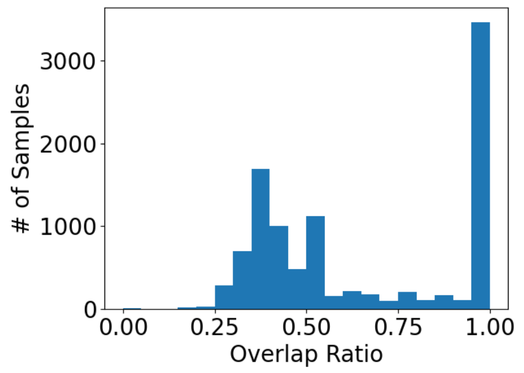

Figure 4.1: Overlap histogram of test samples against the training set on OpenSubtitles.

$R(\mathbf{x}, \mathcal{D}_{\text{train}}) = \max_{\mathbf{x}' \in \mathcal{D}_{\text{train}}} R(\mathbf{x}, \mathbf{x}')$.

Figure 4.1 shows the histogram of the overlap ratio in OpenSubtitles. We find that 34.49% of test samples are identical to training samples, highlighting the overlapping problem. Such overlapping does not naturally arise from human conversations; rather, they are caused by oversights in data collection and preprocessing. For OpenSubtitles, Lison et al. [49] collect subtitles based on their IMDb identifiers. However, we find that the same movie may correspond to different identifiers if it has different versions. For example, the South Korean movie My Sassy Girl[2] and its American remake[3] contain highly overlapping dialogues, but they are treated as different movies based on their IMDb identifiers.

We further observe that overlapping samples have undesired effects on training dialogue systems. We compare the learning curves on the original test set and a deduplicated test set, where we remove samples with an overlap ratio of greater than 0.80. In terms of the neural architecture, we fine-tune a T5-small model [66] here.

Figure 4.2 shows the learning curves in terms of the BLEU-2 metric. The model achieves ∼15 BLEU-2 with the original test set; however, the same model only achieves ∼3 BLEU-2 after deduplication. The results suggest that overlapping sam-

[2]https://www.imdb.com/title/tt0293715/
[3]https://www.imdb.com/title/tt0404254/

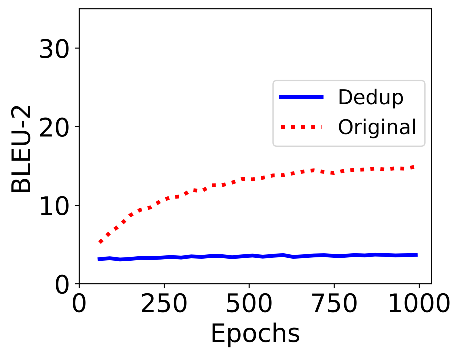

Figure 4.2: BLEU-2 learning curve for the original dataset and the deduplicated dataset for OpenSubtitles. Samples with an overlap greater than 0.80 are considered duplicates and are removed for the deduplicated dataset.

ples heavily inflate BLEU scores, and that the high performances of the alleged "state-of-the-art" models mainly come from memorizing training samples.

In addition, we observe that it takes more epochs for the test set performance to converge when the test samples overlap with the training set. For example, the BLEU-2 score still increases after 1000 epochs. Since most researchers do not train their models for 1000 epochs, their reported performance may be arbitrary, depending on where the model ends up along the learning curve. This highlights the inconsistency of reported performances.

In summary, our qualitative and quantitative analyses show that it is fundamentally flawed to evaluate open-domain dialogue systems on overlapping datasets, which are unfortunately commonly used (and benchmarked) in current research. The next subsection provides a data-cleaning strategy to address this issue.

4.2.2 Dataset Cleaning

We make efforts to clean existing open-domain dialogue datasets. In the literature, both single-turn and multi-turn dialogues are common settings [12, 91, 77, 26]. We propose to deduplicate data samples in the multi-turn setting, i.e., an entire movie in OpenSubtitles is treated as a unit for deduplication. Such a unit can be further split into single turns, so our treatment unifies both settings.

Further, we propose to re-split the training, validation, and test sets of the original corpora during deduplication. If we simply remove the duplicate samples, then at least one of the validation and test sets will be heavily shrunk due to massive overlapping samples, making evaluation noisy and unreliable.

Let us consider two units $\mathbf{u}$ and $\mathbf{v}$ (dialogue sessions or movies) for deduplication. Their overlap ratio is defined as

$$R(\mathbf{u}, \mathbf{v}) = \frac{2|\mathbf{u} \cap \mathbf{v}|}{|\mathbf{u}| + |\mathbf{v}|} \tag{4.1}$$

The overlap ratio of a unit $\mathbf{u}$ is then computed as the maximum overlap against all other units in the dataset $\mathcal{D}$:

$$R(\mathbf{u}, \mathcal{D}) = \max_{\mathbf{u}' \in \mathcal{D} \setminus \{\mathbf{u}\}} R(\mathbf{u}', \mathbf{u}) \tag{4.2}$$

Note that the above overlap ratio is similar to the one in Section 4.2.1. However, (4.2) considers the ratio of a deduplication unit (an entire dialogue session or movie), whereas Section 4.2.1 considers the ratio of a single-turn conversation. Thus, we do not have the min operator here. Further, the ratio of a unit is computed against the rest of the corpus, whereas Section 4.2.1 computes the ratio of a test sample against the training set. Thus, we have the set minus operator in (4.2).

Then, we iterate through all deduplication units in the corpus. If a unit's overlap ratio exceeds a threshold (set to 0.8), then we remove the unit but keep the one that it overlaps the most with. Since there may be more than two units overlapping with each other, we need to repeat the procedure multiple times. Specifically, we recompute the overlap ratios after each iteration through the dataset, and remove additional duplicate samples until convergence.

After that, we split the deduplicated units into training, validation, and test sets. While our deduplication unit is an entire dialogue session or a movie, the resulting corpus can serve for the settings of single-turn and fixed-turn contexts. We simply split a unit into multiple tuples of contexts and responses. This may result in (a

Version	# Train	# Validation	# Test
Wang et al. [77]	1,144,949	20,000	10,000
Cleaned	979,230	11,982	12,152

Table 4.1: Statistics of the original and cleaned OpenSubtitles datasets.

small number of) additional duplicate samples due to the generic conversations, such as *–Thank you. –You're welcome.* Therefore, we further remove exactly overlapping context–response pairs.

Table 4.1 shows the data statistics for our cleaned datasets in the single-turn setting, which is adopted in our experiments. Our data-cleaning strategy enables us to control the size of validation and test sets.

4.3 Metrics

We compare our proposed models with baseline approaches in terms of the following metrics.

BLEU. The BLEU score [59] is a standard metric that measures the similarity between the reference and the model output, and the score is commonly used in dialogue research [90, 22, 77] to measure the quality of the generated responses. Details of the BLEU score are in Section 2.3.3.

Dist. The Dist score [44] is a measure of the diversity of the generated responses, which is another commonly used metric for diverse dialogue generation. Refer to Section 2.3.3 for more details.

pairwise-BLEU. We additionally follow [70] and adopt their pairwise-BLEU to measure the diversity among different responses for the same model input. The pairwise-BLEU measures the pairwise BLEU score among the model generated outputs. Let $R = \{r_1, r_2, \cdots, r_N\}$ be a set of responses and $P = \{(i, j) \mid i \in [1, N], j \in [1, N], i \neq j\}$ be the set of pairwise comparisons. Then the pairwise-BLEU is defined as

$$\text{pairwise-BLEU(R)} = \frac{1}{|P|} \sum_{(i,j)\in P} \text{BLEU}(r_i, [r_j]) \tag{4.3}$$

where BLEU(hypothesis, [reference]) is the function that calculates the standard BLEU score between the hypothesis and a reference. A high pairwise-BLEU is undesired as it shows that all model outputs are highly similar. The pairwise-BLEU complements the Dist metric by focusing on the diversity between the model outputs rather than across the model outputs.

best-BLEU. We additionally evaluate our dialogue systems in terms of the best BLEU score, which shows the performance upper bound for multi-response generation models.

$$\text{best-BLEU}(R) = \max_{r\in R} \text{BLEU}(r, [\text{ref}_1, \cdots]) \tag{4.4}$$

where R is the set of model responses, and $[\text{ref}_1, \cdots]$ is a list of references. In our multi-decoder model, the best-BLEU shows how the model performs if we can make the best choice of the decoder for each input.

4.4 Results on the Weibo Dataset

We first evaluate our approach on the Weibo dataset, which features the one-to-many phenomenon explicitly. We consider the following competing models:

- **T5-small.** We use the T5-small [66] model as the standard Transformer encoder–decoder baseline. T5 is pretrained on various NLP tasks, including machine translation, summarization, and question answering. Such pretraining has shown great success in improving downstream performance, and the model serves as a strong baseline for our multi-decoder models.

- **Multi-Decoder Model.** Our multi-decoder model uses the finetuned T5-small as a starting point. We then insert adapter modules at each layer, allowing different adapters to behave as separate decoders. Additionally, the parameters of the finetuned T5-small is frozen to encourage maximum knowledge sharing among decoders. The multi-decoder model is then trained with different EM variants applied at the batch level:

– **Hard-EM.** Hard-EM assigns samples in a binary fashion. A sample is only assigned to the decoder that has the highest posterior fitness among the other decoders. Other decoders do not get partial assignment for the sample.
– **Soft-EM.** Unlike Hard-EM, Soft-EM assigns samples in a soft manner, where each decoder is assigned with a portion of a sample based on the posterior fitness.
– **EqHard-EM.** Our proposed EqHard-EM algorithm applies the Hungarian algorithm to obtain equal and hard assignment.
– **EqDynamicRandom.** Rather than applying the EM algorithm, this approach dynamically assigns an equal number of samples to each of the decoders. As training continues, every decoder will be trained on all the samples, so this approach should perform similarly to the standard encoder–decoder model, except that it has extra adapter parameters.
– **EqFixedRandom.** This approach also assigns samples randomly. However, unlike EqDynamicRandom, the random assignment is persistent and does not change across epochs.

Table 4.2 shows the performance of our approach and the competing models. As seen, Hard-EM has lower BLEU performance than the T5-small model, which is caused by the non-training collapse. During Hard-EM training, we observe that most of the samples are assigned to one decoder, so only one decoder is properly trained. However, the inadequately trained decoders also participate during the inference process, leading to the degradation of overall BLEU performance. The non-training

Model	BLEU	best-BLEU	pairwise-BLEU	Dist-1	Dist-2
T5-small	7.20	-	-	10.41	37.37
Hard-EM	6.23	10.75	1.20	13.41	48.93
Soft-EM	7.97	10.74	37.46	11.66	44.13
EqDynamicRandom	8.66	11.15	41.97	11.58	43.64
EqFixedRandom	9.88	13.85	19.63	12.64	49.93
EqHard-EM	10.34	15.20	10.92	14.33	57.32

Table 4.2: Main results on the Weibo dataset. Note that all models except T5-small is fine-tuned on 10% of the Weibo dataset due to limited computing resources.

collapse is also shown by the low pairwise-BLEU score and the high Dist scores: the inadequately trained decoders tend to generate random utterances, and therefore the multi-decoder model appears to generate diverse responses. Our experiment demonstrates that Hard-EM suffers from non-training collapse, and it is not suitable for training multi-decoder models.

We also observe that Soft-EM suffers from the synchronous-training collapse. Although every decoder is trained with the Soft-EM algorithm, they are mostly trained synchronously. This causes all decoders to generate similar responses, which is shown by the high pairwise-BLEU score. Despite the synchronous-training collapse, we observe that Soft-EM still outperforms the T5-small baseline in terms of both BLEU and Dist. The slight performance improvement may be attributed to a partial collapse, allowing the multi-decoder model to partially address the mode averaging problem. This result confirms the viability of EM-style training, and it highlights the importance of addressing the collapse issue. Overall, we find Soft-EM also ineffective for multi-decoder training due to synchronous-training collapse.

EqDynamicRandom is another baseline that outperforms the T5-small model. Due to the random assignment during the training, the EqDynamicRandom algorithm asymptotically degenerates into training every decoder with all the training samples. Such training eventually causes all decoders to be similar, achieving the highest pairwise-BLEU score among all models. However, we still observe a significant performance improvement over the T5-small baseline. This is probably because the resulting multiple decoders may still differ from each other more or less, and EqDynamicRandom is essentially an ensemble model [88].

The EqFixedRandom variant randomly assigns samples to decoders beforehand, and such assignment is fixed during training. We see that EqFixedRandom outperforms all the above methods significantly in terms of both BLEU and Dist while achieving the lowest pairwise-BLEU so far. This shows that the idea of allowing different decoders to specialize in different subsets of samples is the key to addressing the mode averaging issue. Overall, EqFixedRandom sets up a strong baseline performance for variants of the EM algorithms.

We then compare our proposed EqHard-EM algorithm against all these baselines. Overall, our EqHard-EM algorithm achieves the best performance across all models. In particular, it achieves the highest BLEU performance while maintaining the diversity in its generations as demonstrated by the high Dist scores. Our EqHard-EM algorithm also successfully addresses the collapse issue in both Soft-EM and Hard-EM, as it achieves significantly lower pairwise-BLEU than all other baselines except EqHard-EM. However, the low pairwise-BLEU score for EqHard-EM is due to random generations from inadequately trained decoders, which is not the case for EqHard-EM. The results show that our EqHard-EM algorithm is successful and improves the other EM variants by addressing the mode averaging issue.

4.5 Results on the OpenSubstitles Dataset

We also evaluate our models on the cleaned OpenSubtitles dataset, where we compare them against both state-of-the-art and classical models, detailed as follows.

- **LSTM with Attention.** We include the long short-term memory network [32] with an attention mechanism [4] as a baseline. This is a standard model before

Model	BLEU	best-BLEU	pairwise-BLEU	Dist-1	Dist-2
LSTM[†]	1.41	-	-	3.10	14.94
Transformer[†]	1.29	-	-	3.05	13.88
GPT-2[†]	2.15	-	-	2.98	11.37
T5-small[†]	2.07	-	-	2.78	8.87
EqFixedRandom	2.33	3.80	33.32	4.08	14.62
EqHard-EM	2.34	5.11	16.08	4.68	15.83

Table 4.3: Main results on the OpenSubtitles dataset. [†]Results for these models are quoted from [82].

the Transformer era.

- **Transformer.** In current research, the Transformer [75] is the most commonly used architecture. It replaces LSTM's recurrent connections with a multi-head attention mechanism and achieves superior performance in different NLP tasks. In this baseline, the Transformer is not pretrained.

- **GPT-2.** The GPT-2 model [65] adopts the Transformer architecture, but is pretrained on massive unlabeled corpora. Pretraining is shown to benefit various downstream tasks.

- **T5-small.** The T5 model [66] also uses the Transformer architecture, but works in an encoder–decoder fashion. It is pretrained on a number of text generation tasks, such as translation and summarization. We adopt the T5-small version in our experiments.

Table 4.3 shows the results on our cleaned OpenSubtitles dataset. Note that since the quoted models only perform single-response generation, their best-BLEU and pairwise-BLEU are missing. Due to limited computing resources, we only evaluate the performance for EqHard-EM and EqFixedRandom, since they are the two most successful variants from our preliminary experiments on the Weibo dataset.

Decoder Index	EqFixedRandom				EqHard-EM			
	BLEU	Dist-1	Dist-2	% Used	BLEU	Dist-1	Dist-2	% Used
1	5.21	9.91	37.79	8.53	6.94	11.38	41.83	2.89
2	5.39	9.70	36.24	14.57	8.65	12.26	47.39	14.44
3	5.14	9.98	39.37	12.47	5.86	10.79	39.01	5.12
4	4.58	10.04	38.99	6.82	9.48	14.44	58.63	30.58
5	5.84	10.76	41.04	9.58	5.75	9.03	34.07	1.71
6	5.46	10.02	39.04	9.06	3.21	7.00	23.34	12.20
7	4.98	10.02	39.24	9.71	4.41	7.50	27.21	1.31
8	5.34	10.70	40.73	9.06	9.21	12.87	51.48	19.29
9	4.46	10.08	37.95	9.19	4.39	8.93	30.76	9.45
10	5.39	10.40	40.18	11.02	4.80	7.72	26.46	3.02

Table 4.4: Individual decoder analysis for EqFixedRandom and EqHard-EM on the Weibo dataset.

As seen, our strong baseline EqFixedRandom outperforms all other standard encoder–decoder models in terms of both BLEU and Dist, except it has slightly lower Dist-2 than the LSTM model. This confirms that the multi-decoder model and our equal assignment are successful in improving standard encoder–decoder models.

Further, our EqHard-EM outperforms EqFixedRandom in every aspect and achieves the highest performance in terms of both BLEU and Dist. EqHard-EM also achieves significantly lower pairwise-BLEU than EqFixedRandom, which shows that our algorithm is successful in allowing different decoders to specialize and generate different responses when given the same context. Overall, our results on OpenSubstitles demonstrate that EqHard-EM works well even when there is no explicit one-to-many phenomenon in the dataset.

4.6 Decoder Analysis

We conduct extensive analysis to better understand how our EqHard-EM algorithm affects individual decoders. Table 4.4 shows the performance and decoder usage of each decoder in the EqFixedRandom and EqHard-EM models. The decoder usage refers to the percentage of times that a decoder is chosen to output the response based on its log-confidence score in (3.31). Note that the decoders between EqFixedRandom and EqHard-EM are unrelated even though they share the decoder indices.

We observe that the performance is relatively uniform across different decoders in the EqFixedRandom model. As shown, all decoders achieve a BLEU score of $\sim$5 and a Dist-1 score of $\sim$10. In addition, they also have mostly uniform usage, with each of the decoders being used roughly 10% of the time. This is unrealistic if the decoders are truly specialized. We expect specialized decoders to have more different overall performance, as they would all excel in different ways. Overall, the decoders of EqFixedRandom appear to behave similarly, which is understandable as the decoders are simply trained with random subsets of the data.

EqHard-EM, on the other hand, achieves drastically different performance across different decoders. For instance, Decoder 5 achieves the lowest BLEU score of 3.21, whereas Decoder 3 achieves the highest score of 9.48. There is also a variety in terms of diversity performance, with the Dist-1 score ranging from 7.00 to 14.44. This shows that the decoders are very different. Certain decoders perform well in general; for example, Decoder 3 achieves a high BLEU score of 8.65. Other decoders are more specialized; for example, Decoder 5 may only perform well in specific cases, achieving a relatively low overall BLEU score. The results suggest that our EqHard-EM algorithm is successful in addressing the mode averaging issue, allowing decoders to specialize in different ways.

We further notice that the specialized decoders from EqHard-EM are well utilized to achieve strong performance. In particular, the full model outperforms all individual decoders in terms of BLEU and Dist, except for Decoder 3 whose Dist scores are slightly higher. This shows that our strong performance is in fact achieved by combining multiple specialized decoders rather than having a particularly well-trained decoder. Notably, Decoder 3 is used 12.20% of the time, even though it has the worst

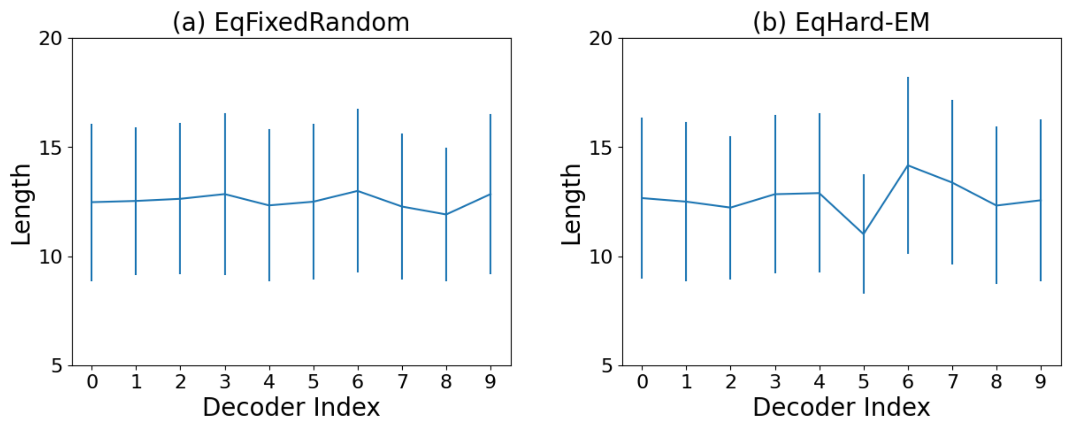

Figure 4.3: Average response lengths for (a) EqFixedRandom and (b) EqHard-EM for individual decoders. The error bars represent the standard deviation.

overall BLEU performance of 3.21. This shows that EqHard-EM enables decoders to specialize and work in a complementary fashion, confirming its ability to address the mode averaging issue.

We further analyze the decoders by examining the lengths of their responses. Figure 4.3 plots the average response lengths of individual decoders, where the error bars represent the standard deviation. For EqFixedRandom, we see that the generated lengths are again similar across different decoders: they all generate around ∼12 words on average. By contrast, we see more variations for EqHard-EM. For instance, Decoder 5 generates the shortest responses with ∼11 words on average, whereas Decoder 6 generates ∼14 words on average. This explains the low overall performance of Decoder 5: short responses are brevity-penalized more by the BLEU metric in general. Overall, our length analysis shows that EqHard-EM allows decoders to specialize in generating responses at different lengths.

4.7 Case Study

We show decoder outputs in Table 4.5 for Soft-EM, Hard-EM, and EqHard-EM. As shown, Soft-EM suffers from the synchronous-training collapse, as all decoders essentially generate the same response, except for Decoder 9 whose response is different.

Input	世界名犬幼年期，萌死你不偿命！	World renowned little puppy, kill you with cuteness!
Soft-EM	1: 我家的狗狗也是这样的	My doggo is also like this
	2: 我家的狗狗也是这样的	My doggo is also like this
	3: 我家的狗狗也是这样	My doggo is also like this
	4: 我家的狗狗也是这样的	My doggo is also like this
	5: 我家的狗狗也是这样的	My doggo is also like this
	6: 我家的狗狗也是这样的	My doggo is also like this
	7: 我家的狗也是这样的	My dog is also like this
	8: 我家的狗也是这样的	My dog is also like this
	9: 我想养一只，可惜没有狗	I want one, too bad no dogs for me
	10: 我家的狗狗也是这样	My doggo is also like this
Hard-EM	1: 我家的狗狗也是这样的	My doggo is also like this
	2: 。！。。	My doggo is also like this
	3: 世界名犬幼年期	World renowned little puppy
	4: ！！！！！ [repeat]	!!!!! [repeat]
	5: 其实。！	In fact.!
	6: 太可爱了，太太太太太 [repeat]	Very cute, very very very very very [repeat]
	7: 。！。！	.!.!
	8: 世界名犬	World renowned dog
	9: 每次都要让我们生活充满着无限的欢乐	Let our lives be filled with unbounded joy every time
	10: 最后一张是什么？求解？	What is the last one? I wonder?
EqHard-EM	1: 好可爱啊，我也想养一只	Very cute, I want one too
	2: 我的妈呀，太可耐了	My dear, too cute
	3: 这是什么品种的狗？	What breed is this?
	4: 我家的狗狗也是小时候	My doggo was too when it was little
	5: 我想说，这是什么品种啊？	I want to say, what breed is this?
	6: 这个。我喜欢！太可爱了	This. I l love it! Very cute
	7: 好可爱啊，我也想养一只	Very cute, I want one too
	8: 我要养只萨摩耶！	I want a Samoyed!
	9: 我的妈呀，这是什么品种啊？	My dear, what breed is this?
	10: 我喜欢第一个，好可爱啊！	I like the first one, so cute!

Table 4.5: The outputs given by Soft-EM, Hard-EM, and EqHard-EM. In the table, [repeat] indicates that additional repetitive words are omitted.

This confirms that Soft-EM causes the multi-decoder model to essentially degenerate into a single-decoder model.

The responses for Hard-EM, on the other hand, demonstrate the non-training collapse. As seen, most decoders fail to generate a coherent response: Decoders 2, 4, and 7 only generate punctuation; Decoders 3 and 8 copy part of the input sentence; Decoder 6 gets stuck in a loop; and Decoder 9 generates a completely irrelevant response. Only Decoder 1 and Decoder 10 generate plausible responses. The failure of these decoders is expected because they are mostly untrained due to the non-training collapse of Hard-EM.

On the contrary, EqHard-EM is able to generate diverse responses when given the same input, essentially overcoming both synchronous and non-training collapse. As shown, all decoders generate unique responses, except for Decoder 1 and Decoder 7 whose responses are the same. In particular, even though Decoders 3, 5, and 9 all talk about the breed of the dog, they use different expressions. Notably, Decoder 8 mentions wanting a Samoyed as a pet, showing its ability to generate responses beyond the surface level understanding. Overall, our case study shows that the EqHard-EM algorithm allows decoders to specialize and generate diverse responses, avoiding both kinds of collapse shown by Soft-EM and Hard-EM.

4.8 Summary

In this chapter, we started by providing details for our evaluation datasets: Weibo and OpenSubtitles. Weibo features explicit one-to-many mapping, where the same input context corresponds to multiple responses in most cases. On the other hand, OpenSubtitles has implicit one-to-many mapping, where most input contexts only have one response. Together, the two datasets help us to comprehensively evaluate our model in both scenarios.

We then moved on to the evaluation metrics, featuring both quality and diversity measures. For response quality, we adopt the commonly used BLEU metric, which measures the lexical similarity between model responses and reference responses. In addition, we measure the performance upper bound of our multi-decoder model through the best-BLEU metric. For response diversity, we adopt the popular Dist metric, which measures the diversity of model responses across different input contexts. We further measure the similarity among decoders through the pairwise-wise BLEU metric. Overall, these metrics help us to carefully examine the quality and diversity of our model outputs.

Finally, we performed experiments on the two datasets and show that our multi-decoder model significantly outperforms baseline models and other EM variants. In addition, we conducted both quantitative and qualitative analyses on individual decoder outputs. Our analyses suggest that decoders trained by our EqHard-EM algorithm are able to specialize and generate diverse responses. Overall, our results show that our multi-decoder model and EqHard-EM algorithm alleviate the mode averaging problem and generate high-quality responses that are also diverse.

Chapter 5

Conclusion and Future Work

In this work, we address the generic response problem in open-domain dialogue systems: traditional encoder–decoder architectures tend to generate generic responses such as *–I don't know.* It is hypothesized that this phenomenon is caused by the one-to-many mapping in the dialogue task. That is, given a context, there are oftentimes many valid responses. Therefore, the goal can be thought of as learning a multi-modal target distribution, where different modes represent different kinds of responses. The standard encode–decoder model fails to capture different modes in the distribution, as cross-entropy training encourages the model to learn an overly smoothed function. As a result, the model has the mode averaging problem and resorts to generating generic responses.

To address this issue, we propose a multi-decoder architecture, which allows the model to capture different modes with different decoders, alleviating the mode averaging problem. We additionally propose to implement different decoders as different adapter modules that are inserted into a shared Transformer, based on the following considerations. First, our multi-adapter implementation is more parameter efficient than allocating full Transformer models, allowing us to scale up the number of decoders with limited GPU memory. Second, parameter sharing with adapter modules enables better knowledge transfer among different decoders.

To train our multi-decoder model, we propose to adopt EM-like algorithms by treating the choice of the decoder as an unobserved latent variable. However, we observe that both traditional Hard-EM and Soft-EM perform poorly in our multi-decoder training. In particular, Hard-EM assigns samples to their best-fit decoders, which causes the rich-gets-richer phenomenon: the best-performing decoder will be selected and therefore trained more, making other decoders even less likely to be selected. Empirically, we find that Hard-EM typically only trains one decoder, which degenerates the multi-decoder model to a single-decoder model. Alternatively, Soft-EM assigns samples in a soft manner, allowing all decoders to be trained. However, Soft-EM tends to train all decoders in a similar fashion due to its soft assignment, which causes the decoders to generate similar responses. As a result, the multi-decoder model also behaves like a single-decoder model in the Soft-EM case.

To this end, we propose Equal-Size Hard Expectation–Maximization (EqHard-EM), which addresses both issues in Soft-EM and Hard-EM. EqHard-EM adopts hard assignment, so it alleviates the problem in the Soft-EM case: decoders are trained with distinct samples, and the decoders are more likely to diverge from each other. In addition, we enforce an equal-assignment constraint, forcing every decoder to be trained with an equal number of samples. This further ensures that all decoders are well trained, unlike Hard-EM training. We solve the equal-assignment problem by reformulating it as a balanced assignment problem, which can be solved by the classic Hungarian algorithm.

Experimental results show that the multi-decoder model trained by our EqHard-EM algorithm significantly outperforms single-decoder models in both quality and diversity metrics. In addition, quantitative and qualitative analyses verify that EqHard-EM enables the decoders to specialize in different ways and generate diverse responses. Overall, our experiments show that the mode averaging problem in dialogue systems is indeed problematic and can be alleviated with well-trained multi-decoder models.

The thesis points to several future directions: multi-decoder initialization and decoder selection.

Decoder initialization is a key design choice that affects the overall performance. Currently, we initialize our model by fine-tuning a standard encoder–decoder model, and then adapter modules are inserted to form different decoders. However, it is likely that the model already suffers from the mode averaging problem before multi-decoder training. Future work may address this issue by diverse decoder initialization.

Another future direction is to explore decoder-selection methods. In our EqHard-EM algorithm, the decoder is selected based on the loss in training; however, the loss is not available during inference time. Therefore, we have to select a decoder based on the model confidence (predicted probability) during inference. This results in a discrepancy between training and inference, which is known as exposure bias. To address the issue, we may further explore the following ideas. First, we may learn a gating function to predict the loss, and use it for both training and inference. Alternatively, we may use model confidence for decoder selection during training. Both approaches address the exposure bias, and we hope they could lead to further improvement.

Bibliography

[1] "Abstractive text summarization using sequence-to-sequence RNNs and beyond". In: *Proceedings of The SIGNLL Conference on Computational Natural Language Learning*. 2016, pp. 280–290. DOI: 10.18653/v1/K16-1028.

[2] Abien Fred Agarap. "Deep learning using rectified linear units (ReLU)". In: *arXiv preprint arXiv:1803.08375* (2018). URL: https://arxiv.org/abs/1803.08375.

[3] Apoorv Agarwal, Boyi Xie, Ilia Vovsha, Owen Rambow, and Rebecca Passonneau. "Sentiment analysis of Twitter data". In: *Proceedings of the Workshop on Language in Social Media*. 2011, pp. 30–38. URL: https://aclanthology.org/W11-0705.

[4] Dzmitry Bahdanau, Kyung Hyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate". In: *International Conference on Learning Representations*. 2015. URL: https://arxiv.org/abs/1409.0473.

[5] Yu Bao, Hao Zhou, Shujian Huang, Lei Li, Lili Mou, Olga Vechtomova, Xin-yu Dai, and Jiajun Chen. "Generating sentences from disentangled syntactic and semantic spaces". In: *Proceedings of the Annual Meeting of the Association for Computational Linguistics*. 2019, pp. 6008–6019. DOI: 10.18653/v1/P19-1602.

[6] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[7] Antoine Bordes, Y-Lan Boureau, and Jason Weston. "Learning end-to-end goal-oriented dialog". In: *International Conference on Learning Representations*. 2017. URL: https://arxiv.org/abs/1605.07683.

[8] Nicolas Boulanger-Lewandowski, Yoshua Bengio, and Pascal Vincent. "Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription". In: *Proceedings of the International Conference on Machine Learning*. 2012, pp. 1159–1166. URL: https://icml.cc/Conferences/2012/papers/590.pdf.

[9] Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gašić. "MultiWOZ - A large-scale multi-domain Wizard-of-Oz dataset for task-oriented dialogue modelling". In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. 2018, pp. 5016–5026. DOI: 10.18653/v1/D18-1547.

[10] David Burkett and Dan Klein. “Two languages are better than one (for syntactic parsing)”. In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. 2008, pp. 877–886. URL: https://aclanthology.org/D08-1092.

[11] Hengyi Cai, Hongshen Chen, Yonghao Song, Zhuoye Ding, Yongjun Bao, Weipeng Yan, and Xiaofang Zhao. “Group-wise contrastive learning for neural dialogue generation”. In: *Findings of the Association for Computational Linguistics: Empirical Methods in Natural Language Processing*. 2020, pp. 793–802. DOI: 10.18653/v1/2020.findings-emnlp.70.

[12] Hengyi Cai, Hongshen Chen, Yonghao Song, Cheng Zhang, Xiaofang Zhao, and Dawei Yin. “Data manipulation: Towards effective instance learning for neural dialogue generation via learning to augment and reweight”. In: *Proceedings of the Annual Meeting of the Association for Computational Linguistics*. 2020, pp. 6334–6343. DOI: 10.18653/v1/2020.acl-main.564.

[13] Hongshen Chen, Xiaorui Liu, Dawei Yin, and Jiliang Tang. “A survey on dialogue systems: Recent advances and new frontiers”. In: *ACM Special Interest Group on Knowledge Discovery and Data Mining Explorations Newsletter* 19 (2017), pp. 25–35. DOI: 10.1145/3166054.3166058.

[14] Ke Chen, Weilin Zhang, Shlomo Dubnov, Gus Xia, and Wei Li. “The effect of explicit structure encoding of deep neural networks for symbolic music generation”. In: *International Workshop on Multilayer Music Representation and Processing*. 2019, pp. 77–84. DOI: 10.1109/MMRP.2019.00022.

[15] Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. “Learning phrase representations using RNN encoder–decoder for statistical machine translation”. In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. 2014, pp. 1724–1734. DOI: 10.3115/v1/D14-1179.

[16] Sumit Chopra, Michael Auli, and Alexander M. Rush. “Abstractive sentence summarization with attentive recurrent neural networks”. In: *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 2016, pp. 93–98. DOI: 10.18653/v1/N16-1012.

[17] Arthur P Dempster, Nan M Laird, and Donald B Rubin. “Maximum likelihood from incomplete data via the EM algorithm”. In: *Journal of the Royal Statistical Society: Series B (Methodological)* 39.1 (1977), pp. 1–22. DOI: 10.1111/j.2517-6161.1977.tb01600.x.

[18] Anup Anand Deshmukh, Qianqiu Zhang, Ming Li, Jimmy Lin, and Lili Mou. “Unsupervised chunking as syntactic structure induction with a knowledge-transfer approach”. In: *Findings of the Association for Computational Linguistics: Empirical Methods in Natural Language Processing*. 2021, pp. 3626–3634. DOI: 10.18653/v1/2021.findings-emnlp.307.

[19] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. “BERT: Pre-training of deep bidirectional transformers for language understanding”. In: *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies.* 2019, pp. 4171–4186. DOI: 10.18653/v1/N19-1423.

[20] Chengzhang Dong, Chenyang Huang, Osmar Zaiane, and Lili Mou. “Simulated annealing for emotional dialogue systems”. In: *Proceedings of the ACM International Conference on Information & Knowledge Management.* 2021, pp. 2984–2988. DOI: 10.1145/3459637.3482182.

[21] Elozino Egonmwan and Yllias Chali. “Transformer and Seq2Seq model for paraphrase generation”. In: *Proceedings of the Workshop on Neural Generation and Translation.* Association for Computational Linguistics, 2019, pp. 249–255. DOI: 10.18653/v1/D19-5627.

[22] Jun Gao, Wei Bi, Xiaojiang Liu, Junhui Li, and Shuming Shi. “Generating multiple diverse responses for short-text conversation”. In: *Proceedings of the AAAI Conference on Artificial Intelligence.* 2019, pp. 6383–6390. DOI: 10.1609/aaai.v33i01.33016383.

[23] Wenliang Gao, Jingxiang Gao, Liu Yang, Mingjun Wang, and Wenhao Yao. “A novel modeling strategy of weighted mean temperature in China using RNN and LSTM”. In: *Remote Sensing* 13.15 (2021), p. 3004. DOI: 10.3390/rs13153004.

[24] D. Goddeau, H. Meng, J. Polifroni, S. Seneff, and S. Busayapongchai. “A form-based dialogue manager for spoken language applications”. In: *Proceeding of the International Conference on Spoken Language Processing.* 1996, pp. 701–704. DOI: 10.1109/ICSLP.1996.607458.

[25] Xiaodong Gu, Kyunghyun Cho, Jung-Woo Ha, and Sunghun Kim. “Dialog-WAE: Multimodal response generation with conditional Wasserstein auto-encoder”. In: *International Conference on Learning Representations.* 2019. URL: https://openreview.net/forum?id=BkgBvsC9FQ.

[26] Xiaodong Gu, Kang Min Yoo, and Jung-Woo Ha. “DialogBERT: Discourse-aware response generation via learning to recover and rank utterances”. In: *Proceedings of the AAAI Conference on Artificial Intelligence.* 2021, pp. 12911–12919. URL: https://ojs.aaai.org/index.php/AAAI/article/view/17527.

[27] Ankush Gupta, Arvind Agarwal, Prawaan Singh, and Piyush Rai. “A deep generative framework for paraphrase generation”. In: *Proceedings of the AAAI Conference on Artificial Intelligence* (2018), pp. 5149–5156. DOI: 10.1609/aaai.v32i1.11956. URL: https://ojs.aaai.org/index.php/AAAI/article/view/11956.

[28] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. “Deep residual learning for image recognition”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.* 2016, pp. 770–778. URL: https://openaccess.thecvf.com/content_cvpr_2016/papers/He_Deep_Residual_Learning_CVPR_2016_paper.pdf.

[29] Matthew Henderson, Blaise Thomson, and Jason D. Williams. "The second dialog state tracking challenge". In: *Proceedings of the Annual Meeting of the Special Interest Group on Discourse and Dialogue*. 2014, pp. 263–272. DOI: 10.3115/v1/W14-4337.

[30] Matthew Henderson, Blaise Thomson, and Jason D. Williams. "The third dialog state tracking challenge". In: *IEEE Spoken Language Technology Workshop*. 2014, pp. 324–329. DOI: 10.1109/SLT.2014.7078595.

[31] Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. "The Goldilocks principle: Reading children's books with explicit memory representations". In: *International Conference on Learning Representation* (2016). URL: https://arxiv.org/abs/1511.02301.

[32] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory". In: *Neural Computation* 9.8 (1997), pp. 1735–1780. DOI: 10.1162/neco.1997.9.8.1735.

[33] Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. "The curious case of neural text degeneration". In: *International Conference on Learning Representations*. 2020. URL: https://openreview.net/pdf?id=rygGQyrFvH.

[34] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. "Parameter-efficient transfer learning for NLP". In: *Proceedings of the International Conference on Machine Learning*. 2019, pp. 2790–2799. URL: http://proceedings.mlr.press/v97/houlsby19a.html.

[35] Chenyang Huang, Hao Zhou, Osmar R. Zaïane, Lili Mou, and Lei Li. "Non-autoregressive translation with layer-wise prediction and deep supervision". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. 2022, pp. 10776–10784. DOI: 10.1609/aaai.v36i10.21323.

[36] Minlie Huang, Xiaoyan Zhu, and Jianfeng Gao. "Challenges in building intelligent open-domain dialog systems". In: *ACM Transactions on Information Systems* 38.3 (2020). DOI: 10.1145/3383123.

[37] K. Kamijo and T. Tanigawa. "Stock price pattern recognition-A recurrent neural network approach". In: *Proceedings of the International Joint Conference on Neural Networks*. 1990, pp. 215–221. DOI: 10.1109/IJCNN.1990.137572.

[38] Kashif Khan, Gaurav Sahu, Vikash Balasubramanian, Lili Mou, and Olga Vechtomova. "Adversarial learning on the latent space for diverse dialog generation". In: *Proceedings of the International Conference on Computational Linguistics*. 2020, pp. 5026–5034. DOI: 10.18653/v1/2020.coling-main.441.

[39] Harold W Kuhn. "The Hungarian method for the assignment problem". In: *Naval Research Logistics Quarterly* 2.1-2 (1955), pp. 83–97. DOI: 10.1002/nav.3800020109.

[40] Souvik Kundu, Qian Lin, and Hwee Tou Ng. "Learning to identify follow-up questions in conversational question answering". In: *Proceedings of the Annual Meeting of the Association for Computational Linguistics.* 2020, pp. 959–968. DOI: 10.18653/v1/2020.acl-main.90.

[41] Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. "RACE: Large-scale reading comprehension dataset from examinations". In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing.* 2017, pp. 785–794. DOI: 10.18653/v1/D17-1082.

[42] Staffan Larsson and David R. Traum. "Information state and dialogue management in the TRINDI dialogue move engine toolkit". In: *Natural Language Engineering* 6 (2000), pp. 323–340. DOI: 10.1017/S1351324900002539.

[43] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. "BART: Denoising Sequence-to-Sequence pre-training for natural language generation, translation, and comprehension". In: *Proceedings of the Annual Meeting of the Association for Computational Linguistics.* 2020, pp. 7871–7880. DOI: 10.18653/v1/2020.acl-main.703.

[44] Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. "A diversity-promoting objective function for neural conversation models". In: *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies.* 2016, pp. 110–119. DOI: 10.18653/v1/N16-1014.

[45] Jiwei Li, Will Monroe, Alan Ritter, Dan Jurafsky, Michel Galley, and Jianfeng Gao. "Deep reinforcement learning for dialogue generation". In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing.* 2016, pp. 1192–1202. DOI: 10.18653/v1/D16-1127.

[46] Jiwei Li, Will Monroe, Alan Ritter, Dan Jurafsky, Michel Galley, and Jianfeng Gao. "Deep reinforcement learning for dialogue generation". In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing.* 2016, pp. 1192–1202. DOI: 10.18653/v1/D16-1127.

[47] Jiwei Li, Will Monroe, Tianlin Shi, Sébastien Jean, Alan Ritter, and Dan Jurafsky. "Adversarial learning for neural dialogue generation". In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing.* 2017, pp. 2157–2169. DOI: 10.18653/v1/D17-1230.

[48] Zichao Li, Xin Jiang, Lifeng Shang, and Hang Li. "Paraphrase generation with deep reinforcement learning". In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing.* 2018, pp. 3865–3878. DOI: 10.18653/v1/D18-1421.

[49] Pierre Lison, Jörg Tiedemann, and Milen Kouylekov. "OpenSubtitles2018: Statistical rescoring of sentence alignments in large, noisy parallel corpora". In: *Proceedings of the Eleventh International Conference on Language Resources and Evaluation.* 2018. URL: https://aclanthology.org/L18-1275.pdf.

[50] Puyuan Liu, Chenyang Huang, and Lili Mou. “Learning non-autoregressive models from search for unsupervised sentence summarization”. In: *Proceedings of the Annual Meeting of the Association for Computational Linguistics*. 2022, pp. 7916–7929. DOI: 10.18653/v1/2022.acl-long.545.

[51] Yeqi Liu, Chuanyang Gong, Ling Yang, and Yingyi Chen. “DSTP-RNN: A dual-stage two-phase attention-based recurrent neural network for long-term and multivariate time series prediction”. In: *Expert Systems with Applications* 143 (2020), p. 113082. DOI: https://doi.org/10.1016/j.eswa.2019.113082.

[52] Samuel Louvan and Bernardo Magnini. “Recent neural methods on slot filling and intent classification for task-oriented dialogue systems: A survey”. In: *Proceedings of the International Conference on Computational Linguistics*. 2020, pp. 480–496. DOI: 10.18653/v1/2020.coling-main.42.

[53] Mantas Lukoševičius and Herbert Jaeger. “Reservoir computing approaches to recurrent neural network training”. In: *Computer Science Review* 3 (2009), pp. 127–149. DOI: https://doi.org/10.1016/j.cosrev.2009.03.005.

[54] Qi Lyu, Zhiyong Wu, Jun Zhu, and Helen Meng. “Modelling high-dimensional sequences with LSTM-RTRBM: Application to polyphonic music generation”. In: *Proceedings of the International Joint Conference on Artificial Intelligence*. 2015, pp. 4138–4139. URL: https://www.ijcai.org/Abstract/15/582.

[55] Mausam, Michael Schmitz, Stephen Soderland, Robert Bart, and Oren Etzioni. “Open language learning for information extraction”. In: *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. 2012, pp. 523–534. URL: https://aclanthology.org/D12-1048.

[56] Walaa Medhat, Ahmed Hassan, and Hoda Korashy. “Sentiment analysis algorithms and applications: A survey”. In: *Ain Shams Engineering Journal* 5.4 (2014), pp. 1093–1113. DOI: https://doi.org/10.1016/j.asej.2014.04.011.

[57] Ning Miao, Hao Zhou, Lili Mou, Rui Yan, and Lei Li. “CGMH: Constrained sentence generation by Metropolis-Hastings sampling”. In: *Proceedings of the AAAI Conference on Artificial Intelligence* (2019), pp. 6834–6842. DOI: 10.1609/aaai.v33i01.33016834. URL: https://ojs.aaai.org/index.php/AAAI/article/view/4659.

[58] Un Yong Nahm and Raymond J Mooney. “Text mining with information extraction”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. 2002, pp. 60–67. URL: https://www.aaai.org/Papers/Symposia/Spring/2002/SS-02-06/SS02-06-013.pdf.

[59] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. “BLEU: A method for automatic evaluation of machine translation”. In: *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*. 2002, pp. 311–318. DOI: 10.3115/1073083.1073135.

[60] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. "On the difficulty of training recurrent neural networks". In: *Proceedings of the International Conference on Machine Learning*. 2013, pp. 1310–1318. URL: https://proceedings.mlr.press/v28/pascanu13.html.

[61] Baolin Peng, Xiujun Li, Lihong Li, Jianfeng Gao, Asli Celikyilmaz, Sungjin Lee, and Kam-Fai Wong. "Composite task-completion dialogue policy learning via hierarchical deep reinforcement learning". In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. 2017. DOI: 10.18653/v1/D17-1237.

[62] Arthur Petrosian, Danil Prokhorov, Richard Homan, Richard Dasheiff, and Donald Wunsch. "Recurrent neural network based prediction of epileptic seizures in intra- and extracranial EEG". In: *Neurocomputing* 30 (2000), pp. 201–218. DOI: 10.1016/S0925-2312(99)00126-5.

[63] Olivier Pietquin, Matthieu Geist, Senthilkumar Chandramohan, and Hervé Frezza-Buet. "Sample-efficient batch reinforcement learning for dialogue management optimization". In: *ACM Transactions on Speech and Language Processing* 7 (2011). DOI: 10.1145/1966407.1966412.

[64] Chen Qu, Liu Yang, Minghui Qiu, W. Bruce Croft, Yongfeng Zhang, and Mohit Iyyer. "BERT with history answer embedding for conversational question answering". In: *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2019, pp. 1133–1136. DOI: 10.1145/3331184.3331341.

[65] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. "Language models are unsupervised multitask learners". In: *OpenAI Blog* (2019). URL: https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf.

[66] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, Peter J Liu, et al. "Exploring the limits of transfer learning with a unified text-to-text transformer." In: *Journal of Machine Learning Research* 21.140 (2020), pp. 1–67. URL: https://jmlr.org/papers/v21/20-074.html.

[67] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. "SQuAD: 100,000+ questions for machine comprehension of text". In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. 2016, pp. 2383–2392. DOI: 10.18653/v1/D16-1264.

[68] Siva Reddy, Danqi Chen, and Christopher D. Manning. "CoQA: A conversational question answering challenge". In: *Transactions of the Association for Computational Linguistics* 7 (2019), pp. 249–266. DOI: 10.1162/tacl_a_00266.

[69] Alexander Schrijver. *Theory of Linear and Integer Programming*. John Wiley & Sons, 1998.

[70] Tianxiao Shen, Myle Ott, Michael Auli, and Marc'Aurelio Ranzato. "Mixture models for diverse machine translation: Tricks of the trade". In: *Proceedings of the International Conference on Machine Learning*. 2019, pp. 5719–5728. URL: http://proceedings.mlr.press/v97/shen19c.html.

[71] Satinder Singh, Diane Litman, Michael Kearns, and Marilyn Walker. "Optimizing dialogue management with reinforcement learning: Experiments with the NJFun system". In: *Journal of Artificial Intelligence Research* 16 (2002), pp. 105–133. DOI: 10.1613/jair.859.

[72] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. "Sequence to sequence learning with neural networks". In: *Advances in Neural Information Processing Systems*. 2014.

[73] Chongyang Tao, Lili Mou, Dongyan Zhao, and Rui Yan. "RUBER: An unsupervised method for automatic evaluation of open-domain dialog systems". In: *Proceedings of the AAAI Conference on Artificial Intelligence* (2018). DOI: 10.1609/aaai.v32i1.11321.

[74] Van-Khanh Tran and Le-Minh Nguyen. "Semantic refinement GRU-based neural language generation for spoken dialogue systems". In: *Computational Linguistics*. 2018, pp. 63–75. DOI: 10.1007/978-981-10-8438-6_6.

[75] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. "Attention is all you need". In: *Advances in Neural Information Processing Systems* (2017). URL: https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

[76] Ashwin K Vijayakumar, Michael Cogswell, Ramprasath R Selvaraju, Qing Sun, Stefan Lee, David Crandall, and Dhruv Batra. "Diverse beam search: Decoding diverse solutions from neural sequence models". In: *arXiv preprint arXiv:1610.02424* (2016). URL: https://arxiv.org/abs/1610.02424.

[77] Yida Wang, Yinhe Zheng, Yong Jiang, and Minlie Huang. "Diversifying dialog generation via adaptive label smoothing". In: *Proceedings of the Annual Meeting of the Association for Computational Linguistics and the International Joint Conference on Natural Language Processing*. 2021, pp. 3507–3520. DOI: 10.18653/v1/2021.acl-long.272.

[78] Yu Wang, Yilin Shen, and Hongxia Jin. "A bi-model based RNN semantic frame parsing model for intent detection and slot filling". In: *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 2018, pp. 309–314. DOI: 10.18653/v1/N18-2050.

[79] Bolin Wei, Shuai Lu, Lili Mou, Hao Zhou, Pascal Poupart, Ge Li, and Zhi Jin. "Why do neural dialog systems generate short and meaningless replies? A comparison between dialog and translation". In: *Proceedings of the 2019 IEEE International Conference on Acoustics, Speech and Signal Processing*. 2019, pp. 7290–7294. DOI: 10.1109/ICASSP.2019.8682634.

[80] Zhongyu Wei, Qianlong Liu, Baolin Peng, Huaixiao Tou, Ting Chen, Xuanjing Huang, Kam-fai Wong, and Xiangying Dai. "Task-oriented dialogue system for automatic diagnosis". In: *Proceedings of the Annual Meeting of the Association for Computational Linguistics.* 2018, pp. 201–207. DOI: 10.18653/v1/P18-2033.

[81] Tsung-Hsien Wen, David Vandyke, Nikola Mrkšić, Milica Gašić, Lina M. Rojas-Barahona, Pei-Hao Su, Stefan Ultes, and Steve Young. "A network-bbsed end-to-end trainable task-oriented dialogue system". In: *Proceedings of the European Chapter of the Association for Computational Linguistics.* 2017, pp. 438–449. URL: https://aclanthology.org/E17-1042.

[82] Yuqiao Wen, Guoqing Luo, and Lili Mou. "An empirical study on the overlapping problem of open-domain dialogue datasets". In: *Proceedings of the 13th Conference on Language Resources and Evaluation.* 2022, pp. 146–153. URL: http://www.lrec-conf.org/proceedings/lrec2022/pdf/2022.lrec-1.16.pdf.

[83] Paul Werbos. "Backpropagation through time: What it does and how to do it". In: *Proceedings of the IEEE.* 10. 1990, pp. 1550–1560. DOI: 10.1109/5.58337.

[84] Jason Williams, Antoine Raux, Deepak Ramachandran, and Alan Black. "The dialog state tracking challenge". In: *Proceedings of the Special Interest Group on Discourse and Dialogue.* 2013, pp. 404–413. URL: https://aclanthology.org/W13-4065.

[85] Guangfeng Yan, Lu Fan, Qimai Li, Han Liu, Xiaotong Zhang, Xiao-Ming Wu, and Albert Y.S. Lam. "Unknown intent detection using Gaussian mixture model with an application to zero-shot intent classification". In: *Proceedings of the Annual Meeting of the Association for Computational Linguistics.* 2020, pp. 1050–1060. DOI: 10.18653/v1/2020.acl-main.99.

[86] Yuting Yang, Junyu Dong, Xin Sun, Estanislau Lima, Quanquan Mu, and Xinhua Wang. "A CFCC-LSTM model for sea surface temperature preeeiction". In: *IEEE Geoscience and Remote Sensing Letters* 15.2 (2018), pp. 207–211. DOI: 10.1109/LGRS.2017.2780843.

[87] Steve Young, Milica Gašić, Simon Keizer, François Mairesse, Jost Schatzmann, Blaise Thomson, and Kai Yu. "The hidden information state model: A practical framework for POMDP-based spoken dialogue management". In: *Computer Speech & Language* 24.2 (2010), pp. 150–174. DOI: https://doi.org/10.1016/j.csl.2009.04.001.

[88] Cha Zhang and Yunqian Ma. *Ensemble Machine Learning: Methods and Applications.* Springer, 2012.

[89] Lin Zhao and Zhe Feng. "Improving slot filling in spoken language understanding with joint pointer and attention". In: *Proceedings of the Annual Meeting of the Association for Computational Linguistics.* 2018, pp. 426–431. DOI: 10.18653/v1/P18-2068.

[90] Tiancheng Zhao, Ran Zhao, and Maxine Eskenazi. "Learning discourse-level diversity for neural dialog models using conditional variational autoencoders". In: *Proceedings of the Annual Meeting of the Association for Computational Linguistics*. 2017, pp. 654–664. DOI: 10.18653/v1/P17-1061.

[91] Wangchunshu Zhou, Qifei Li, and Chenle Li. "Learning from perturbations: Diverse and informative dialogue generation with inverse adversarial training". In: *Proceedings of the Annual Meeting of the Association for Computational Linguistics and the International Joint Conference on Natural Language Processing*. 2021, pp. 694–703. DOI: 10.18653/v1/2021.acl-long.57.

[92] Qi Zhu, Kaili Huang, Zheng Zhang, Xiaoyan Zhu, and Minlie Huang. "CrossWOZ: A large-scale Chinese cross-domain task-oriented dialogue dataset". In: *Transactions of the Association for Computational Linguistics* 8 (2020), pp. 281–295. DOI: 10.1162/tacl_a_00314.