Self-Training for Natural Language Processing

by

Hongyin Luo

B.E., Tsinghua University (2016)
S.M., Massachusetts Institute of Technology (2019)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 2022

Author . . . .
Department of Electrical Engineering and Computer Science
May 13, 2022

Certified by . . . .
James R. Glass
Senior Research Scientist
Computer Science and Artificial Intelligence Laboratory
Thesis Supervisor

Accepted by . . . .
Leslie A. Kolodziejski
Professor of Electrical Engineering and Computer Science
Chair, Department Committee on Graduate Students

Self-Training for Natural Language Processing

by

Hongyin Luo

Submitted to the Department of Electrical Engineering and Computer Science
on May 13, 2022, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

Data annotation is critical for machine learning based natural language processing models. Although many large-scale corpora and standard benchmarks have been annotated and published, they cannot cover all possible applications. As a result, it is difficult to transfer models trained with public corpora to tasks that require domain-specific knowledge, different inference skills, unseen text styles, and explainability. In this thesis, we explore self-training methods for mitigating the data distribution gaps between training and evaluation domains and tasks. In contrast to traditional self-training methods that study the best practice of training models with real data and pseudo labels, we also explore the possibility of automatically generating synthetic data for better explainability, robustness, and domain adaptation performance. We show the performance improvement achieved by our methods on different natural language understanding and generation tasks, including question answering, question generation, and dialog response selection.

Thesis Supervisor: James R. Glass
Title: Senior Research Scientist
Computer Science and Artificial Intelligence Laboratory

Acknowledgments

I worked as a TA for three semesters since 2019, because I felt bad about taking RA supports without producing impactful research. At that point, I wanted to escape from the Ph.D. journey as soon as possible.

However, together with my advisor, Dr. James Glass, lab members of SLS, my parents, and friends, I managed to put together this Ph.D. thesis. The communications became more difficult because of the pandemic, but there have been constant supports that encouraged me to focus on my own research. After years of exploring, we final found a topic to pursue. The wondering and painful experience is the greatest gift I received during the journey. I want to thank everyone and everything that encouraged me not to give up thinking and trying.

I would like to thank Jim for always being nice and supportive. I was encouraged to explore any interesting directions, try different ideas, and fail. Every time I got frustrated, I managed to restart my exploration because of Jim's support and patience. Thanks to Jim, I did not give up, and the experience of keeping trying and failing has become the most important experience and I have learned a lot from it. I am also grateful to receive valuable suggestions from my committee members, professor Peter Szolovitz and Yoon Kim. Your suggestions make the thesis much better than I originally imagined.

I would like to thank Professor Regina Barzilay, who admitted me to MIT. However, I did not join her group because of a misunderstanding about the admission. I believe this is one of the chocolates that I have to take in my life, and I'm also grateful about being recognized and admitted by Regina. I also appreciate the kind support and help from the EECS graduate office, especially Prof Leslie Kolodziejski, Ms. Janet E. Fisher, and Ms. Alicia Duarte.

I once felt difficult to communicate with my parents when I was an undergraduate student, but I finally found that their understanding and support are very important for me to keep trying and finally complete the journey. I want to sincerely thank my parents for their unconditional love and support.

I would like to thank Mr. Ming Liao and Prof. Xinwang Zhou, who encouraged me to apply for MIT when I was hesitating. It was a big decision that made a significant difference in my life. I deeply appreciate their encouragement and support.

I want to thank Shang-Wen Li, Shuyan Dong, Mengyang Yuan, Yuchen Wang, Meitong Li, Mingye Gao, Zhantao Chen, Mantian Xue, Felix Wong, Mo Deng, Tao Feng, Di Jin, Wenjie Yao, and Yuan Gong for being the greatest friends and advisors. Jun Wan, Kaixuan Yao, Xinliang Zhong, Shaoying Tan, Shuyun Xiao, Yuheng Zhong, Ran Yan, Jiachuan Zhang, Yuanxi Wang, Yu Xia, Xiang Li, Xin Qian, Ruogu Gao, Xu Liu, and Jiarui Xu, getting through the pandemic and finishing the thesis became easier because of your online supports.

Alpha and Blizzard, living with the troubles you have caused helps me calm down about the thesis and COVID. Your novel ways of destroying my home have always been inspiring. Thank you and please be good pets!

This research was supported in part by funds from FinTech@CSAIL initiative and by a grant from the Hong Kong Innovation AI program.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

One of the ultimate goals of artificial intelligence (AI) research is building systems that can interact with humans using natural language. Natural language processing (NLP) is the field focusing on this goal, and it consists of two significant research areas, natural language understanding (NLU) and natural language generation (NLG). Traditionally, both NLU and NLG mainly rely on large-scale human-annotated corpora for model development. Although it is possible to get annotations for popular languages and tasks, large-scale human annotation is prohibitively time-consuming and costly, especially if domain knowledge is needed. Furthermore, it is challenging to conduct real-time labeling with throughput sufficient for building mainstream NLP models on streaming data and use cases. These problems hinder the democratization of NLP research to more applications.

The problems and difficulties associated with getting humans to annotate data for NLP tasks impacts the performance for machine learning models since vast quantities of potentially useful data are unable to be leveraged due to lack of annotation. In real-world applications, human-annotated data is often lacking in one or more of the following aspects:

- Domain knowledge - It is normal that models are applied to process texts containing knowledge about specific domains that requires additional knowledge to understand, for example, finance, medicine, etc. Since new knowledge and concepts are created everyday, it is not possible for human annotators to cover and follow all available domain knowledge.

- Difficulty of reasoning - Some complex texts are difficult to understand and generate in different ways, for example, text style, implications, language conventions, and noise/typos, etc. It is impossible for human annotators to cover all possibilities since the number of possible natural text is infinite.
- Explainability - Most annotated benchmarks only contain input-label pairs, but do not include the reason for making a prediction. As a result, a machine learning model has to learn the input-lable mappings in a black-box style. However in practice, it is dangerous to trust model predictions without explanations in many areas.

To address the annotation bottleneck, previous studies have proposed different methods for few-shot learning [Finn et al., 2017] and transfer learning [Bengio, 2012]. Few-shot learning stands for the situation where only a few training cases are available, while transfer learning aims at generalizing a model pretrained on a source training set to a target task. Under both settings, there is a gap between the source training data and the target task caused by the difference of domain, difficulty, language style, etc. In other words, the target task has a different data distribution. Existing solutions for the problem are:
Model agnostic meta-learning (MAML) [Finn et al., 2017] introduces a second-order optimization on the source training data to learn a good initialization parameter setting for adapting to the new task by learning a small training set of the target task. This method requires an annotated training set including different tasks or domains, and learns to minimize the transferring difficulty among different training tasks. Although achieving impressive improvement on few-shot learning, the disadvantage of this method is obvious. Firstly, it needs a fine-grained training set annotation, and secondly, the computation of second-order gradients is very expensive, making it difficult to conduct this method on modern large-scale neural network-based language models.

Large-scale pretrained language models (PLM) was originally proposed by [Peters et al., 2018b] for contextualized word representation learning, but later studies found that training large-scale transformer [Vaswani et al., 2017] language models on large corpora can improve the finetuning performance on downstream NLP tasks [Devlin et al., 2018, Brown et al., 2020b]. The PLMs are pretrained with self-supervised learning on large corpora, for example, the entire Wikipedia corpus, to learn as much knowledge as possible. This pretraining strategy leads to fast and stable finetuning of downstream tasks. Pretraining also mitigates the domain and language gaps between the training and target datasets, since a wide range of vocabulary and sentences have been learned by the model. In this chapter, we will introduce the development of different neural language models, and then present the target and scope of this thesis.

1.1 Thesis Scope

In this thesis, we explore self-training methods for different natural language processing tasks. We first focus on learning soft label presentations in dialog response and action selection tasks to improve both accuracy and interpretability. In Chapter 3, we propose an interpretable dialog response selection by learning an evidence extraction model with pseudo-labels generated by a retrieval model. In Chapter 4, we propose a few-shot dialog action selection model for automatic diagnosis by learning prototypical dialog action embeddings in a self-supervised manner. Experiments show that learning with soft labels can improve the performance of both dialog-related tasks.

To further explore the potential of self-training models, we propose models for zero-shot domain adaptation for question answering (QA) models. In chapter 5, we proposed a self-supervised prompt pretraining method that learns domain knowledge before prompt tuning QA models. In Chapter 6, we explore directly training QA models on synthetic textual data generated by a cooperative question generation and answering pipeline, RGX.

For question generation (QG), we propose a sequence-to-sequence generation model

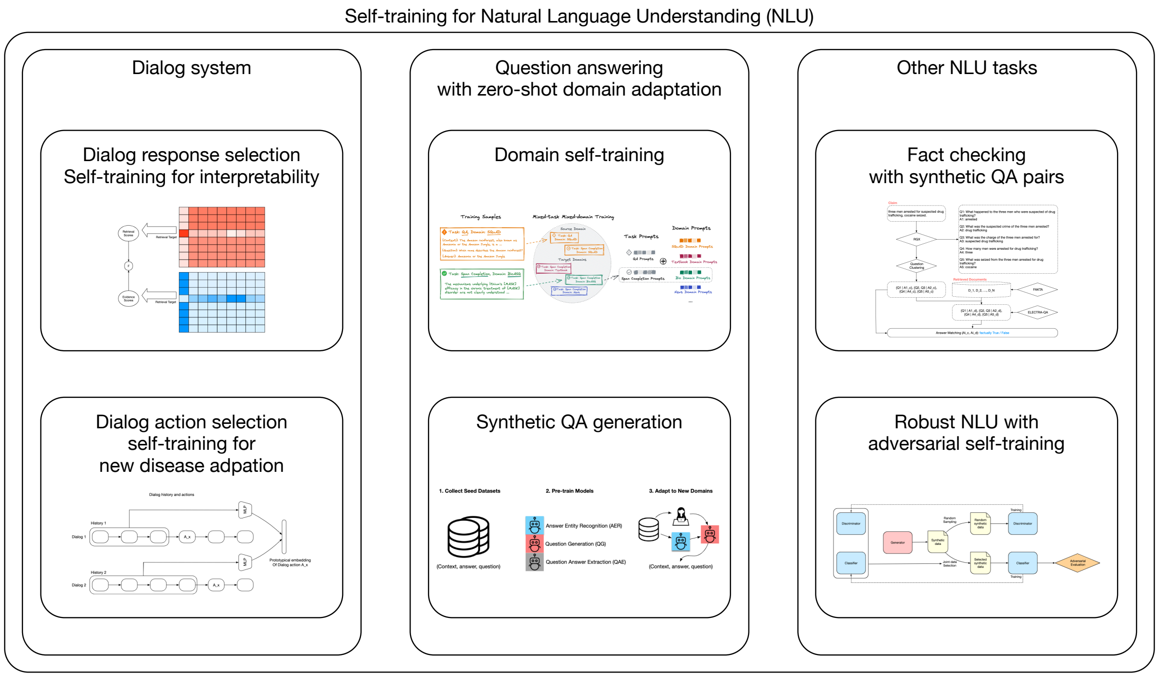


Figure 1-1: Overview of the structure of this thesis.

based on an answer-entity recognition. The question generation model can be further improved by using the pseudo labels generated by pretrained question answering models, with an appropriate synthetic data selection method.

For question answering (QA), we extend previous studies that train QA models with synthetic QA pairs by jointly tuning QA and QG models in a cooperative self-training pipeline, which encourages the QG model to generate non-trivial questions to improve the QA training. With the jointly learned QA and QG models, we build a system that automatically generates diverse and accurate synthetic QA pairs.

In Chapter 7, we extend the self-training method to other natural language understanding tasks, including fact checking and language entailment. We propose a QA-based fact checking pipeline, which achieved better performance under unsupervised settings comparing to supervised end-to-end models. We propose an adversarial self training method for the language entailment tasks, and present the experiment results on both regular and adversarial language understanding benchmarks.

The overview of the thesis is shown in Figure 1-1.

Chapter 2

Background

2.1 Neural Language Models

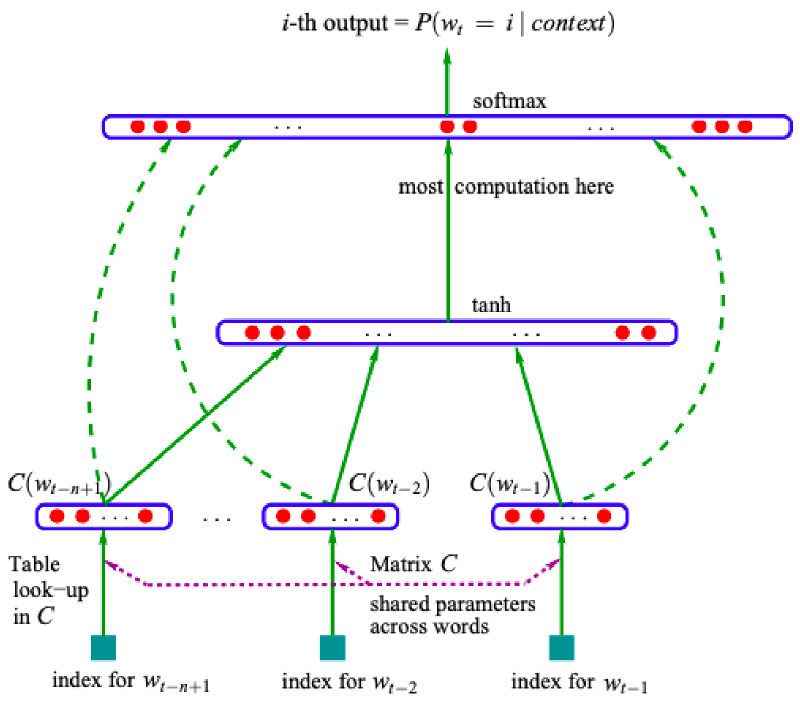


Figure 2-1: Architecture of a MLP-based language model for next-word prediction [Bengio et al., 2003].

Language modeling is a task that requires models to estimate the distribution of words given surrounding context. To improve the generalization ability of traditional N-gram language models, [Bengio et al., 2000] proposed a neural network-based language model that predicts the next word based on given contexts using multi-layer perceptron (MLP). The architecture of the model is shown in Figure 2-1. By learning to estimate the probability

$$P(w_t|w_0, w_1, \ldots, w_{t-1}) \tag{2.1}$$

the neural network gains semantic and syntactic knowledge about the language it processes. Although applying different neural network architectures, later language models, including the latest ones, are trained with similar methods that models the probability of a word given its context. [Mikolov et al., 2013a] and [Pennington et al., 2014] proposed models that learn distributed word embeddings by maximizing the mutual information between words with their surrounding contexts with linear models.

To improve the performance of the language models on different tasks, more complicated neural networks are applied in language modeling, including long short-term memory (LSTM) [Hochreiter and Schmidhuber, 1997] and transformer [Vaswani et al., 2017] networks. [Merity et al., 2017] proposed AWD-LSTM, concluding that recurrent neural network (RNN) based language models can achieve low generation perplexity after careful regularization. [Dai et al., 2019] proposed Transformer-XL, which improves language perplexity by processing longer contexts. Both LSTM and transformer models are also used for different language understanding and generation tasks, including coreference resolution [Lee et al., 2017], question answering [Rajpurkar et al., 2016], machine translation [Bahdanau et al., 2014], text retrieval and dialog response selection [Humeau et al., 2019a].

Although achieving significant improvement on different NLP tasks with complicated neural language models, there is an obvious gap between the language modeling task itself and other downstream NLP tasks, including language understanding and generation. For downstream tasks, the models are trained on annotated training sets. However, language modeling does not require human annotation since the training dataset labels itself - the inputs are contexts, and the label is the next word. As a result of this difference, training models for downstream NLP tasks heavily relies on human

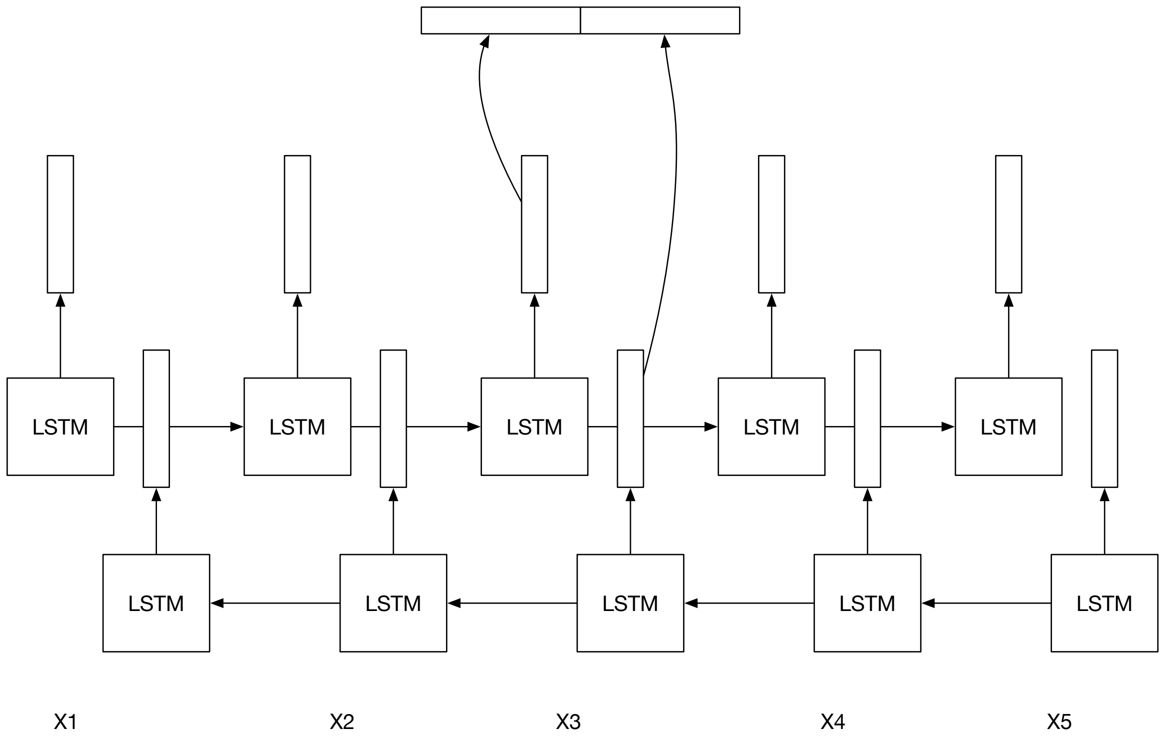


Figure 2-2: Architecture of the ELMO model [Peters et al., 2018a], which generates contextualized word embeddings by combining the hidden states output by both LSTM networks.

annotation, but language model training can be conducted on any publicly available texts with self-supervised learning. To utilize this advantage of language model learning, [Peters et al., 2018a] proposed a contextualized word embedding method, where a bi-directional LSTM is pretrained on large corpora for a bi-directional language modeling task. For other NLP tasks, the pretrained LSTM can be finetuned as an encoder of input texts. Compared with a randomly initialized neural network, the pretrained LSTM provides a high-quality prior for better training generalization since it has already learned to process a large amount of text data and gains richer semantic and syntactic knowledge. The architecture of the proposed model, ELMO, is shown in Figure 2-2.

A similar idea is also implemented with the stronger transformer model. The authors of [Devlin et al., 2018] proposed a masked language model task for language model pretraining. For a piece of given context, a transformer model is trained to recover randomly masked or replaced tokens. Note that the masked language model is similar to the continous bag-of-words model proposed in [Mikolov et al., 2013b], but instead of training a linear maximum mutual information model, BERT learns word-context knowledge with a deep transformer network. To better represent sequence-level information for sequence-level tasks, BERT is pretrained to predict the next sentences in addition to masked word prediction. Like ELMO, BERT can also be finetuned for downstream NLP tasks. The fact that transformer models do not have recurrent architecture leads to two benefits. Firstly, the transformer model does not need a bidirectional architecture since it processes sequences as a batch of ordered words, and secondly, transformer models are easier to train so a larger numbers of layers and trainable parameters can be utilized. As a result, the BERT model proposed in this study significantly outperformed the LSTM-based ELMO model. An example of the masked language modeling task is shown in Figure 2-3.

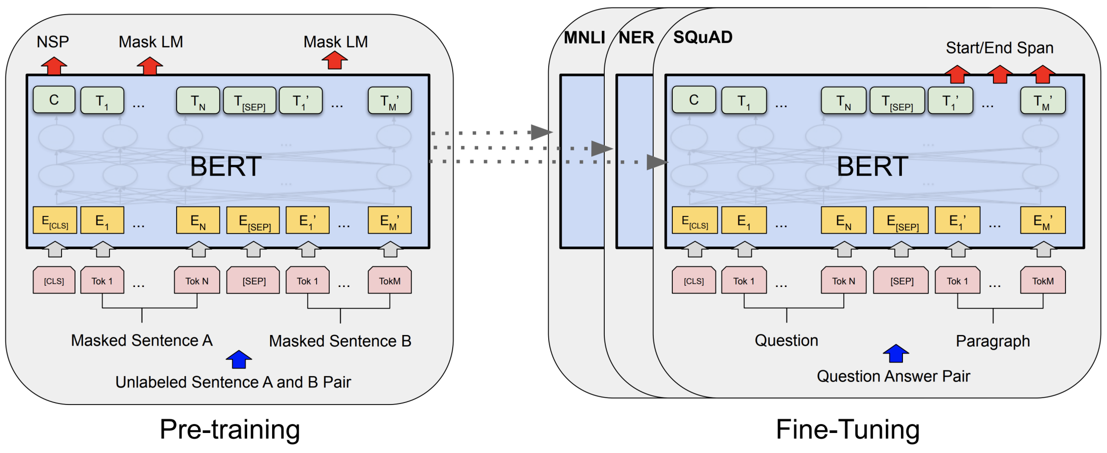


Figure 2-3: The training and finetuning illustration of the BERT model. The image is created by the authors of [Devlin et al., 2018].

While ELMO and BERT models are mainly applied as the encoder for language encoding and understanding, neural language models are also pretrained on large corpora for language generation tasks. GPT-2 [Brown et al., 2020b] is a pretrained left-to-right generative language model, while BART [Lewis et al., 2019a] and T5 [Raffel et al., 2019] employ a sequence-to-sequence architecture. These models can be finetuned to solve language generation tasks, for example machine translation, text summarization, and narrative question answering, while they can also solve language understanding tasks in a generative manner. The architecture of left-to-right and sequence-to-sequence language generators are shown in Figure 2-4.

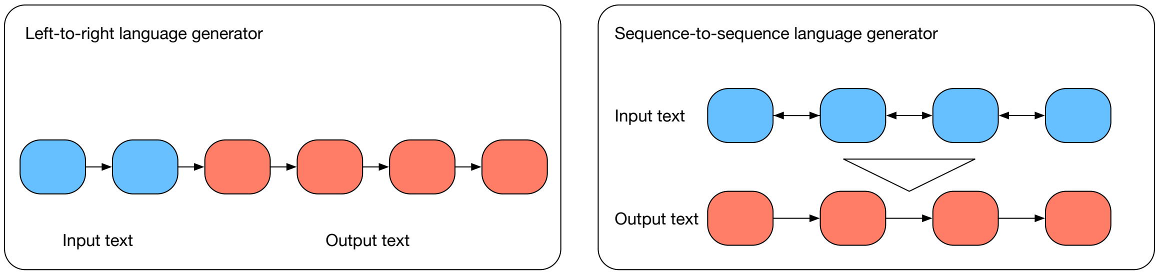

Left-to-right language generator
Input text
Output text
Sequence-to-sequence language generator
Input text
Output text

Figure 2-4: Left-to-right and sequence-to-sequence language generation models.

Besides the pretrained language models for general tasks, some language models are pretrained for different downstream tasks with tast-specific training strategies or datasets. For example, DialoGPT [Zhang et al., 2019] and BlenderBot [Shuster et al., 2020a] are pretrained on conversational corpora for dialog generation. Realm [Guu et al., 2020] and DPR [Karpukhin et al., 2020a] are pretrained for passage retrieval and open-domain question answering. All models listed above are pretrained with a self-supervised strategy and can be finetuned for related downstream NLP tasks. A summarization of proposed pretrained language models is shown in Figure 2-5.

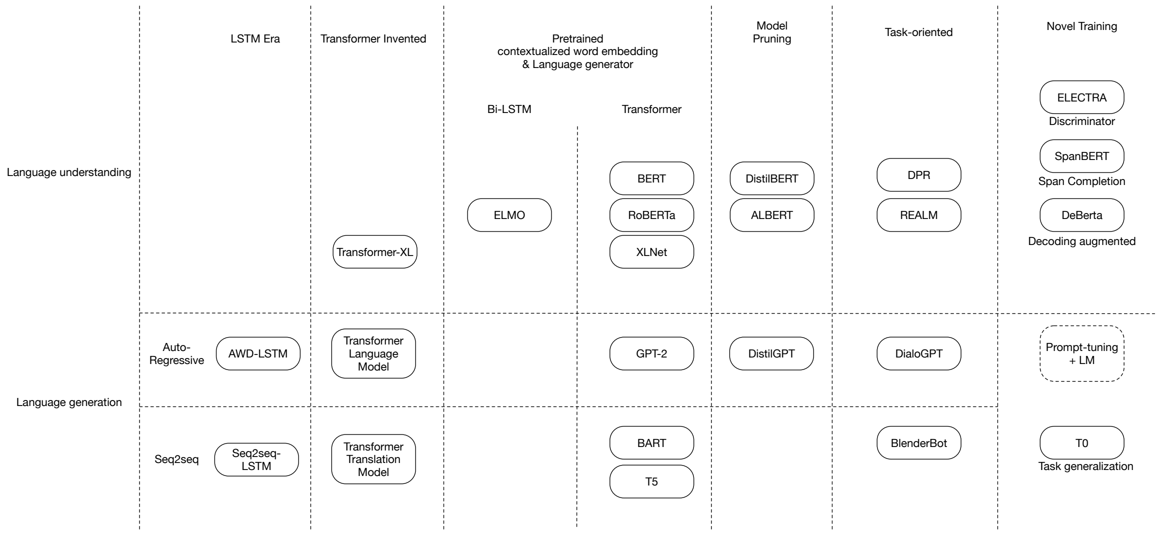


Figure 2-5: A summarization of the development of neural language models.

2.2 Training and Finetuning Language Models

The language models introduced in the previous section are trained with large corpora using a self-supervised training strategy, which models the coherence of a piece of text x with its context c

$$P(x|c) = \texttt{LM}(c) \tag{2.2}$$

where `LM` stands for neural language models. For BERT-like models, x can be a masked token, word, span, or the following sentences. For left-to-right language models like GPT-2, c stands for the current input and x is the next word. For sequence-to-sequence models, c represents the input sequence and x is the target output. The models are usually trained with cross entropy losses for each token to maximize the likelihood that x is observed around context c. Some additional training strategies can improve the finetuning performance for specific tasks. For example, modeling word replacement can improve the accuracy of extractive question answering [Clark et al., 2020], and pretraining with a maximum mutual-information objective benefits both dialog generation and machine translation models [Zhang et al., 2019]. Language encoders can be improved with decoding-aware training [He et al., 2020], and pretraining can become more efficient with downstream-aware task generalization [Sanh et al., 2021].

Finetuning pretrained language models can follow the regular task-specific training pipeline to tune all parameters, while another option is prompt tuning [Lester et al., 2021]. In prompt tuning, model parameters are fixed and N prompt tokens, each with a $d-$dimensional trainable prompt embedding, are concatenated to the input texts. As a result, instead of tuning a large number of parameters in the language model, the size of training parameters in prompt tuning is only $N \cdot d$. Experiments in [Lester et al., 2021] showed that by fixing the majority of parameters, prompt-tuned models are more robust against domain and task switch by mitigating the overfitting led by regular whole-model tuning. The prompt tuning method is shown in Figure 2-6. Because of the nature of data-driven, maximum-likelihood optimization methods, the performance of both model tuning and prompt tuning strategies is influenced

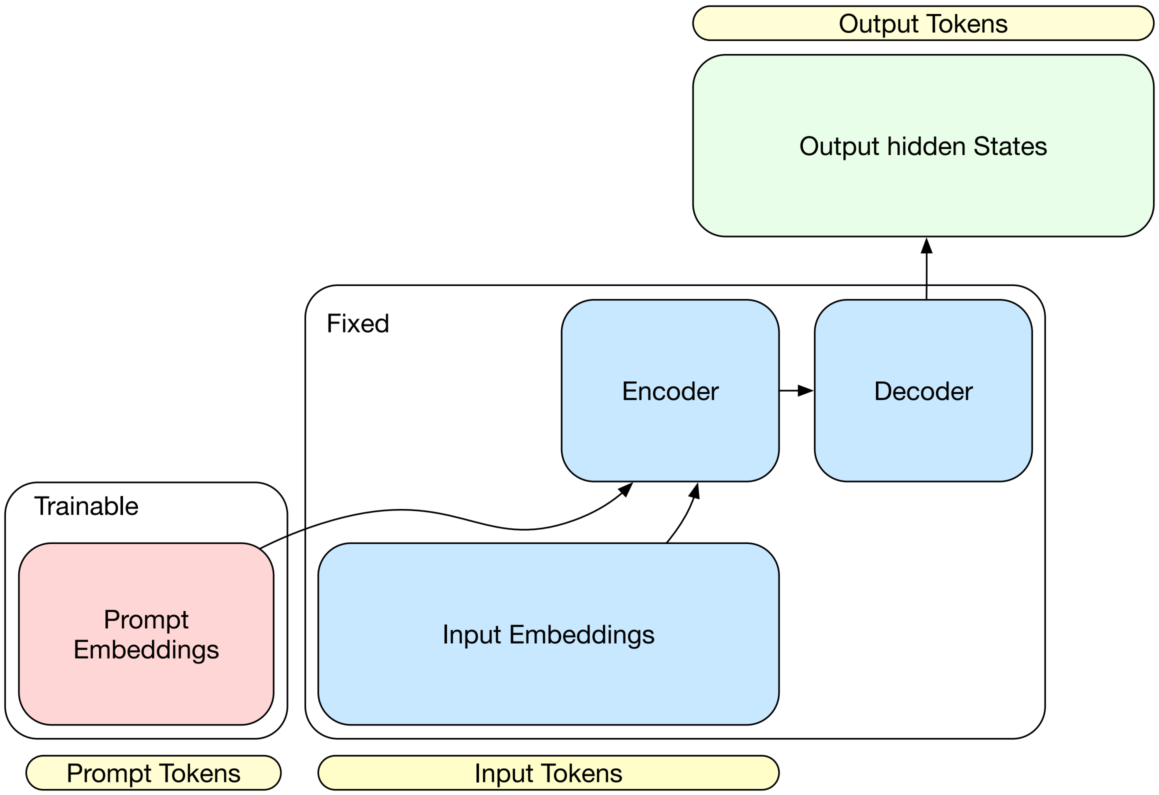


Figure 2-6: Prompt tuning of pretrained language models.

when there is not enough data for downstream tasks. A common solution to this is processing as much information as possible in the pretraining phase.

2.3 Reinforcement Language Modeling

Besides regular pretraining and finetuning on large text corpora and task benchmarks, there are situations that we hope the language models can be trained dynamically where the available annotated task data is not enough. To achieve this goal, the models are tuned with policy gradients to optimize undifferentiable objectives.

Self-critical sequence training [Rennie et al., 2017] if proposed for directly optimizing the BLEU and Rouge scores for image captioning models using reinforcement learning, since the BLEU and Rouge scores between generated and annotation captioning texts are not differentiable w.r.t. model parameters. A similar method is proposed in [De Vries et al., 2017], where an image caption model and an image retrieval model collaborate to complete an image guessing game. To successfully retrieval the target image, the captioning model needs to provide a natural and accurate description. The success of image retrieval is also undifferentiable, thus policy gradient is applied to encourage the captioning model to generate informative queries.

Another task is generating natural texts that are similar to human-generated sentences. To achieve this goal, adversarial generative training [Goodfellow et al., 2014] is applied to train sequence generation models, namely SeqGAN [Yu et al., 2017]. There is no downstream task provided for the generative training, but a discriminator is trained to classify machine- and human-generated texts. The text generator is rewarded if it successfully makes the discriminator to make a wrong prediction. Since the sequence is discrete so that the entire pipeline is not differentiable as GANs that generates images, the training signals are also propagated with policy gradients.

2.4 Transfer Learning

Although PLMs have led to significant improvements on different NLP benchmarks and mitigated the difficulty of applying language models to different tasks and domains, the data distribution gap between training and evaluation domains remains a difficult problem to solve since self-supervised pretraining does not involve any information about downstream tasks, which is crucial for finetuning, for example, question answering and sentiment analysis. With the pretrained parameters, the language models can converge quickly during finetuning, but also can overfit simple training cases. The finetuning performance can be decreased to the level of random guessing by creating adversarial evaluation sets [Bartolo et al., 2020, Wang et al., 2021].

Another method for transfer learning is automatically generating training data on unlabeled corpora using pretrained models, and train a new model on the synthetic data, namely self-training in related preliminary studies [Rosenberg et al., 2005, Zoph et al., 2020, He et al., 2019]. Although it is obvious that the synthetic data is more noisy than human annotated data and the domain / task gap still exists, training language models with synthetic data allows the model to process the texts of the target domain and partly capture the data distribution of the target domain. [He et al., 2019] also points out that training on synthetic labels benefits the training as a smoothing regularization. However, the proposed self-training methods for NLP tasks are straightforward, and there is no systematic study to explore the best practice of self-training for natural language processing.

Chapter 3

Joint Retrieval-Extraction Training for Evidence-Aware Dialog Response Selection

3.1 Introduction

In this chapter, we propose a self-training method that learns interpretable dialog response generation by generating soft pseudo labels.[1] Dialog response selection is an important function in a complete dialog system to retrieve human generated texts or re-rank machine generated responses. Given an input query and candidate texts, an encoder is applied to encode, score, and rank the query-candidate pairs. To achieve this goal, the models are optimized for assigning higher scores to the ground-truth candidates as compared to negative candidates during training [Dinan et al., 2019, Gunasekara et al., 2019].

State-of-the-art dialog response selection systems apply the Transformer architecture [Vaswani et al., 2017] with pre-trained parameters, for example BERT [Devlin et al., 2018] and ConveRT [Henderson et al., 2019a]. The model architecture that achieves the best scoring performance with a pre-trained transformer model is called a cross-encoder

[1]Based in part on the paper "Joint Retrieval-Extraction Training for Evidence-Aware Dialog Response Selection," by H Luo, J Glass, G Lalwani, Y Zhang, SW Li, in Proc. Interspeech, 2021.

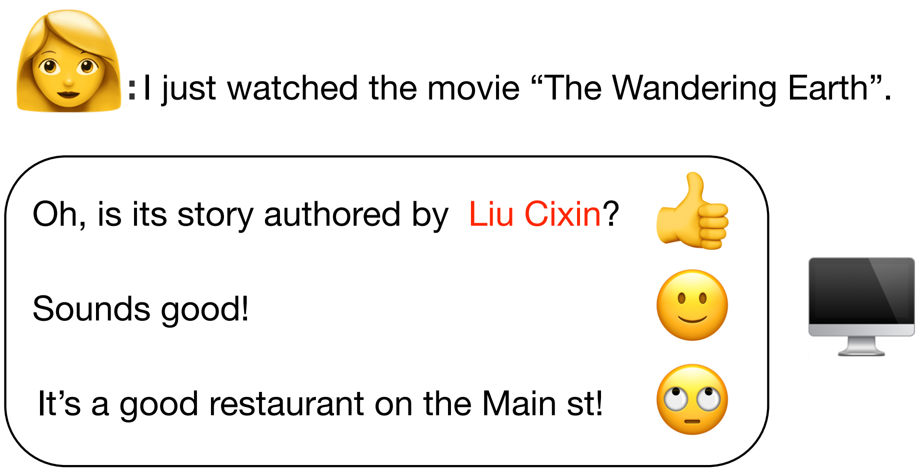

: I just watched the movie "The Wandering Earth".
Oh, is its story authored by Liu Cixin?
Sounds good!
It's a good restaurant on the Main st!

Figure 3-1: An example of a dialog response selection. The phrase in red is the reason why the first candidate is the best response for the given dialog history. Human can tell the best response as well as the reason or evidence in a dialog response selection task.

[Humeau et al., 2019b]. By concatenating the input query with each candidate and encoding them jointly with a transformer network, improved representations of query-candidate pairs are generated. The candidate score of a sequence is obtained by feeding the output embedding of the first token of the entire sequence (e.g. [CLS] in BERT) into a linear layer. The sequence which is assigned with the highest score among all candidates is selected as the retrieved result. Under this setting, the information encoded in other positions / tokens is not used explicitly.

On the other hand, extractive question answering (QA) and reading comprehension [Rajpurkar et al., 2016, Gao et al., 2019] models apply similar architectures for sequence scoring but utilize output embeddings of all the tokens during training and inference. Each question and context passage are concatenated into a sequence, and the embedding of each token in the sequence is fed into standard feed-forward and softmax layers to compute the token score. These token scores are used to extract the span of tokens that can answer the question given the context. The question is predicted as not answerable given the context if the [CLS] token yields the maximum score. The architecture suggests that token-level (i.e., span extraction) and sequence-level (i.e., answerable or not) tasks can be modeled jointly.

Motivated by the joint modeling of token- and sequence-level information, we propose a novel Retrieval-EXtraction (REX) training strategy for dialog response selections. In REX, we design two objectives for retrieval and extraction, which guides the model to select the correct responses and attend to the relevant token-level evidence respectively. As compared to the conventional cross-encoder, the evidence attention improves both performance and interpretability. It is worth noting that in REX the retrieval scores from the embedding of the first-token ([CLS]) and the extraction scores work in the opposite manner as compared to QA - higher scores from [CLS] in REX represents possible responses. In addition, most dialog response selection corpora annotate no token-level evidence that explains the best response. To learn the evidence without human annotation, we design a transformation and a regularization function to relate the retrieval and extraction scores properly, learn token-level attention in an unsupervised way, and stabilize the training. We evaluate REX with the ConvAI2 and DSTC7 Track 1 challenges, showing REX achieves the state-of-the-art (SOTA) result by a large margin. To summarize our main contributions:

- We propose a combined retrieval and extraction training method for cross-encoder models (REX-encoder).
- We design a transformation and a regularization function to relate the retrieval and extraction scores, encourage the attention to token-level evidence, and stabilize model training.
- We show REX achieves a new SOTA in the popular dialog response selection corpus, DSTC7, and demonstrate that the learned evidence attention improves interpretability.

3.2 Related Work

There are many tasks that adopt the setup of scoring or ranking a set of candidates given some query (context). One of the most common tasks is Dialog Response Selection. The task is either employed by dialog retrieval systems or in general conversational systems for evaluating their language understanding capabilities. Recently, it has become popular to use this task as one of the pre-training strategies for learning powerful pre-trained representations to be leveraged in conversational systems instead of BERT. [Henderson et al., 2019b, Henderson et al., 2019a, Humeau et al., 2019b] use a dialogue response selection objective and a huge Reddit conversational corpus to pre-train more conversational encoders. Experiments show the encoders yield better performance as compared to general BERT representations pre-trained with Wikipedia data. [Wu et al., 2020] jointly learn a response contrastive loss and masked language modeling loss to train a more task-oriented dialog pre-trained model initialized from BERT parameters. The neural language models are widely applied in the domain of goal-oriented dialog systems [Ramadan et al., 2018, Wen et al., 2017b], end-to-end dialog generation [Zhao et al., 2019, Liu et al., 2018, Wen et al., 2017a], dialog state tracking [Williams, 2014].

Besides dialogs, sequence scoring is directly applicable to Open Domain QA [Chen et al., 2017a, Lee et al., 2019]. In the task, the first step is to retrieve top relevant context passages or documents that may contain the answer, and then apply machine comprehension models to extract a token span for answers [Chen et al., 2017b]. Recent approaches like [Lee et al., 2019, Karpukhin et al., 2020b] leverage pre-trained transformer models to encode a question and documents, and then retrieve relevant documents using cosine similarity between these encodings. More recently, sequence scoring is also used in obtaining language agnostic sentence representations [Feng et al., 2020] by ranking various target sentences based on similarity in cross-lingual embedding space.

3.3 Cross-Encoders

In this section, we introduce two of the most popular methods for sequence scoring with the transformer models. In the following, we refer to dialog context or questions as queries and sequences for selecting response or answer spans as candidates. The first method for scoring query-candidate pairs is a bi-encoder, that generates the query and candidate embeddings separately and calculates their inner products as scores. A bi-encoder is fast to calculate since the embeddings can be pre-computed separately and reused, but yields suboptimal performance. In contrast, a cross-encoder, the type of method explored in this work, yields state-of-the-art (SOTA) performance for sequence scoring. The method encodes the concatenation of query Q, candidates C, and special tokens,

$$E^i = T([CLS], Q, [SEP], C^i, [SEP])$$

$$X_i = E^i \cdot W$$

where $Q = [q_1, q_2, \ldots, q_m]$ is the input query, $C^i = [c_1^i, c_2^i, \ldots, c_n^i]$ is the $i-$th candidate sequence, T stands for a transformer model, $E^i = [e_1^i, e_2^i, \ldots, e_{m+n+3}^i]$ are output embeddings of all tokens in the concatenated sequence including special tokens, and W represents a linear layer. The final output scores are $X_i = [x_0^i, x_1^i, \ldots, x_{m+n+3}^i]$. Given that the cross-encoder considers concatenated query-candidate pairs and computes self-attention over entire concatenated sequence, it is much more expensive than a bi-encoder.

In a retrieval task such as dialog response selection, the model is trained to select the best candidate sequence. Given a set of candidates, the retrieval probability of candidate sequence i is calculated with a Softmax using the scores of the first tokens or the candidates,

$$P_R(i) = \frac{\exp(x_0^i)}{\sum_j \exp(x_0^j)} = \frac{\exp(x_0^i)}{Z_R} \tag{3.1}$$

Where the superscripts i and j stand for sequence IDs, and the subscript 0 stands for the token ID within a sequence. On the other hand, in an extraction task, such as extractive question answering or dialog state tracking, the goal of a model is to extract an answer span if a reasonable answer exists. For simplicity, we assume the extraction targets are single-token, so that the probability a token k extracted in candidate i is

$$P_E(k)^i = \frac{\exp(x_k^i)}{\sum_j \exp(x_j^i)} = \frac{\exp(x_k^i)}{Z_E^i} \tag{3.2}$$

where Z_R and Z_E^i are retrieval and extraction potentials respectively. If no answer is available, the model extracts the first token, [CLS], of the input sequence.

3.4 REX Encoder

3.4.1 Bridging Retrieval and Extraction

In our work, we propose a combined supervised **Retrieval**, unsupervised **EXtraction** learning algorithm based on cross encoders (REX encoder) for dialog response selection. The retrieval and extraction task guides the model to rank candidate texts and to attend to supporting evidence respectively. The REX encoder learns retrieval- and extraction- probabilities jointly by assuming that when the training converges, both retrieval and extraction probabilities (P_R and P_E) follow the properties below. If sequence i is the true candidate, then

$$P_R(i) = P_E(k)^i = 1, \exists k > 0 \tag{3.3}$$

$P_R(i)$ is the probability of sequence i being the true candidate. $P_E(k)^i = 1, \exists k > 0$ indicates that a token other than [CLS] in the sequence is extracted as the answer. If sequence i is not the true candidate (i.e., negative sequence), the retrieval probability is 0 and the model extracts the [CLS] token as the answer. That is,

$$P_R(i) = 0; P_E(0)^i = 1 \tag{3.4}$$

With the above properties, we expect that for all candidate sequences, when the training converges,

$$P_R(i) = 1 - P_E(0)^i \tag{3.5}$$

3.4.2 Transformed Extraction Scores

Note that in the retrieval task, a large x_0^i score indicates that the sequence should be selected, while in the extraction, a large x_0^i score means that the context does not contain the answer. As a result, it is necessary to resolve this conflict when computing $P_R(i)$ and $P_E(k)^i$ in a combined model.

To derive the solution, we first formulate the retrieval and extraction probability with the token scores. Given the raw outputs from the model $X_i = [x_0^i, x_1^i, \ldots, x_N^i]$, we use x_0^i to compute the retrieval probability with Equation 3.1. For extraction, we first transform x_k^i with a function $f(\cdot)$, and then apply Softmax to obtain the probability

$$P_E(k)^i = \frac{\exp(f(x_k^i))}{\sum_j \exp(f(x_j^i))} = \frac{\exp(f(x_k^i))}{Z_E^i} \tag{3.6}$$

Obviously, to avoid a conflict, we need $f(x)$ to be small when x is large and vice versa. In the following, we show that $f(x)$ can be as simple as $-x$, and derive some modification required in training loss for making the transformation.

Theorem 1. If $f(x) = -x$, our ultimate goal, Equation 3.5, holds when

- true candidates and $\log Z_E^i = \log Z_E^{i,[1:N]}$,
- wrong candidates and $-\log \sum_{j=1}^N e^{-x_i^j} = \log Z_R$.

Proof. To derive the exact form of the transformation function $f(\cdot)$, we rewrite Equation 3.5 in log probability space, and we can get the following equation

$$\log Z_E^i - \log Z_E^{i,[1:N]} = -x_i^0 + \log Z_R \tag{3.7}$$

where N is the length of the entire sequence and

$$Z_E^{i,[1:N]} = Z_E^i - \exp(f(x_i^0)) \tag{3.8}$$

which is the potential of the sequence without the [CLS] token. Note that $\log Z_E$ and $\log Z_E^i[1:N]$ are *LogSumExp* functions, which smoothly approximate the maximum extraction scores

$$\log Z_E^i \approx \max(f(x_i)_0^N) \tag{3.9}$$

If the candidate is true, it is trivial that when both model converges,

$$\log Z_E^i - \log Z_E^{i,[1:N]} \approx -x_i^0 + \log Z_R \approx 0 \tag{3.10}$$

With the approximations, we attempt to calculate $f(\cdot)$ by assuming $f(x) = -x + c$, where c is a constant scalar. Consider the left hand side of Equation 3.7 and assume the sequence is not the retrieval result. In this case, x_0^i is low and $f(x_0^i)$ is the highest extraction score among all tokens

$$\begin{aligned}
&\log Z_E^i - \log Z_E^{i,[1:N]} \\
&= \log \sum_{j=0}^{N} e^{-x_j^i + c} - \log \sum_{j=1}^{N} e^{-x_j^i + c} \\
&= \log \sum_{j=0}^{N} e^{-x_j^i} + c - \log \sum_{j=1}^{N} e^{-x_j^i} - c \\
&\approx -x_0^i - \log \sum_{j=1}^{N} e^{-x_j^i}
\end{aligned} \tag{3.11}$$

The last line of Equation 3.11 suggests that the value of c is irrelevant to the residual value of the two extraction potentials. Thus, we can simply set

$$f(x) = -x \tag{3.12}$$

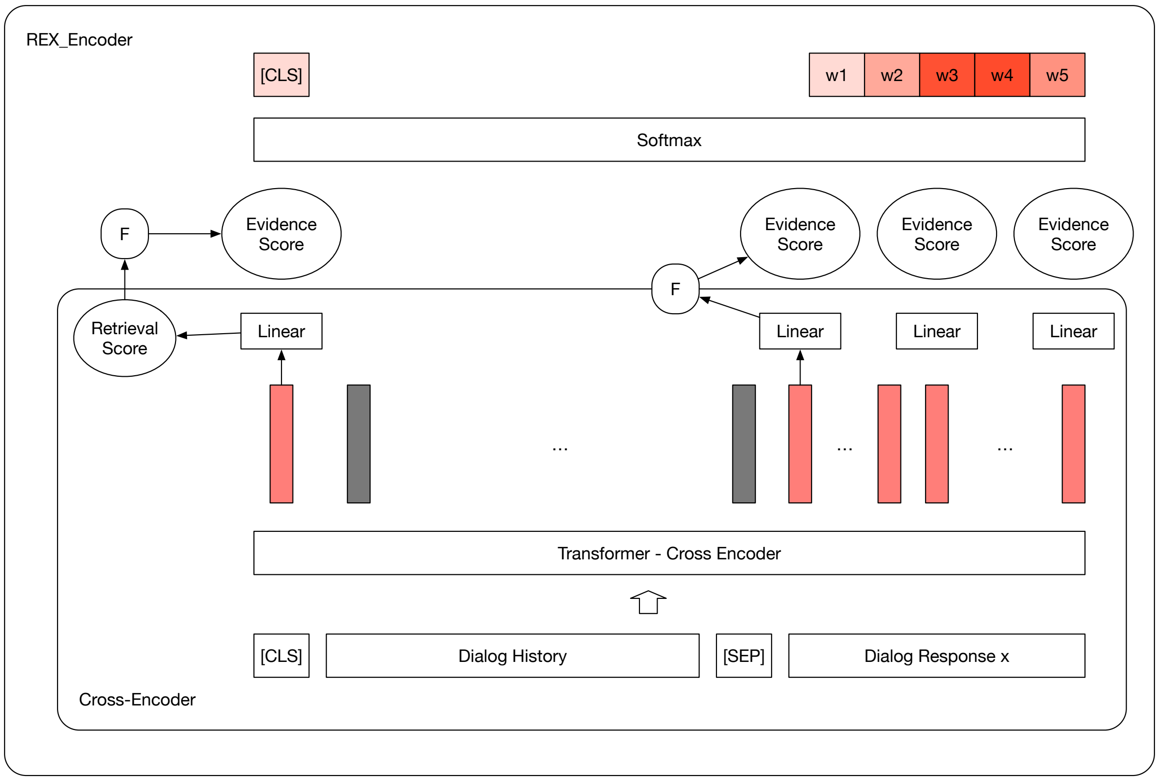
For Equation 3.7 to hold, we should make sure that

$$-x_0^i - \log \sum_{j=1}^{N} e^{-x_j^i} = -x_0^i + \log Z_R \tag{3.13}$$

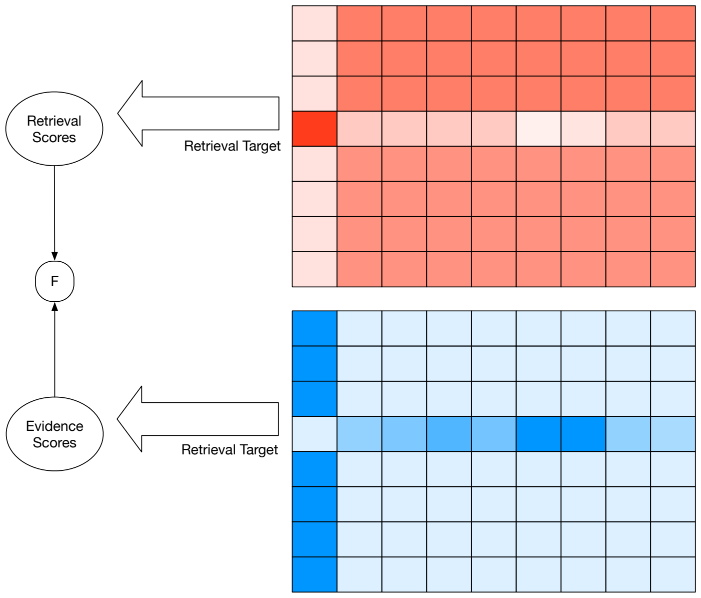
We can eliminate x_0^i on both sides of the equation and obtain $-\log \sum_{j=1}^{N} e^{-x_j^i} = \log Z_R$. To encourage the trained model satisfying $-\log \sum_{j=1}^{N} e^{-x_j^i} = \log Z_R$ (and thus Equation 3.5) when converged, we add a regularization term, called distribution bridging loss, during training

$$l_{bridge}^i = || - \log \sum_{j=1}^{N} e^{-x_j^i} - \log Z_R ||_1 \tag{3.14}$$

Note that the term only applies to the negative sequences.



(a) REX-encoder architecture



(b) Learned retrieval and evidence scores

Figure 3-2: The REX-encoder model and an example of target retrieval and extraction scores. In the REX-encoder in (a), the inner block contains the standard cross-encoder architecture, and the REX encoder consists of a cross-encoder model and an unsupervised extraction module. As shown in (b), retrieval and extraction are opposite to each other. The first token of the ground-truth candidate should receive a high retrieval score and a low extraction score. Each row stands for a candidate responses, and each column stands for tokens in the utterances.

3.4.3 Unsupervised Extraction Learning

In most retrieval tasks and corpora, only the true passages for retrieval are annotated and there are no explicit token-level labels for supporting evidence. Thus, it is difficult to obtain supervised extraction learning signals. With this limitation, we introduce an unsupervised method for the extraction learning, to train the model to focus on the [CLS] tokens for negative sequences and attend to keywords for true candidates.

Negative Sequences

For negative sequences, we expect the extraction model to focus on the [CLS] token because there is no evidence that supports the sequences to be good retrieval targets. To achieve this goal, we generate a one-hot label that highlights the [CLS] token

$$l = [1, 0, 0, \cdots, 0] \tag{3.15}$$

Then, we train the extraction model with an extraction loss, which is the cross-entropy loss based on l plus the distribution bridging loss discussed in Equation 3.14.

$$l_E^{i,neg} = 1 \cdot \log \frac{\exp f(x_0)}{Z_E^i} + \lambda \cdot l_{bridge} \tag{3.16}$$

Here the label l is used to make the model focus on the [CLS] token in negative sequences. Since we generate l from the retrieval label automatically, the extraction learning for negative sequences is unsupervised.

True Sequences

For true sequences, the goal of the extraction model is to extract keywords, or evidence, from the text instead of paying attention to the [CLS] token. In this case, a trained model should assign small values to $f(x_i^0)$. Hence, our approximation in Equation 3.11, which assumes that $f(x_i^0)$ is the largest extraction score in a sequence, does not work anymore.

Instead, for a true sequence i, the model should assign a high retrieval probability

$$\log P_R(i) = x_i^0 - \log Z_R \approx 0 \tag{3.17}$$

and low extraction score $f(x_0)$ that makes

$$\log Z_E^i - \log Z_E^{i,[1:N]} \approx 0 \tag{3.18}$$

with the above approximations, Equation 3.7 holds

$$\log Z_E^i - \log Z_E^{i,[1:N]} \approx -x_0^i + \log Z_R \approx 0 \tag{3.19}$$

Note that $-x_0^i + \log Z_R = 0$ is the objective function of the retrieval training, and we force the leftmost part to approach zero by minimizing it as our extraction loss:

$$l_E^{i,true} = \log Z_E^i - \log Z_E^{i,[1:N]} \tag{3.20}$$

Since we usually do not have any annotation on the extraction target (i.e., evidence or keywords), we design the implicit supervision to reward any attention distributed on text tokens and penalize the attention assigned to the [CLS].

To sum up, for a query and a set of candidates including one true sequence, i, and others as negative samples, we propose the overall loss function for unsupervised extraction learning

$$l_E = l_E^{i,true} + \sum_{j \in NEG} (l_E^{j,neg} + \lambda \cdot l_{bridge}^j) \tag{3.21}$$

3.4.4 Overall Architecture

As mentioned above, we combine a supervised-retrieval and an unsupervised-extraction task to train the REX encoder. We show the complete architecture of the proposed REX-encoder model in Figure 3-2a along with an example of the learned retrieval and extraction scores in Figure 3-2b. The overall loss function for REX training is a

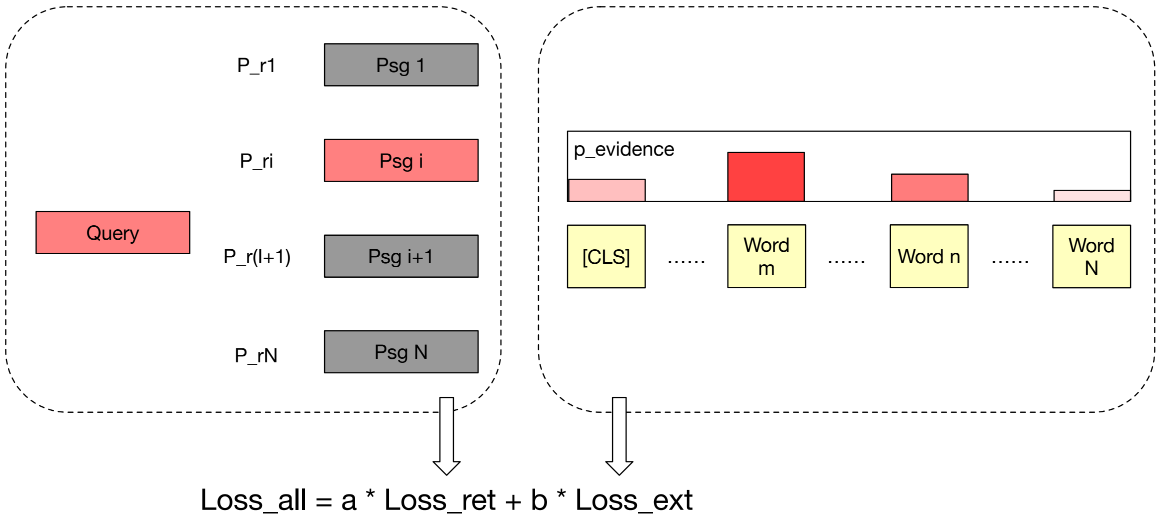


Figure 3-3: The loss function for training the REX encoder.

weighted combination of retrieval and extraction losses with coefficients α and β,

$$L = \alpha \cdot l_R + \beta \cdot l_E \tag{3.22}$$

where l_E has been shown in Equation 3.21 and

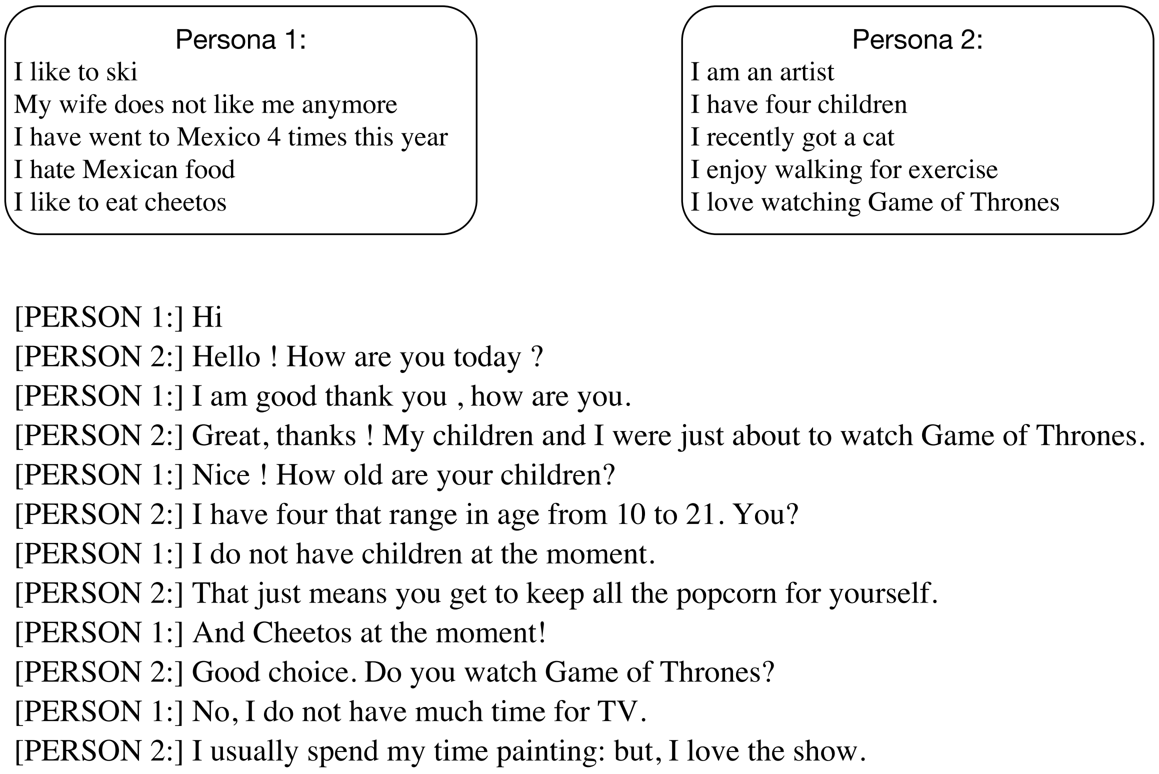
$$l_R = -\log \frac{\exp(x_0^i)}{Z_R} \tag{3.23}$$

In the following sections, we will evaluate REX on dialog response selection and explore the effect of the values of α and β. The loss function is illustrated in Figure 3-3.

3.5 Experiments

3.5.1 Task and Corpus

We evaluate the dialog response selection performance of the model on the ConvAI2 [Dinan et al., 2019] task and the DSTC7 challenge Track 1 Ubuntu dialog corpus [Gunasekara et al., 2019]. ConvAI2 is based on the persona-chat dataset [Zhang et al., 2018] where each participant of a conversation is given a persona The dataset contains 1,155 personas, each consisting of at least 5 profile sentences. The



Persona 1:
I like to ski
My wife does not like me anymore
I have went to Mexico 4 times this year
I hate Mexican food
I like to eat cheetos

Persona 2:
I am an artist
I have four children
I recently got a cat
I enjoy walking for exercise
I love watching Game of Thrones

[PERSON 1:] Hi
[PERSON 2:] Hello ! How are you today ?
[PERSON 1:] I am good thank you , how are you.
[PERSON 2:] Great, thanks ! My children and I were just about to watch Game of Thrones.
[PERSON 1:] Nice ! How old are your children?
[PERSON 2:] I have four that range in age from 10 to 21. You?
[PERSON 1:] I do not have children at the moment.
[PERSON 2:] That just means you get to keep all the popcorn for yourself.
[PERSON 1:] And Cheetos at the moment!
[PERSON 2:] Good choice. Do you watch Game of Thrones?
[PERSON 1:] No, I do not have much time for TV.
[PERSON 2:] I usually spend my time painting: but, I love the show.

Figure 3-4: Example dialog of the ConvAI2 task.

dataset contains 17, 878 conversations for training and 1, 000 conversations for evaluation An example conversation is shown in Figure 3-4. The DSTC7 corpus contains conversations about Ubuntu-related topics, most of which are questions and answers among users of the Ubuntu community. The entire corpus contains 135,078 dialog threads. The annotations of the challenge only contains true responses, with no token-level evidence. Some examples are shown in Figure 3-6 and 3-7. We conduct supervised retrieval training with the annotated ground-truth responses and unsupervised extraction training following the process described in Section 3.4.3.

3.5.2 Implementation Details

We implemented our model with the ParlAI framework[2] based on its cross-encoder implementation. We used the BERT-large model pre-trained on the Reddit dialog corpus [Mazaré et al., 2018] provided by ParlAI. Following [Shuster et al., 2020b], we

[2]`https://parl.ai/`

apply the Adam [Kingma and Ba, 2015] optimizer with 0.01 weight decay rate. We initialize the learning rate as $5e-5$ with 1000 warmup steps and the decay rate for every half epoch is 0.4. In practice, we set $\alpha = 1$, $\beta = 5$, and the regularization coefficient $\lambda = 10^{-5}$. The authors of [Shuster et al., 2020b] trained the state-of-the-art model with 8 Volta 100 GPUs with batch size as 16. Due to limited computational resources, we trained our model on 2 Volta 100 GPUs and the largest batch size we were able to use is 8. As a result, we are comparing our model with both the reported performance in [Shuster et al., 2020b] and our reproduced state-of-the-art baseline performance on our own machines. We conduct all experiments with the settings described above unless specified otherwise.

3.5.3 Retrieval Performance

We evaluate our model on the dialog response selection task with two metrics, Recall@1 (R@1) given C candidates, and mean reciprocal rank (MRR). The experiment results are summarized in Table 3.1. Since the test set ConvAI2 challenge is not publicly available, we evaluate our model on the development set. For a fair comparison, we used the officially recommended hyper-parameter settings[3] without any tuning. Experiments show that the REX encoder outperforms the original SOTA model on ConvAI2 Dev set and achieved new state-of-the-art performance on the DSTC7 Track 1 challenge despite the fact that we have limited computational resources.

Analyzing the performance gaps according to R@1/100 scores on the DSTC7 corpus suggests that REX yields significant improvement. As shown in Table 3.2, the improvement achieved by applying the poly-encoder instead of the bi-encoder is 0.5%, while the state-of-the-art cross-encoder model outperformed the bi-encoder by 0.8%. Meanwhile, our model outperforms our own cross-encoder by 2.1%, and even the reported state-of-the-art performance of cross-encoders by 1.2%, which is still larger than both the reported Cross vs Bi and Poly vs Bi performance gaps.

[3] `https://parl.ai/projects/polyencoder/`

Models	R@1	MRR
ConvAI2-Dev		
Cross-encoder (Ours)	90.0	94.1
Rex-encoder	**90.5**	**94.3**
DSTC7		
[Gu et al., 2018]	60.8	69.1
[Chen and Wang, 2019]	64.5	73.5
Bi-encoder	70.9	78.1
Poly-encoder 360	71.4	78.3
Cross-encoder	71.7	79.0
Cross-encoder (Ours)	70.8	78.0
Rex-encoder	**72.9**	**79.4**

Table 3.1: Retrieval performance of the baseline models and the proposed REX encoder on ConvAI2 and DSTC7 track 1 challenge, where the cross-encoder is the previous state-of-the-art model. We compare both the reported performance trained with 8 Volta 100 GPUs and batch size 16, and the performance reproduced by training on our machine with 2 Volta 100 GPUs and batch size 8. The latter is denoted as Cross-encoder (Ours). The experiment results of bi-, poly-, and cross-encoders are reported by [Shuster et al., 2020b].

3.5.4 Effects of Regularization

We further analyze the effect of the distribution-bridging regularization with the bridging loss l_{bridge} described in Equation 3.14. Results with and without the bridging regularization are listed in Table 3.3. According to the results, the regularization improves the retrieval performance under our best model setting.

3.5.5 Which Task is More Important

As we discussed before, we combine the retrieval and extraction losses with coefficients α and β. While the primary task is retrieval, it is interesting to explore different α and β settings. If α is larger, the training is mostly guided by the retrieval loss; if β is larger than α, the majority of training signals would come from the extraction task.

For better visualization of the importance of the retrieval and extraction sub-tasks, we fix $\alpha = 1$ and show the performance of different choices of β. The performance of different settings of task weighting is shown in Figure 3-5. The plot indicates that

Comparisons	**R@1/100 Gaps**
Poly vs Bi	0.5
Cross vs Bi	0.8
REX vs Poly	1.5
REX vs Cross	1.2
REX vs Cross (Ours)	**2.1**

Table 3.2: Comparing the improvement of different model pairs. The improvement of our model over baseline is significant, since the performance gap between REX- and cross-encoders is larger than the improvement of the cross-encoder over the bi-encoder.

Metrics	**REX-Encoder**	**w/o Reg.**
R@1/100	72.9	71.9
MRR	79.4	78.6

Table 3.3: The comparison of the REX-encoder with and without the distribution bridging regularization.

applying a large value of β (i.e., having more training signals from the extraction tasks) leads to better performance, but we did not observe any improvement beyond $\beta = 5$. The result suggests that the token-level information is more important than sequence-level for the REX encoder.

3.5.6 Multi-tasking vs. Pooling

As we discussed above, one advantage of the REX-encoder model is that it utilizes all the output embeddings during training. There are other simple ways to use all the embeddings such as pooling. To show that the unsupervised extraction learning provides additional knowledge as compared to the simple approach, we compare the REX-encoder to cross-encoder with the following pooling methods,

- Scores mean and max pooling (S-mean/max): feed the output embedding of each token to a linear scorer, and calculate the retrieval score of the sequence by pooling the output scores of the linear layer.

- Embedding mean and max pooling (E-mean/max): pool the output embedding of each token, and calculate the retrieval score of the sequence by feeding the

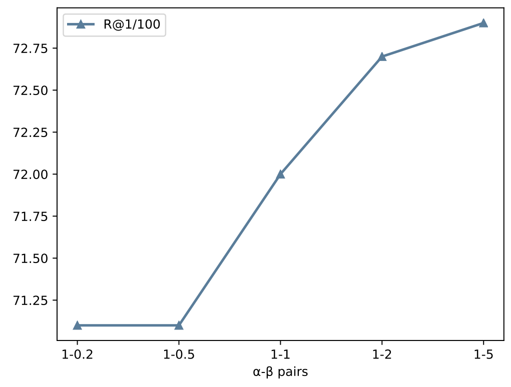


Figure 3-5: R@1 performance of different β settings.

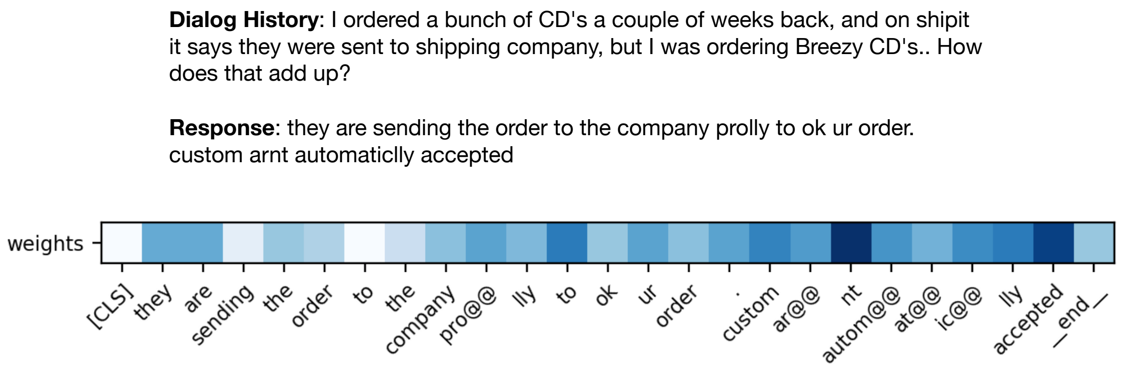
Strategies	**R@1/100**	**MRR**
[CLS]	70.8	78.0
S-Mean	70.1	77.2
S-Max	69.8	77.3
E-Mean	71.1	78.4
E-max	70.7	78.2
REX	**72.9**	**79.4**

Table 3.4: Comparing the REX-encoder with baseline models that also explicitly use all the output embeddings of the cross-encoder with different pooling methods.

pooled sequence embedding to a linear layer.

The above strategies make it possible for the cross-encoder model to use the output embedding of each token of the given sequence explicitly. Also, we use [CLS]-pooling to denote the original cross-encoder model.

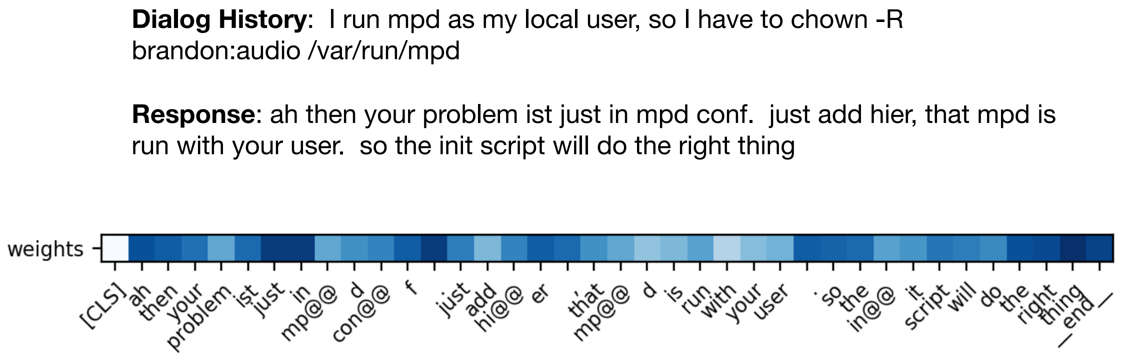
The experimental results in Table 3.4 showed that the no pooling strategy significantly outperforms the baseline. Except for the E-mean strategy that obtains slightly higher accuracy, all other strategies perform worse than the standard cross-encoder model applying the [CLS] pooling strategy. The result suggests that the embeddings of other tokens cannot provide additional information to improve the sequence-level embedding for response prediction if no extraction training signal is provided.



Dialog History: I ordered a bunch of CD's a couple of weeks back, and on shipit it says they were sent to shipping company, but I was ordering Breezy CD's.. How does that add up?

Response: they are sending the order to the company prolly to ok ur order. custom arnt automaticlly accepted

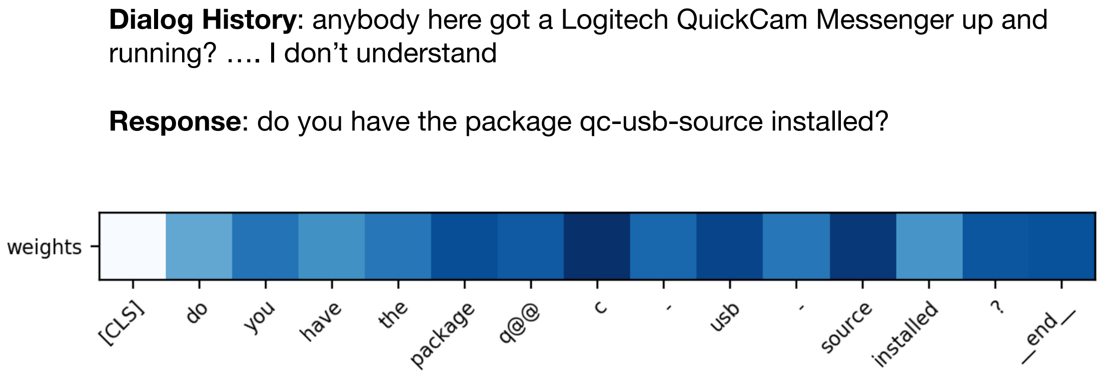
(a)



Dialog History: I run mpd as my local user, so I have to chown -R brandon:audio /var/run/mpd

Response: ah then your problem ist just in mpd conf. just add hier, that mpd is run with your user. so the init script will do the right thing

(b)

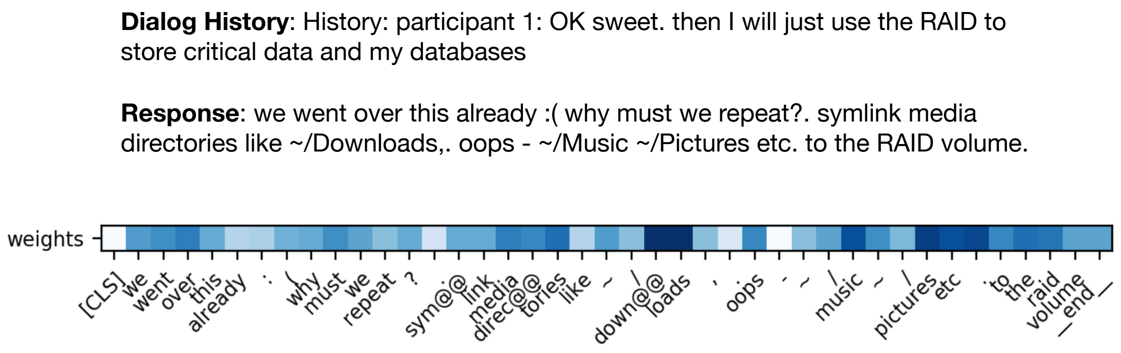


Dialog History: anybody here got a Logitech QuickCam Messenger up and running? …. I don't understand

Response: do you have the package qc-usb-source installed?

(c)

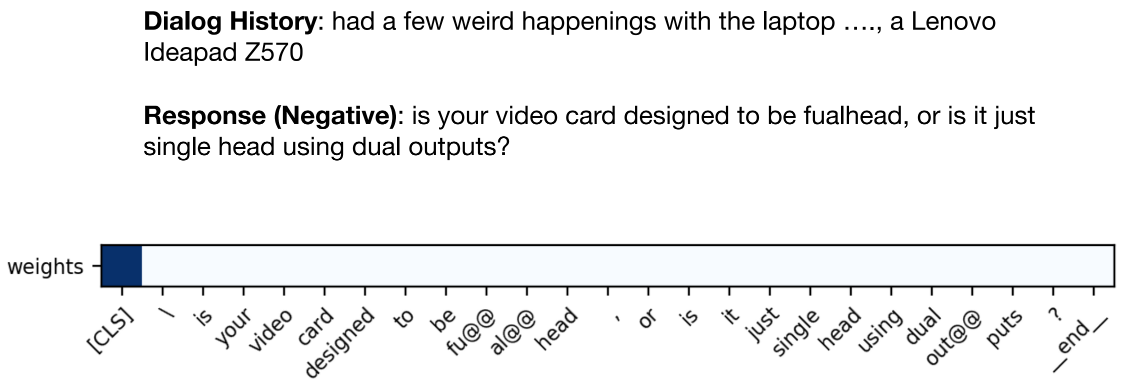
Figure 3-6: Visualization of the learned extraction model on the DSTC7 test set. The model attends to the evidence in the true response, and to the [CLS] token in a negative candidate.



Dialog History: History: participant 1: OK sweet. then I will just use the RAID to store critical data and my databases

Response: we went over this already :(why must we repeat?. symlink media directories like ~/Downloads,. oops - ~/Music ~/Pictures etc. to the RAID volume.

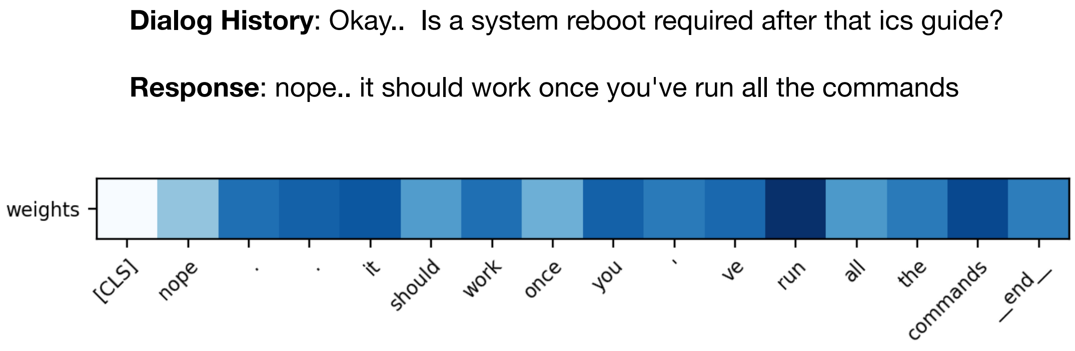
(a)



Dialog History: had a few weird happenings with the laptop …., a Lenovo Ideapad Z570

Response (Negative): is your video card designed to be fualhead, or is it just single head using dual outputs?

(b)



Dialog History: Okay.. Is a system reboot required after that ics guide?

Response: nope.. it should work once you've run all the commands

(c)

Figure 3-7: Visualization of the learned extraction model on the DSTC7 test set. The model attends to the evidence in the true response, and to the [CLS] token in a negative candidate.

3.5.7 Evidence Extraction

In this section, we analyze the quality of the learned extraction function. We calculate the extraction attention for each token with the learned model and visualize the scores in Figure 3-6 and 3-7 for sequences randomly sampled from the test set of DSTC7. The example in Figure 3-7b shows a negative candidate, while the others are ground-truth responses.

Visualizing the calculated extraction scores shows that the model attends to the [CLS] token in a negative candidate in Figure 3-7b. For the true candidates, the model attends to the response text and highlight the words that are coherent with the dialog history. In the example in Figure 3-6a the extraction model focuses on words "aren't" and "accepted" to answer the shipment question. The dialog history of Figure 3-6b discusses about mdp configurations, and the ground-truth is a long response containing three short sentences. the extraction model attends to each period. The most important part of Figure 3-6c is "qc-usb-source". The model successfully attends to "qc", "usb", and "source". In Figure 3-7a, the model focuses on the word "download" and "pictures", which link to the dialog history that discusses about critical and non-critical data. In Figure 3-7c, the dialog history asks about if a reboot is needed and the extraction focuses on "run commands" to answer the question.

3.6 Chapter Summary

In this chapter, we proposed a supervised-Retrieval, unsupervised-EXtraction (REX) method based on the cross-encoder transformer neural network to improve the accuracy and interpretability of dialog response selection. To achieve better retrieval performance with token-level evidence, we designed an extraction score transformation function and a regularization term for joint training. Experiments showed that REX significantly outperforms the cross-encoder baseline and achieves the new SOTA performance on the DSTC7 Track 1 challenge, without increasing the number of trainable parameters. Our analysis suggests that the proposed unsupervised extraction training leads to more improvement than simple poolings. The visualizations of the extraction results demonstrate that the model attends to evidence keywords helping determine whether the candidate is a good response, and thus enhance interpretability. The overall results showed that the self-training method based on soft pseudo evidence labels improves both accuracy and interpretability of dialog response selection systems.

Chapter 4

Prototypical Q Networks for Automatic Conversational Diagnosis and Few-Shot New Disease adaptation

4.1 Introduction

The previous chapter discussed dialog response selection, and in this chapter, we apply a self-training method for dialog action selection in an automatic diagnosis systems by learning prototypical label embeddings in a self-supervised manner.[1] Recently, spoken dialog systems have been a popular research topic in the human language technology (HLT) area with various applications. Among these applications, the dialog system for clinical conversation (i.e., medical agent) is a rising direction for its widespread and impactful use [Wei et al., 2018]. A medical bot assists medical practitioners to converse with patients, collecting information about their symptoms, physical and mental conditions, or even making suggestions on the diagnosis. The bot has significant potential to make the diagnostic procedure more efficient. An example of an automatic diagnosis dialog system is shown in Figure 4-1. Starting

[1]Based in part on the paper "Prototypical Q Networks for Automatic Conversational Diagnosis and Few-Shot New Disease adaptation," by H Luo, SW Li, and J Glass, 2020. Proc. Interspeech 2020.

from a self-report, the medical bot collects and distills symptom information before it makes the disease prediction.

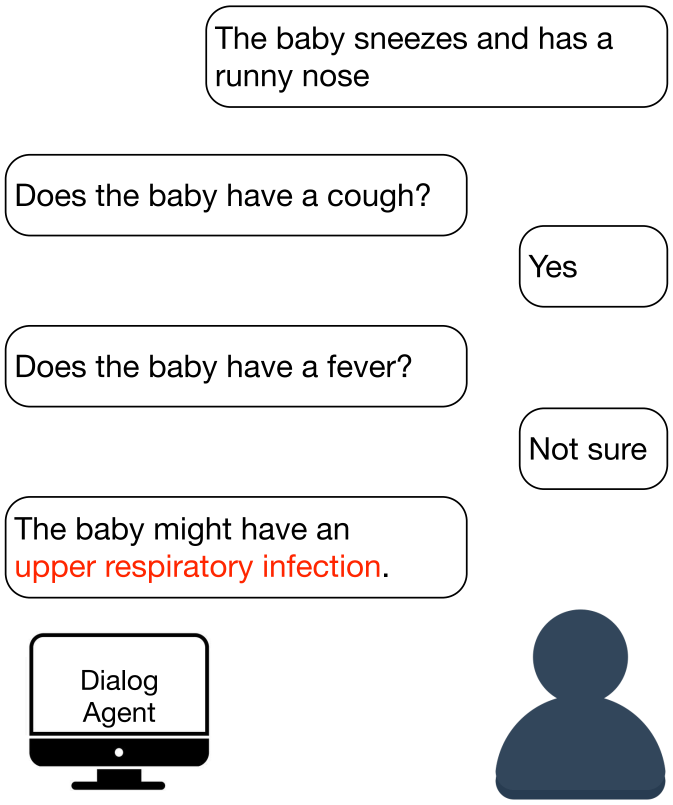

The baby sneezes and has a runny nose
Does the baby have a cough?
Yes
Does the baby have a fever?
Not sure
The baby might have an upper respiratory infection.
Dialog Agent

Figure 4-1: An example of a dialog between a patient and a medical agent. First, the patient provides a self report. Then the agent conducts a dialog by requesting symptoms and concludes by making a decision about the disease.

One of the core challenges of building such a dialog system is to design and train a dialog policy manager that can reason and decide the action to take based on the understanding of user intentions and conversation context. It is more challenging for a medical bot because of the need to integrate medical knowledge for reasoning and decision making. [Wei et al., 2018] proposed a reinforcement learning (RL) framework with multi-layer perceptron deep Q networks (MLP-DQN) [Mnih et al., 2013]. [Xu et al., 2019] extended the study by enhancing the DQN with hand-crafted features among diseases and symptoms, generated from the training set. However, both models cannot directly learn from real doctor-patient conversations. For the RL agent to fully explore the entire action space, the agent can only learn by interacting with rule-based user simulators, which can not learn from the dialog histories between real doctors and patients.

Another difficulty faced by the RL-based dialog manager is adapting a trained policy to new tasks (e.g., adapting trained medical bot to serve new diseases), since adaptation data is usually limited and hard to collect. On the other hand, Meta-learning algorithms are proposed to improve the model performance when training or adaptation data is limited. [Finn et al., 2017, Snell et al., 2017]. Since both many- and few-shot learning methods depend heavily on the quality of learned representations, these studies encouraged us to combine meta-learning and deep reinforcement learning models to improve the dialog agents for automatic diagnosis in both scenarios by learning better representations of dialog actions and domain knowledge.

In this chapter, we propose prototypical Q networks (ProtoQN), borrowing the ideas of prototypical networks [Snell et al., 2017] and matching networks [Vinyals et al., 2016]. We evaluate the model in the medical dialog domain since it is important and the medical conversations are a scarce resource. The model makes full use of real doctor-patient conversations by calculating prototype disease and symptom embeddings by encoding the dialog histories in the training set. Experimental results have shown that by learning a shared prototype embedding space, the ProtoQN outperforms MLP-DQN under both experiment settings. The experiments in this chapter focus on medical dialog to show the benefit of the proposed method, but we believe the conclusion can be generalized to other domains since we do not use handcrafted features or external domain-specific information.

4.2 Related Work

4.2.1 Deep Q Networks

The deep Q networks (DQNs) are proposed in [Mnih et al., 2013] for handling Atari video games. [Silver et al., 2016] proposed a deep reinforcement learning architecture for mastering the game of GO. In the area of dialog systems, the DQN is a popular model for building dialog managers and learning dialog policies [Zhao and Eskenazi, 2016, Yang et al., 2017, Lipton et al., 2016, Peng et al., 2017, Fazel-Zarandi et al., 2017].

The goal of the DQN [Mnih et al., 2013] is to estimate $q(s, a)$, the Q value of taking action a at state s. At each time step, a DQN-based agent selects an action with a ϵ-greedy strategy, i.e., epsilon % of the time selecting action with largest Q value given current state, and picking a random action for the rest. Meanwhile, the transition of the current step, $(S_t, A_t, R_t, \gamma, S_{t+1})$, is added to a memory buffer for future learning [Lin, 1992]. Here, S_t is the state at time t, A_t is the action taken at time t, R_t stands for the immediate reward at time step t, and γ is a discount rate. The objective function for training the neural network is

$$L = (R_t + \gamma max_{a'} q_{\bar{\theta}}(S_{t+1}, a') - q_\theta(S_t, A_t))^2$$

where θ stands for the parameters set of the current network and $\bar{\theta}$ is the parameters of the target network. The parameters are updated by stochastic gradient descent (SGD).

4.2.2 Spoken Dialog Systems

Spoken dialog systems aim at completing specific tasks [Papineni et al., 2001, Scheffler and Young, 2002, Young et al., 2010, Luo et al., 2019b] by interacting with users through natural language. Conventionally, a dialog manager is built to learn the dialog policy, which decides actions by reasoning from dialog state (the combined representation of user intentions and context). Dialog management is often formulated as a partially observable Markov decision process (POMDP), and solved as a reinforcement learning (RL) problem [Young et al., 2013]. As of late, many state-of-the-art dialog systems achieve satisfactory results by leveraging DQNs [Mnih et al., 2013, Silver et al., 2014], to learn policy and manage dialogs [Zhao and Eskenazi, 2016].

4.2.3 Meta-Learning

Recently meta-learning has gained attention among the machine learning field for improving model performance when little labeled training data is available. Model-Agnostic Meta-Learning [Finn et al., 2017] optimizes parameter initialization over multiple out-of-domain subtasks for the initialization to be generalizable in targeted tasks after fine-tuning on in-domain labels. Neural Turing machines [Graves et al., 2014] augment neural models with memory modules to improve performance in the limited-data regime. Metric-based meta learning, such as prototypical networks (ProtoNets) [Snell et al., 2017], siamese neural networks [Koch et al., 2015], matching networks [Vinyals et al., 2016], and structure induction models [Shen et al., 2017, Luo et al., 2019a] learn embedding or metric spaces such that the space can be adapted to domains unseen in the training set with only a few examples from the unseen domains. Meta-learning models have also been applied in dialog generation [Qian and Yu, 2019] for quick policy adaptation in different dialog domains. In this chapter, our proposed model is evaluated in the medical domain that requires not only dialog policy, but also multi-step reasoning with domain-specific knowledge.

4.3 Method

4.3.1 Dialog State Representations

Following the method proposed in [Wei et al., 2018] for vectorizing the dialog states, each dialog state consists of 4 parts:

I. UserAction: The user action of the previous dialog turn:

- **Request**: A user sends a self-report containing a set of explicit symptoms and request for diagnosis.
- **Confirm**: A user confirms the existence of an agent-inquired symptom.
- **Deny**: A user denies the existence of a symptom.
- **NotSure**: A user is not sure about the inquired symptom, which usually happens when an unrelated symptom is inquired.

II. AgentAction: The previous action of the dialog agent:

- **Initiate**: The agent initiates the dialog and asks the user to self-report.
- **Request**: The agent asks the user if a symptom exists.
- **Inform**: The system predicts and informs the user of the disease.

III. Slots: The set of symptoms that appear in the dialog history and their status. Each symptom has 4 possible status levels:

- **Confirmed**: The existence of the symptom is confirmed.The existence is denied by the user.
- **Unrelated**: The symptom is not necessary for the doctor to make an accurate diagnosis.
- **NotInquired**: The symptom has not been inquired about.

IV. NumTurns: The length of the dialog history, in other words, the current number of turns.

In each dialog turn, we represent UserAction, AgentAction, and NumTurns with one-hot vectors a^u, a^r, and n respectively. We use a 66-dimension vector s to represent the Slots, where each dimension indicates the status of a symptom. A confirmed, denied, unrelated, and not inquired symptom possesses values 1, -1, -2, and 0 in the corresponding dimension. The final input of the neural network at the t-th turn is represented as

$$s_t = [a_t^u, a_t^r, n_t, s_t] \tag{4.1}$$

4.3.2 Prototypical Q Networks

For modeling dialogs, we propose the prototypical Q networks (ProtoQNs) as well as corresponding training and evaluation pipelines, based on conventional DQNs and ProtoNets.

We first define the notation of our dataset as follows. The dataset S contains N doctor-patient conversations, $\{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$, where y_i stands for the disease label of the $i-$th training case, and x_i stands for the corresponding dialog history, where x_i can be further represented as

$$x_i = \{u_0^i = E^i, (a_1^i, u_1^i), \ldots, (a_k^i, u_k^i), a_{k+1}^i = D^i\} \tag{4.2}$$

Here E stands for explicit symptoms reported at the beginning of each dialog, a stands for agent follow-up inquiries, u stands for user responses, and D stands for the predicted disease.

The core of the ProtoNets [Snell et al., 2017] is calculating and updating the prototype embeddings of the output classes. The network classifies examples by comparing the input embedding with the prototypes and predicts the class with its prototype closest to the input embedding. As compared to other single-stage reasoning tasks, such as image classification or object detection [Snell et al., 2017], medical dialog requires multi-stage reasoning to infer dialog action through states. As a result, we propose a method for computing action prototypes via dialog state embedding. The calculation process of prototypical embeddings is shown in Figure 4-2

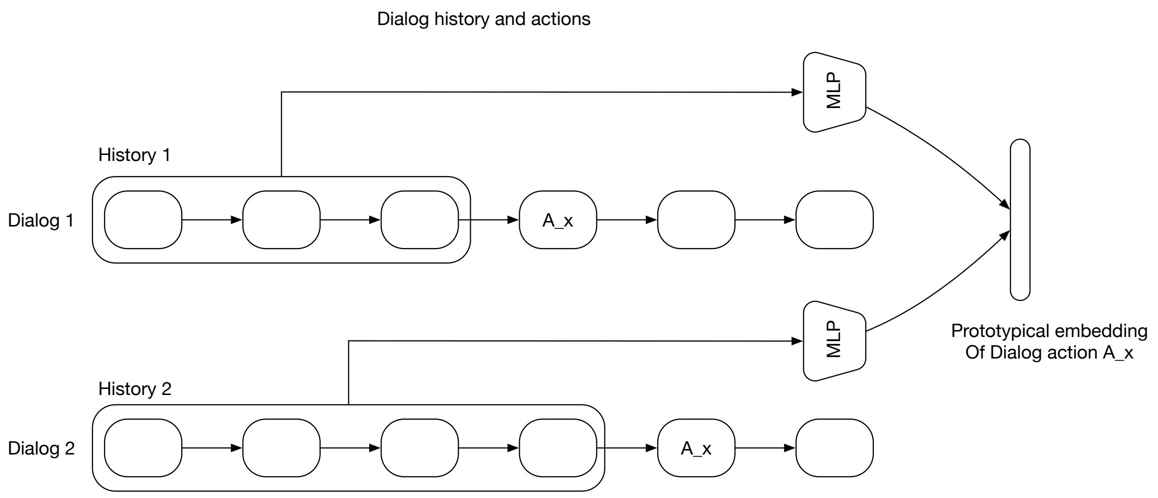


Figure 4-2: Calculating prototypical embeddings of dialog actions using the history of training dialogs.

Dialog state embedding

For each training conversation, x_i, as defined in Equation 4.2, the dialog state s_j^i at time step j can be represented as

$$s_j^i = \{u_0, (a_1^i, u_1^i), \ldots, (a_{j-1}^i, u_{j-1}^i)\} \tag{4.3}$$

s_j^i is then converted into a representation vector following Equation 4.1, denoted as f_{enc}, to obtain the state embedding, e_j^i. That is

$$e_j^i = f_{enc}(s_j^i) \tag{4.4}$$

With the approach described above, for any conversation we can get e_t, the embedding of the dialog state at time step t (i.e., s_t).

Dialog action prediction

At each step of a dialog, the dialog system is provided a dialog state encoding s_t in Equation 4.1. With the same encoder f_{enc} for embedding dialog states in the training set, we calculate the embedding of the input dialog state e_t in the new dialogs generated in both training and evaluation phases with Equation 4.4.

Then we can further compute prototypes and predict dialog action. First, with the state embedding e_t, the protoQNs calculate the Q value of the m-th dialog actions a_m by

$$q(a = a_m) = e_t \cdot P_m \tag{4.5}$$

where a_m is the embedding of the m-th dialog action, generated by mean-pooling a number of dialog states followed by a_m, and

$$P_m = \frac{\sum_{i,j \in D} v(a_j^i) \cdot 1(a_j^i = a_m)}{\sum_{i,j \in D} 1(a_j^i = a_m)} \tag{4.6}$$

Here, $1(\cdot)$ is the indicator function, and D is the set of examples used for computing prototypes. In training, D is a small subset of dialog states sampled from the training set followed by action a_m. In evaluation, D is the entire training set. In Algorithm 1, we show the calculation of prototype embeddings.

Algorithm 1 Calculating prototype embeddings

Function $protoEmbed(A, H)$

Inputs: dialog action set A, entire dialog history in the training corpus $H = \{(s_1, a_1), \dots, (s_m, a_m)\}$.

Outputs: Prototype embeddings of dialog actions P.

```
1: for $a' \in A$ do
2:     $A_{a'} = \{(s_k, a_k) \in H, a_k = a'\}$
3:     if Training then
4:         Sample D from $A_{a'}$ with size = 10
5:     else if Evaluating then
6:         D = $A_{a'}$
7:     end if
8:     Calculate $P_{a'}$ with Equation 4.6
9: end for
```

Disease prediction and training

Each dialog starts from an explicit symptom set provided by a user goal, and the model inquires about a set of symptoms before making the final disease prediction. For each inquiry, the user simulator replies based on the implicit symptom set as described in Section 4.3.1. The conversation stops when the systems output a disease prediction. Summarizing the previous sections and descriptions, we provide the complete procedure of a medical dialog in Algorithm 2.

For each simulated dialog described above, the model sees a success or failure reward when it informs the user of a predicted disease. The ProtoQN updates its weights based on the reward with stochastic gradient descent (SGD) following the standard pipeline of training a DQN applied in [Wei et al., 2018, Xu et al., 2019]. For evaluation, the ProtoQN generates prototype embeddings only once before the evaluation begins with all real dialog histories in the training corpus.

Algorithm 2 Automatic diagnosis dialog process

Inputs: Explicit symptom set E, initial turn Id $t = 1$, empty implicit symptom set $I = \{\}$, dialog history for training H, encoder f_{enc}

Outputs: Final disease prediction D.

```
1: $P = protoEmbed(A, H)$
2: while Dialog not end do
3:     $s_t = State(E, I, t)$ with Equation 4.1
4:     $h_t = f_{enc}(s_t)$
5:     $a_t =_a h_t \cdot P_a^T$ with Equation 4.5
6:     if $a \in$ Symptom then
7:         $u_t =$ user response
8:         $I = I + (a_t, u_t)$
9:         $t = t + 1$
10:    else if $a \in$ Disease then
11:        $D = a_t$
12:        $endDialog()$
13:    end if
14: end while
```

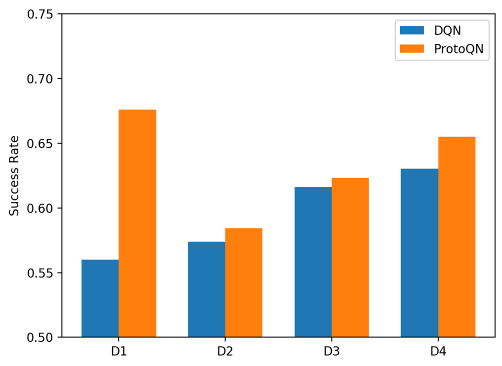
4.4 Experiments

4.4.1 Data and Experiment Settings

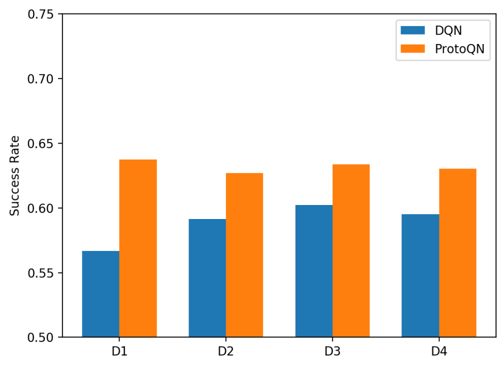
We evaluate the dialog models on Muzhi dataset [Wei et al., 2018]. The dataset contains 710 medical dialogs between real doctors and patients, and each is annotated as a user goal, covering 4 different diseases and 66 symptoms. For our experiments, we apply the official train-test split. 568 user goals are used for training and the other 142 are used for evaluation.

We apply a simulator for providing user responses in conversations. To simulate speech noise and mistakes led by user knowledge biases, we apply intent and slot noise to the simulator [Xu et al., 2019] in order to evaluate the model performances under different levels of noise. In our experiments, we apply 0%, 10%, 20%, and 30% error rates, i.e., the probability that the status of an inquired symptom is sampled at random rather than based on annotation.

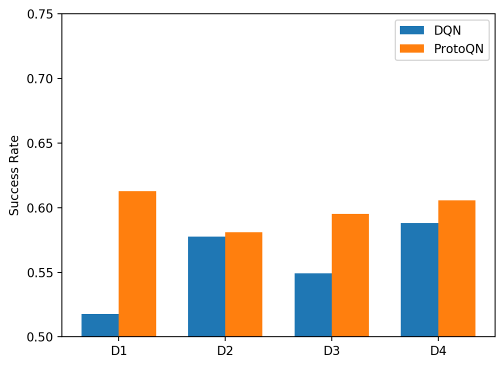
We also conduct two groups of experiments for evaluating model performance under various conditions. First, we train both ProtoQN and DQN on the entire training set. This is conventional with a fully supervised learning setting. Second,



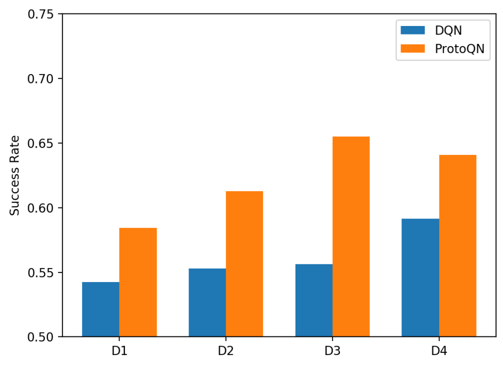
(a) Noise = 0.0



(b) Noise = 0.1



(c) Noise = 0.2



(d) Noise = 0.3

Figure 4-3: Performances of ProtoQN and DQN on 4 noise levels. Each figure stands for noise, while the blue bars stand for the performances of DQNs, and the orange bars stand for the performances of ProtoQNs.

we adopt a meta-learning-like setting to evaluate the model performance at few-shot learning. We pre-train the models with three diseases and fine-tune the models with randomly selected training samples from the trained diseases plus a few cases of the new disease. In both learning tasks, the neural models are evaluated on the entire public test set. For our experiments we set the success reward of ProtoQN to 20, the failure reward to 0, and the maximum number of turns is 44. For DQN, we keep the settings in [Wei et al., 2018].

4.4.2 Fully Supervised Learning

We first compare the ProtoQN with DQN in the normal supervised learning setting to evaluate their ability to learn dialog policy with enough training data. Experimental results are shown in Table 4.1. We obtain the DQN performance directly from [Wei et al., 2018].

Table 4.1: Experimental results of ProtoQN and DQN on supervised learning.

Models	Success Rate (%)	Reward
DQN	65	20.51
ProtoQN	70.42	23.58

Experiments show that the ProtoQN significantly outperforms the DQN baseline, without adding external knowledge and hand-crafted features [Xu et al., 2019]. The improvement shows that utilizing dialog history between real doctors and patients to calculate prototype embeddings is effective and provides a better ability for the model to learn the dialog policy. While DQN learns Q-values indirectly from simulated conversations, ProtoQN directly relates Q-values with dialog actions and states in real conversation.

4.4.3 Few-shot New Disease adaptation

We next investigate model performance in the situation when a new disease appears after the model was trained, and only a few examples are available for training. To evaluate this situation, we adopt an experimental setting popular in few-shot learning, where we first pre-train the models on the training set for a subset of diseases. Then the models are fine-tuned (i.e., adapted) on a small number, N, of examples for a new disease. To prevent catastrophic forgetting [Riemer et al., 2018], we also randomly select N samples from each pre-trained disease to compose the adaptation set. Since the Muzhi corpus includes 4 diseases, we conduct 4 iterations of evaluation, with each corresponding to one of the 4 diseases as the new disease, and the remaining as the pre-trained ones. After fine-tuning, the models are also evaluated with the public test

Table 4.2: Average success rates (%) of the meta-learning tasks with DQN and ProtoQN under different noise levels.

Noise	0	0.1	0.2	0.3
DQN	59.51	58.89	55.81	56.07
ProtoQN	63.47	63.21	59.85	62.32

set. Also, we consider different noise levels in this task. Our purpose is to evaluate how well the models learn new diseases with a few examples without forgetting knowledge for the pre-trained diseases.

The experiment results of the new disease adaptation are shown in Table 4.2. From the table we see that the ProtoQN significantly outperforms the DQN model and achieves SOTA performance under a few-shot learning setting. This result shows that learning shared embeddings from real conversations is more efficient when adapting the model to new diseases with few examples. Meanwhile, much-shared knowledge from pre-trained diseases can still be applied to the new ones. The learned knowledge about symptoms allows the neural network to learn new diseases with a better initialization of dialog policy and thus adapts faster and better. We also found that although the increase in noise level degrades the performance of DQNs, ProtoQN yields a steady success rate when the noise level varies. we believe the robustness results from the ensemble nature of ProtoNet's inference mechanisms.

4.5 Chapter Summary

In this chapter, we propose a novel dialog management model, prototypical Q network, for supervised and few-shot dialog policy learning. We apply this model in the area of automatic conversational diagnosis. Experiments show that the ProtoQN outperforms the DQN model in both supervised and few-shot settings. In the supervised setting, ProtoQNs achieve results comparable to SOTA without using domain-specific features. As for the few-shot experiment, ProtoQN learns new diseases under the few-shot training setting without forgetting previously learned diseases, and achieves SOTA. The model also shows less degradation as we inject noise into a conversation. Our study suggests that modeling real conversations directly reinforces simulator-based dialog policy learning. Embeddings of dialog actions are shareable among tasks (diseases, in our case) and benefit the fast adaptation to new ones. Here we show promising results in the medical domain. Future work could investigate more adaptive models as well as different domains and corpora toward the goal of modeling new dialog tasks better and with fewer examples.

Chapter 5

Self-trained Prompt Composition for Domain Adaptation in Question Answering

Recent studies have found fine-tuning large-scale pretrained language models an effective method for natural language processing (NLP) tasks [Radford et al., 2019a, Devlin et al., 2018, Liu et al., 2019, Clark et al., 2020]. However, a key limitation of the fine-tuning method is its space cost: in practice, a different fine-tuned model needs to be saved for each task or domain.

An alternative paradigm, *prompt*-based learning, has been found to effectively reduce the space cost [Radford et al., 2019b, Brown et al., 2020a, Liu et al., 2021]. In these methods, an NLP task is first converted to a generation task, and a sequence of hand-designed prompt tokens are used as a prefix to guide a language model to complete the task. [Li and Liang, 2021] and [Lester et al., 2021] further discovered that, instead of using hand-designed prompt tokens, an alternative *soft prompt tuning* method, where randomly initialized prompt embeddings for a task are fine-tuned on a supervised dataset with the model frozen, can achieve comparable performance with full model fine-tuning. In [Lester et al., 2021]'s approach, the space cost is reduced from $O(N \cdot M)$ for model fine-tuning to $O(N \cdot m + M)$, where N is the number of tasks, M the size of the entire model and m the size of a set of prompt embeddings for one task.

More importantly, [Lester et al., 2021] found that soft prompt tuning enables better domain adaptation: when the prompt is tuned on a source domain, task performance on target domains consistently improves compared to model fine-tuning on the same data. Yet, a key limitation of their method is that no domain information is leveraged during either training or inference. This neglects the large amount of unsupervised data that one can often easily access for a target domain.

In this chapter, we address this issue and improve domain adaptation by designing a framework where we learn target domain representations via a self-supervised objective during prompt tuning, and leverage the learned domain representations for inference.[1] Specifically, we propose a *prompt composition* method, where we divide a set of learnable prompt tokens into two groups, *task* prompts and *domain* prompts. We tune the prompts for an NLP task with annotated data on a source domain, together with an unsupervised span completion objective on unannotated data from both source and target domains. At test time, we compose the learned NLP task prompt with a target domain prompt for inference. Our goal is to decouple the learning of task information and domain information into different groups of prompt tokens. While our method is task-agnostic, we focus our evaluation on the question answering (QA) task, a main task that has shown to benefit from prompt tuning. On the MRQA benchmark, we show that our method leads to notable improvement on all 6 out-of-domain test sets over the baseline method by [Lester et al., 2021]. With a T5-Base model, our method shows an average gain of +2.1 F_1 on all datasets, with the most gain of +7.4 F_1 observed for TextbookQA. Our findings open new doors to exploring the compositionality of prompt tokens for NLP tasks.

[1]Based in part on the paper "Prompt Composition for Improved Domain Adaptation in Question Answering," by H Luo, Y Zhang, B Athiwaratkun, X Ma, C Nogueira dos Santos, A Arnold, B Xiang, ACL Rolling Review submission, 2022.

5.1 Method

We first briefly overview the soft prompt tuning method by [Lester et al., 2021], then extend this method by designing prompt composition for individual tasks and domains, and introduce a mixed-task, mixed-domain recipe for prompt tuning.

5.1.1 Background: Soft Prompt Tuning

In [Lester et al., 2021], all tasks (such as QA) are cast as a text generation task, and can be viewed as $y = f_{\text{gen}}(x)$, where f_{gen} represents a "text-to-text" generative model, x the input tokens, and y the output tokens. While the conventional way of adapting f_{gen} for different tasks or domains is to fine-tune its entire parameters θ for a target task/domain, [Lester et al., 2021] instead proposes to fine-tune the embeddings of a set of prompt tokens p parameterized by θ_p while keeping θ fixed, and model the task as:

$$y = f_{\text{gen}}([p; x]), \tag{5.1}$$

where p is prepended to the original input sequence x. They showed that this "soft" prompt tuning approach outperforms the use of manually designed prompts; furthermore, it approaches the results of full model tuning for each task when a sufficiently large f_{gen} is used, while only needing to learn a fraction of the parameters for each task (θ_p vs. θ).

5.1.2 Prompt Composition

Despite the impressive results, a key limitation of this method in zero-shot domain adaptation is that only a single set of prompt tokens (p) can be learned for the task and no domain information can be leveraged for inference in different domains.

To address this issue, we propose a new method where we divide all prompt tokens into two groups, *task* prompts and *domain* prompts, and concatenate them for prediction:

$$y = f_{\text{gen}}([p_t; p_d; x]), \tag{5.2}$$

where p_t represents a task prompt, p_d a domain prompt. As we will show, this compositional design has two key advantages: 1) it allows us to design a training recipe that decouples the learning of task and domain information; 2) for inference, it allows us to compose a new (task, domain) prompt pair, which improves out-of-domain performance.

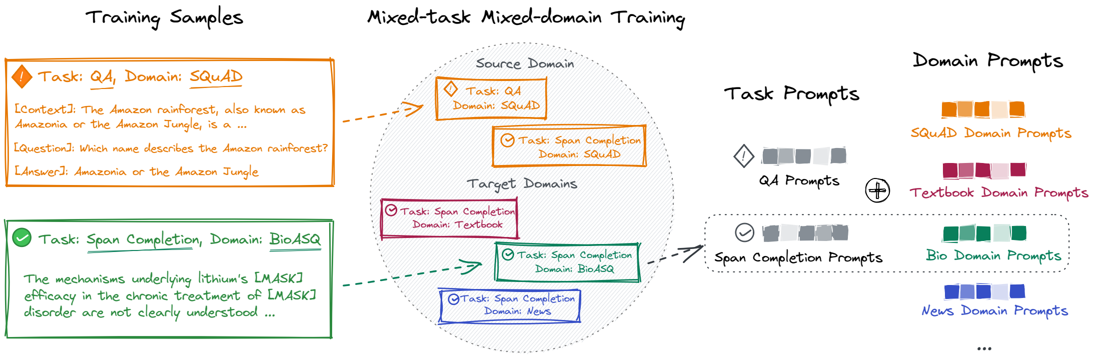

Training Samples
Mixed-task Mixed-domain Training
Task: QA, Domain: SQuAD
[Context]: The Amazon rainforest, also known as Amazonia or the Amazon Jungle, is a ...
[Question]: Which name describes the Amazon rainforest?
[Answer]: Amazonia or the Amazon Jungle
Task: Span Completion, Domain: BioASQ
The mechanisms underlying lithium's [MASK] efficacy in the chronic treatment of [MASK] disorder are not clearly understood ...
Source Domain
Task: QA Domain: SQuAD
Task: Span Completion Domain: SQuAD
Target Domains
Task: Span Completion Domain: Textbook
Task: Span Completion Domain: BioASQ
Task: Span Completion Domain: News
Task Prompts
QA Prompts
Span Completion Prompts
Domain Prompts
SQuAD Domain Prompts
Textbook Domain Prompts
Bio Domain Prompts
News Domain Prompts
...

Figure 5-1: Overview of our mixed-task, mixed-domain prompt tuning pipeline.

5.1.3 Mixed-task & Mixed-domain Training

We are interested in the unsupervised domain adaptation of question answering systems. In this setup, we assume access to a source domain $\mathcal{D}_s$, where we have human annotations of QA pairs for both training and testing; and a set of target domains $\mathcal{D}_t$, where for each domain we only have access to a small test set of human-annotated QA pairs, as well as unannotated text sampled from this domain. Our goal is to learn a QA model and evaluate its performance on the unseen test sets from $\mathcal{D}_t$.

The baseline prompt-tuning method in Equation 5.1 trains the prompt embeddings θ_p with QA examples from $\mathcal{D}_s$ and evaluates the model directly on $\mathcal{D}_t$. However, this method does not leverage any information from the target domain, which is often easily accessible in practice. In contrast, following Equation 5.2, our goal is to encode information about a target domain $j \in \mathcal{D}_t$ into its domain prompt p_d^j in an unsupervised manner, and effectively leverage this domain information at inference time via p_d^j.

To do this, apart from the original QA task for which we only have annotations from $\mathcal{D}_s$, we introduce a self-supervised objective to help us learn domain prompts for all domains in $\mathcal{D}_s$ and $\mathcal{D}_t$. For simplicity, we adopt the masked span completion objective from [Raffel et al., 2019]. Given the input text, we randomly mask a span from the text, and train the model to generate the masked span as:

$$s = f_{\text{gen}}([p_t^{\texttt{span}}; p_d; \texttt{mask}(x, s)]), \tag{5.3}$$

where $\texttt{mask}(x, s)$ is a function that replaces span s in context x with a [$\texttt{mask}$] token, and $p_t^{\texttt{span}}$ is a set of span completion prompt tokens. Similarly, for the main QA task, we train it with $y = f_{\text{gen}}([p_t^{\texttt{QA}}; p_d; x])$, where x is the concatenation of input context and question tokens, y the answer tokens, and $p_t^{\texttt{QA}}$ the QA prompt tokens.

A naive approach of learning all prompt tokens is to run the training sequentially for both the span completion and QA tasks on all domains where examples are available. However, this naive approach is problematic because it does not encourage decoupled encoding of task and domain information in different prompts. Instead, we find a mixed-task, mixed-domain training recipe to be useful (see Figure 7-6): during training, we mix and shuffle all training examples from the supervised and self-supervised tasks, and at each step randomly sample a mini-batch of examples from the mixed pool. Formally, for each training example we have

$$y = f_{\text{gen}}([p_t^i, p_d^j, x]), \tag{5.4}$$

where $i \in \{\texttt{span}, \texttt{QA}\}$; for $i = \texttt{span}$, we set $j \in \mathcal{D}_s \cup \mathcal{D}_t$; for $i = \texttt{QA}$, we set $j \in \mathcal{D}_s$.

At test time, for inference on domain j, we concatenate the learned QA prompt $p_t^{\texttt{QA}}$ and the domain prompt p_d^j, and calculate the output with Eq. 5.2.

We further find two other techniques helpful in our experiments, which we document below.

Answer Span Completion. For span completion, instead of using a naive span masking strategy where the span is drawn completely randomly, we found it more effective to select meaningful spans that may benefit the QA task. Thus, we employ the answer entity recognition (AER) model to tag spans in the input text that are

Model	Metric	Baseline	Our Model	w/o Mix	w/o AER	w/o Combine
BioASQ	EM	49.8	**51.0**	49.2	48.8	50.3
	F1	66.8	**68.2**	66.5	65.9	67.4
TextbookQA	EM	16.4	**18.6**	15.7	15.5	17.6
	F1	32.3	**39.7**	31.4	30.6	38.1
RACE	EM	13.0	**14.5**	14.5	12.1	14.2
	F1	46.4	**48.8**	48.8	45.7	48.1
RelExt.	EM	73.3	**73.8**	73.8	71.6	73.5
	F1	84.6	**84.8**	84.8	82.3	84.7
DuoRC	EM	43.6	**44.0**	43.6	42.0	43.7
	F1	60.3	**60.7**	60.1	59.2	60.7
DROP	EM	29.3	**30.0**	28.2	26.4	27.1
	F1	39.2	**39.8**	37.2	35.4	37.6
Average	EM	37.6	**38.6**	37.0	36.1	37.7
	F1	54.9	**57.0**	54.3	53.2	56.1

Table 5.1: Results of different prompt tuning strategies with the T5-Base model on the MRQA out-of-domain datasets. The baseline is the soft prompt tuning model proposed in [Lester et al., 2021]

likely to be answers, and compose the span completion task by randomly drawing from these spans. The AER model is trained on plain texts without questions to recognize potential answers.

Domain Prompt Combination. For evaluation on a target domain $j \in \mathcal{D}_t$, we found it useful to further combine the learned domain prompt p_d^j with the source domain prompt via a simple averaging operation as $\bar{p}_d^j = (p_d^j + p_d^s)/2$, where p_d^s represents the domain prompts for the source domain. We then leverage $\bar{p}_d^j$ for prediction and see improved results. We conjecture that this is because only the source domain prompt p_d^s has seen the QA task prompt at training time, and therefore averaging it with p_d^j improves stability for the QA task.

5.2 Experiments

We focus our evaluation on domain adaptation for the QA task. We compare mainly to the soft prompt tuning method by [Lester et al., 2021] as our baseline, given that

Models (T5-Large)	**BioASQ**	**TQA**	**RE**	**Avg.**
Baseline	71.1	61.9	86.5	65.0
Our model	**74.3**	**66.7**	**87.4**	**66.8**
– w/o Mix	70.1	60.0	83.8	62.5
– w/o AER	68.3	58.7	82.3	61.4
– w/o Combine	72.7	62.3	87.1	65.6

Table 5.2: Answer F_1 results on MRQA with T5-Large model. "TQA" represents TextbookQA; "Avg." represents average results from all 6 test datasets.

our method is built on top of theirs.

Following [Lester et al., 2021], we use SQuAD v1.1 [Rajpurkar et al., 2016] as the source domain dataset, and evaluate on the out-of-domain datasets from the MRQA benchmark [Fisch et al., 2019]. The SQuAD dataset is annotated with text from Wikipedia, and the evaluation benchmark of MRQA contains data from 6 different domains, including BioASQ [Tsatsaronis et al., 2012], TextbookQA [Kembhavi et al., 2017], RACE [Lai et al., 2017], RE [Levy et al., 2017], DuoRC [Saha et al., 2018], and DROP [Dua et al., 2019], covering the domains of biomedicine, books, exam, relation extraction, movie and multi-step reasoning.

For main experiments, we select the T5-Base and T5-Large models [Raffel et al., 2019] as our pretrained generator (f_{gen}). We did not train larger T5 models due to computational constraints. For soft prompt tuning baseline, we follow [Lester et al., 2021] and tune 100 prompt tokens. For our model, we use 50 tokens for task prompt and 50 for domain prompt. For the training of the span completion objective, we collect training passages from the target domain datasets but discard the annotated QA pairs (thus no annotation is exposed), and train the span completion prompt and domain prompts with the unannotated passages.

For all experiments, we use the AdaFactor optimizer [Shazeer and Stern, 2018], and the learning rate is set to 0.3, which achieves the highest performance for the baseline model. For the initialization of the prompt embeddings, we calculate the mean and standard deviation of all token embeddings of the pretrained T5 models, and randomly initialize the prompt embeddings from a Gaussian distribution accordingly.

To construct the training dataset, we collect the training set of each corpus from the MRQA test domains and sample 2,000 passages for each domain for balanced training. For target domains, we only use the passages for training and discard all the annotated QA pairs. We also sample 2,000 passages from SQuAD for training the SQuAD domain prompts with the span completion objective. For each passage, we sample 10 spans for the span completion task. Together with the SQuAD training set, our new training set contains 227,599 training examples, including 87,599 QA examples and 140,000 span completion examples. In our full model where the AER model is used for span selection, we select the top-10 answer spans predicted by the AER model. In our ablation study where the AER module is not used, we randomly sample span lengths and span start index for the training spans. Our training ends when all examples, including QA and span completion examples, are run for 2 epochs. The learning rate is fixed (0.3) during the entire training process.

5.3 Results and Analysis

We present our main results on the MRQA datasets with the T5-Base model in Table 5.1. Results from both the baseline model and our model are averaged across 3 independent training runs. We find that our full model which leverages prompt composition outperforms the baseline soft prompt tuning approach on test sets from all 6 domains. The largest gains are observed on the TextbookQA dataset, where using our prompt composition leads to a remarkable improvement of +7.4 on the answer F_1 score. When we average the results over all 6 out-of-domain test sets, a notable improvement of +2.1 is observed for the answer F_1 score. The lowest improvement is observed for the RE dataset, with a gain of +0.2 F_1 score. We conjecture that the RE task cannot benefit much more from prompt composition over SQuAD, because 1) the text is from the same Wiki domain as the SQuAD dataset, and 2) the style of questions in RE is similar to SQuAD questions.

To make sure that our findings generalize to larger models, we further verify the results on the T5-Large model, which contains 770M parameters. We present answer F_1 results on selected MRQA datasets in Table 5.2. We observe very similar trends:

the largest gain of +4.8 F_1 is observed on the TextbookQA dataset, and a notable +1.8 improvement is observed on average across all 6 datasets. Furthermore, we find that our T5-Large results with prompt composition on some datasets are comparable to the T5-11B results from [Lester et al., 2021]: 66.7 F_1 on TextbookQA vs. their 66.8 F_1 from T5-11B; 87.4 F_1 on RE vs. their 88.8 F_1 from T5-11B. These results are achieved with a 14x reduction in model size.

Comparison with Sequential Training. As mentioned in Section 5.1.3, we find that our proposed mixed-task, mixed-domain training is key for the success of prompt composition. To demonstrate this, we compare our results to a sequential training recipe, where we 1) run span completion training with the span prompt on each of the domains with its corresponding domain prompt; 2) run QA training with the QA prompt and SQuAD domain prompt on SQuAD, with the domain prompt frozen; and 3) combine the QA prompt with each target domain prompt for prediction. We present its results in Tables 5.1 and 5.2 (w/o Mix). We find that for both models and all datasets, the performance decreases substantially without the mixed training. Further, the average results are even lower than the baseline, highlighting the importance of the mixed training recipe for prompt composition.

Ablation Studies. We further ablate our full model by removing the answer span recognition component or the domain prompt combination component and present the results in the tables. Without answer span recognition (w/o AER), we see a substantial drop of performance across all the datasets. We conjecture that this is because the answer span recognition component helps us learn domain prompts in a way that more closely resembles the QA task. Similarly, we observe a mild drop of performance for all the datasets when we remove domain prompt combination (w/o Combine).

Performance on Source Domain. We examine how the QA performance on the source domain, SQuAD, is impacted by prompt composition, by applying the learned QA prompt with the SQuAD domain prompt on the SQuAD test set. For T5-Base, we found that the baseline prompt tuning achieved an average F_1 of 90.6 on SQuAD, while our method achieved an average F_1 of 90.8, showing no significant change in performance.

5.4 Chapter Summary

In this chapter, we present prompt composition, which allows us to compose task-specific and domain-specific prompts for the improved out-of-domain performance of QA models. Experiments show that the self-supervised pretraining of the domain and task prompt embedding can improve the domain adaptation ability of language models. Our method opens new ways to explore the compositionality of prompt tokens for improved NLP capabilities.

Chapter 6

Cooperative Self-training of Machine Reading Comprehension

6.1 Introduction

Recent studies have shown that language model pretraining provides high-quality text representations and significantly improves neural networks' performance on a variety of natural language processing (NLP) tasks [Peters et al., 2018a]. Based on the popular Transformer architecture [Vaswani et al., 2017], various language models have been proposed [Devlin et al., 2018, Liu et al., 2019, Clark et al., 2020]. These models are pretrained to predict a masked word in a given context from large corpora, and generate a contextual representation that encodes semantic and syntactic information. After finetuning, these representations significantly improve performance on downstream NLP tasks. Although masked language modeling is a powerful self-supervised learning technique, annotation on large-scaled data is still necessary for finetuning on difficult downstream tasks, including extractive question answering (QA)[1] where a large number of labeled question-answer pairs are required as a training corpora.

Previous studies showed that the QA models can be improved by training on synthetic question-answer pairs, namely self-training [Sachan and Xing, 2018, Puri et al., 2020,

[1]Also referred to as machine reading comprehension. The two terms are used interchangeably in this paper.

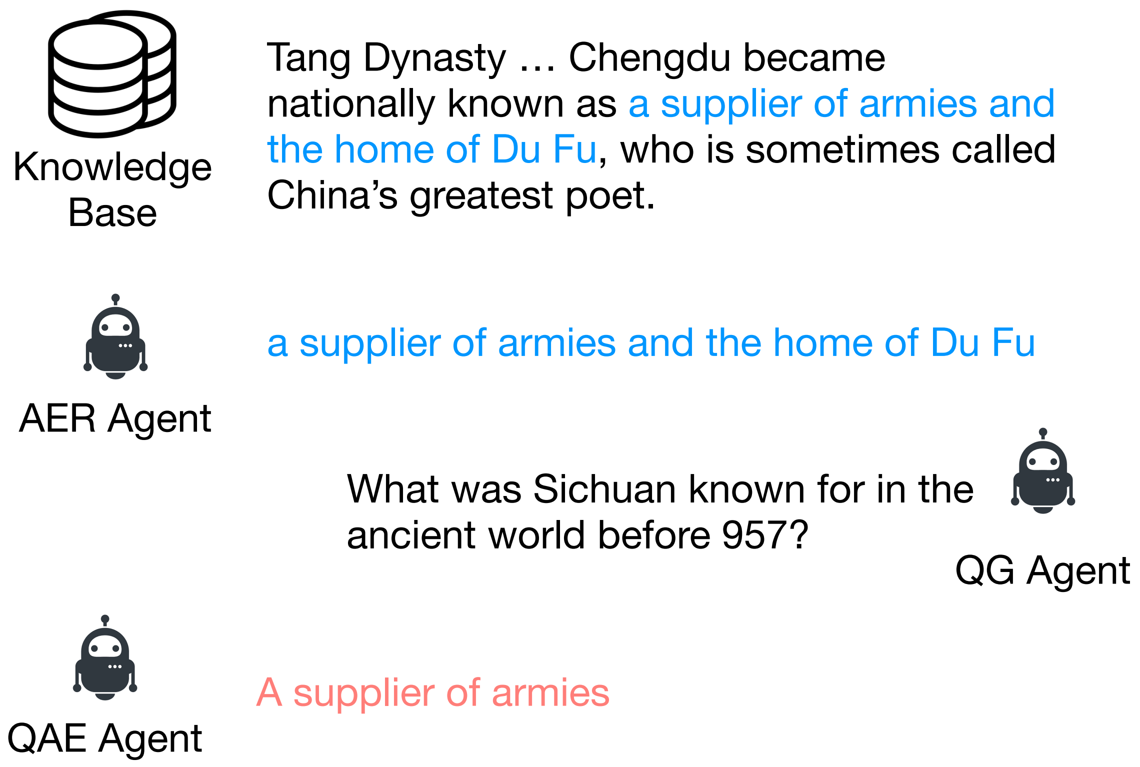

Knowledge Base
Tang Dynasty … Chengdu became nationally known as a supplier of armies and the home of Du Fu, who is sometimes called China's greatest poet.
AER Agent
a supplier of armies and the home of Du Fu
What was Sichuan known for in the ancient world before 957?
QG Agent
QAE Agent
A supplier of armies

Figure 6-1: The pipeline of semi-supervised question answering (machine reading comprehension) by RGX. The answer entity **R**ecognition agent recognizes answer entity from a given passage; the question **G**enerator outputs a question based on the passage and entity; the question-answering e**X**tractor predicts answers from the question and passage.

Shakeri et al., 2020, Bartolo et al., 2021]. The central aspect of these works is pretraining a question-answer pair synthesis model on a seed corpus, and applying the generator on target domains to obtain synthetic training data. The QA model learns domain knowledge after finetuning on the synthetic data, and thus domain adaptation is improved. However, the gap between the pretraining (i.e., seed) and the target corpus still exists, in terms of domain knowledge, question difficulty, and language style. The gap affects the quality of the synthetic training data.

In this chapter, we propose a framework that allows cooperative self-training for both QA pair synthesis and question answering to better adapt the synthesis models to the target domain and improve the learning of the QA models, as shown in Figure 6-2.[2]

[2]Based in part on the paper "Cooperative Self-training of Machine Reading Comprehension," by H Luo, SW Li, M Gao, S Yu, and J Glass, to appear NAACL, 2022.

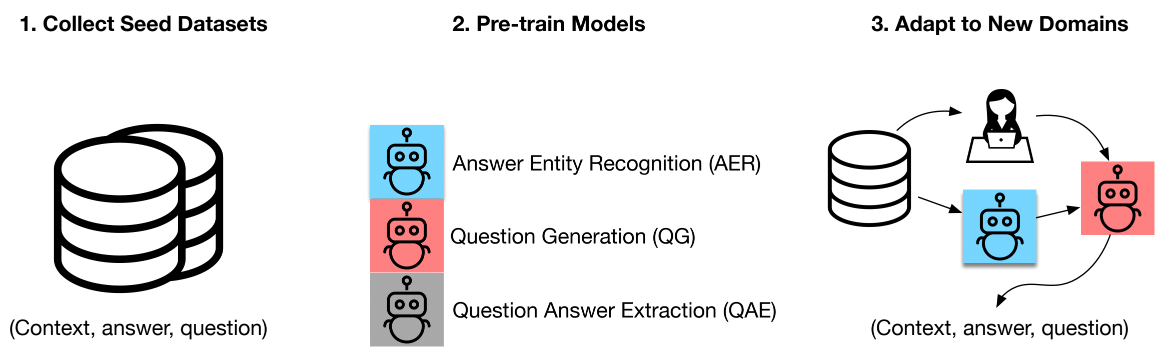

1. Collect Seed Datasets
2. Pre-train Models
3. Adapt to New Domains
Answer Entity Recognition (AER)
Question Generation (QG)
Question Answer Extraction (QAE)
(Context, answer, question)
(Context, answer, question)

Figure 6-2: Overview of the self-training QA framework.

In the framework, we construct a cooperative environment where a question generator and an answer extractor work together to solve a masked entity prediction problem. We first leverage an entity recognizer to mask out an entity in a provided passage. The question generator then outputs a question based on the masked passage. With the generated question and the original, unmasked passage, we train the answer extractor to select the correct answer spans, which are the masked entity. The extractor is also the final model used for extractive QA. To extract the spans accurately, the generator has to provide a good question, and the extractor should select the most likely tokens. We design the reward function such that it favors the questions leading to correct answers. We also gradually increase the difficulty of generated questions [Karpukhin et al., 2020a] by rewarding the questions that are not answered correctly but with low extraction losses via a stochastic expectation-maximization technique. The technique allows us to train the extractor with challenging examples incrementally. We call our algorithm RGX since it incorporates an answer entity **R**ecognizer, a question **G**enerator, and an answer e**X**tractor. The RGX pipeline is illustrated in Figure 6-6.

With RGX, we can train a QA model for any unlabeled target domain given the corresponding text corpora and a labeled QA corpus in a seed domain (either the same or different from the target). We show that RGX outperforms SOTA approaches in QA benchmark datasets when domain specific human labels are not available during finetuning. In this work, we make the following contributions:

1. We propose a cooperative self-training framework, RGX, which contains an

answer entity recognition, question generation, and answer span extraction to automatically generate non-trivial QA pairs on unlabeled corpora.

2. We design a expectation-maximization synthetic QA selection that identifies difficult but answerable questions without supervision to incrementally train the QA model with challenging examples, and a AER-based maximum mutual information inference method for question answering.

3. Experiments show that our method significantly outperforms SOTA pretrained QA models and self-training QA baselines.

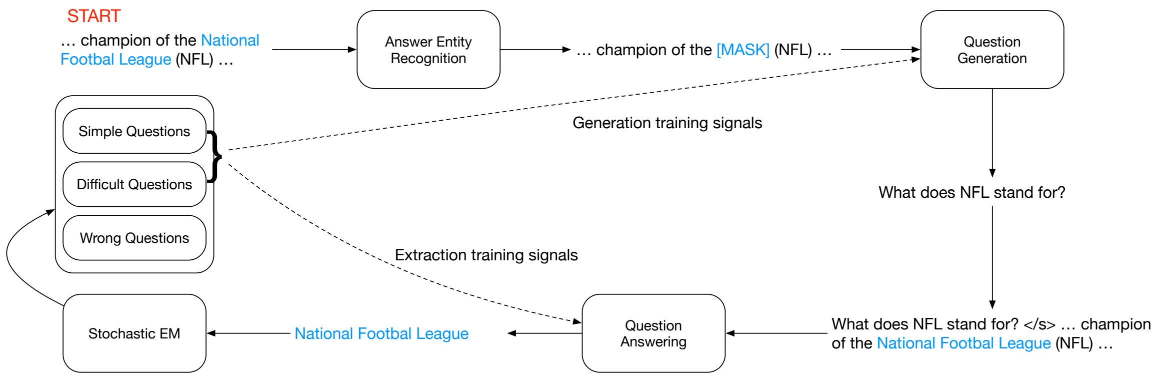

START
... champion of the National Footbal League (NFL) ...
Answer Entity Recognition
... champion of the [MASK] (NFL) ...
Question Generation
Simple Questions
Difficult Questions
Wrong Questions
Generation training signals
What does NFL stand for?
Extraction training signals
Stochastic EM
National Footbal League
Question Answering
What does NFL stand for? </s> ... champion of the National Footbal League (NFL) ...

Figure 6-3: The cooperative learning pipeline for question answering. The pipeline starts from a passage and follows the steps: (1) recognizing a potential answer entity, (2) generating a question asking about the answer entity, and (3) answering the question by extracting the answer span in the passage.

6.2 Related Work

Reinforcement learning and self-training have emerged recently for learning language generation in addition to maximum likelihood training. To optimize text generation models directly with non-differentiable objective functions, [Rennie et al., 2017] proposed self-critical sequence training (SCST) using a policy gradient [Kakade, 2001, Silver et al., 2014]. On the other hand, self-training has been shown to be effective in many tasks, such as machine translation [He et al., 2019], image classification [Xie et al., 2020], and structured database-grounded question answering [Xu et al., 2020].

In the domain of question answering, a question generator can be used for joint answer prediction [Tang et al., 2017, Duan et al., 2017], and synthetic QA data are used for in-domain data augmentation [Sachan and Xing, 2018, Puri et al., 2020, Liu et al., 2020, Klein and Nabi, 2019] and out-of-domain adaptation. [Lewis et al., 2019b] and [Lee et al., 2020] introduced models for question answering under unsupervised/zero-shot settings. [Shakeri et al., 2020] proposed generating synthetic question-answer pairs with an end-to-end model simultaneously. [Bartolo et al., 2021] improved the question synthesis by training with difficult QA cases from the AdversarialQA corpus [Bartolo et al., 2020] and fine-grained answer synthesis by multi-model voting.

In this work, we mainly compare our method with latest baselines, [Shakeri et al., 2020] and [Bartolo et al., 2021] that reported results on out-of-domain adaptation. Besides improved QA performance, our framework, RGX, differs from the previous work in the following aspects: (1) the method features reinforced finetuning of the QA Synthesizer, (2) the framework supports and improves maximize mutual information inference in test time, and (3) the method does not require complicated data annotation, e.g. AdversarialQA.

Representation learning has been an important topic in NLP area since neural language models were proposed [Bengio et al., 2003]. Based on word co-occurrence, [Mikolov et al., 2013b] and [Pennington et al., 2014] proposed language embedding algorithms to model word-level semantics. Recent studies have focused on pretraining contextualized word representations with large-scaled corpora [Peters et al., 2018a]. State-of-the-art representation models are pretrained with the masked language modeling task [Devlin et al., 2018, Liu et al., 2019, Clark et al., 2020] using the Transformer architecture [Vaswani et al., 2017].

Different variants of masked language models have been investigated to improve performance in downstream tasks. [Joshi et al., 2020] leveraged a masked span generation task instead of word prediction. [Fei et al., 2020] and [Shen et al., 2020] proposed models that learns better syntax knowledge with syntactic distances [Shen et al., 2018] and heights [Luo et al., 2019a]. [Henderson et al., 2019a] and [Humeau et al., 2019a] showed that pretraining language models on dialog corpora perform better on dialog-related downstream tasks, as compared to pretraining on Wikipedia. A span selection pretraining objective is proposed in [Glass et al., 2019] to reduce the gap between the pretraining and downstream finetuning stages and to improve the performance on the QA task. Some applications of generated questions are shown in [Lewis et al., 2021, Jia et al., 2021].

In contrast to self-training methods that usually adopt a teacher-student learning strategy, cooperative learning pipelines contain several agents working together to learn as much knowledge as possible. A typical cooperative learning framework is generative adversarial networks (GAN) [Goodfellow, 2016, Goodfellow et al., 2014], where a generator is optimized to confuse a discriminator, and a discriminator is trained to distinguish real examples from generated ones. Sequence GAN is further designed for learning diverse text generation [Yu et al., 2017]. Unlike the adversarial learning method where two networks work for opposite goals, other studies proposed learning environments in which different agents learn the same objective functions for language emergence [Lazaridou et al., 2016, Mordatch and Abbeel, 2018, Havrylov and Titov, 2017], including simple natural language, compositional language, and symbolic language.

6.3 RGX Framework

In this section, we first introduce (1) the QA synthesis pipeline, (2) cooperative self-training for both QA synthesis and question answering, and (3) an improved maximum mutual information inference strategy. The self-training pipeline of RGX is shown in Figure 6-3.

6.3.1 Data Synthesis

Given a passage p, our goal is to generate a set of questions q and answers a for the self-training of the QA model. The RGX model first recognizes potential answer entities (AE) in p with an answer entity recognition (AER) model, and then generates questions based on the recognized AEs with a question generation (QG) model, and refines the AEs with a pretrained question answer extraction (QAE) model.

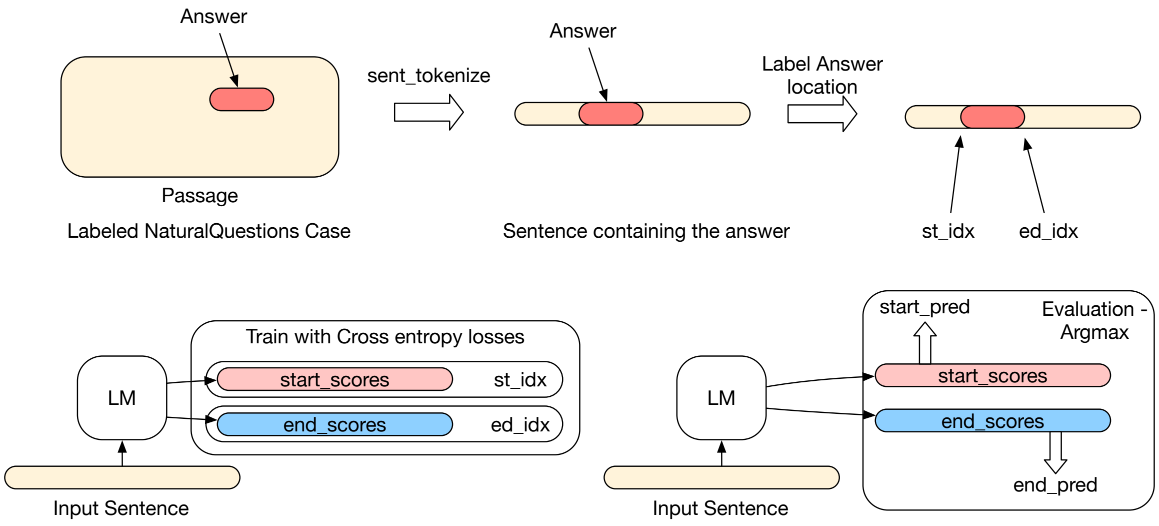


Figure 6-4: Pipeline of the answer entity recognition (AER) model.

Answer Entity Recognition (AER)

Recent QA synthesis models, QAGen2S [Shakeri et al., 2020] and SynQA [Bartolo et al., 2021], directly generate questions from passages by modeling $P_{qg}(q|p)$. In RGX, we first recognize all potential answer entities in a passage before generating questions for (1) increasing question diversity and coverage, and (2) modeling the mutual information between question generation and answering models in test time. The AER model is trained on the seed QA corpus.

We found that using an off-the-shelf named entity recognition (NER) model pretrained on the CONLL 2003 shared task [Bender et al., 2003] performs poorly as a AER model (shown in our experiments). To learn an effective recognizer, given a passage p and an annotated answer entity e, we select the sentence s containing e from p and train language models to recognize e in s. We tried two models for this task: a BIO sequence tagging model (AER-Tag) and an extractive AER model, which is similar to an extractive question answering model, for easier decoding. The model predicts the start and end positions of the answer entity e. With this method, we get potential answer entities by probabilities of all candidate spans. The pipeline of the AER model is shown in Figure 6-4.

Masked Question Generation

With AER, we replace the answer entity e in the passage p with a [MASK] token and obtain the masked passage p^*. We then build a question generator Q (denoted as QG interchangeably) that outputs answerable questions q in natural language with the concatenation of p^* and e as input, i.e., $q = Q([p^*, e])$. We adopt the BART sequence-to-sequence model [Lewis et al., 2019a] as the architecture of Q in our implementation, and we train Q on the question-answer pairs in the seed corpus by maximizing the likelihood of annotated questions.

Answer Extraction as Fine-grained AER

The answer extraction model A (denoted as QAE, question answering extractor) takes generated question q and the original passage p as inputs. Following the standard extractive QA method, we predict the answers by

$$I_{st}, I_{ed} = A([q, p]) \tag{6.1}$$

where I_{st} and I_{ed} stand for the start and end positions of e in p, respectively. We train the QAE model to predict I_{st} and I_{ed} separately with cross entropy losses.

Besides being trained with synthetic QA pairs and evaluated for the final QA performance, the QAE model is also a part of the data synthesis pipeline. After generating questions with the QG model, we use a pretrained QAE model to answer the generated questions. The final synthetic dataset is constructed by selecting generated questions and their corresponding QAE outputs.

6.3.2 Cooperative Self-training

Although the pretrained models can generate synthetic QA pairs from corpora in unseen domains, there is always a domain shift from the seed QA corpus for pretraining to the target. To efficiently adapt the pretrained models to the new domains, we propose a cooperative self-training algorithm that allows finetuning on the target corpora without additional annotations. The finetuning is based on a three-agent (AER, QG, QAE) cooperative framework, RGX. The pipeline is illustrated in Figure 7-6 and comprises the following steps:

1. Produce a masked passage by replacing an answer entity selected by AER with the '[MASK]' token.
2. Generate a question asking about the masked entity.
3. Feed the generated question and the original passage into the QAE to predict an answer span.
4. Optimize the QAE model with selected QA pairs.
5. Optimize the QG model with selected QA pairs.

In the proposed pipeline, all the AER, QG, and QAE models need pretraining to provide a reasonable start point for the cooperative self-training. However, the domain gap between the pretraining and the target corpus causes performance degradation. To mitigate the gap, we propose to measure the quality of generated questions and incorporate the measurement in loss functions. The quality is defined in two folds, correctness and difficulty. Firstly, the question should be fluent and answerable, and secondly, it should not be too trivial. To automatically select high-quality generated QA pairs, we introduce a expectation-maximization (EM) method based on QAE losses that learns the question quality without supervision.

Synthetic QA Selection with EM

To select synthetic QA pairs for finetuning, we first divide the generated questions based on the QAE loss for each question into three groups: low-, medium-, and high-loss questions. We can interpret questions with low loss as simple ones that the QAE model can easily answer. Medium-loss questions are challenging for the QAE, while those with high loss usually contain noise (e.g., containing grammatical errors or asking about incorrect answers). If we train the answering model with all questions, the training signal would be very noisy due to the high-loss questions. If we only reward questions that are correctly answered, the generator will converge to a trivial local optimum. Thus, we train the QG and QAE model with the low- and medium-loss questions, namely simple and challenging questions. For the entire pipeline to be fully-automatic, we classify a given QA pair into one of the three types described above. Note that simply setting the thresholds as hyper-parameters is difficult since the loss decreases as the QAE model varies with different passages and domains. In order to find the thresholds adaptively, we apply an expectation-maximization (EM) algorithm to cluster synthetic QA pairs for each passage.

We fine-tune both QG and QAE models with the selected simple and challenging QA pairs. After the training, re-running the RGX pipeline with the fine-tuned question generation model leads to improved data synthesis. Training the QAE model on the updated synthetic dataset can significant outperform the previous fine-tuned QAE model.

Maximum Mutual Information QA

[Li and Jurafsky, 2016] proposed a maximum mutual information (MMI) decoding method for machine translation, and [Tang et al., 2017] proposed a MMI method for jointly learning question generation and answering models. There is no previous study to our knowledge that applies MMI inference in test time of question answering that improves the final performance, because (1) modeling $P(q|p, a)$ for all possible answers (spans) a is too inefficient, and (2) unlike the QAE model that receives loss signals from all words in a given passage, the QG model does not receive loss signal from the passage directly, so $P_{qg}(q|p, a)$ it is less accurate for ranking answer spans.

However, the AER and self-training strategy enables efficient MMI inference for QA,

$$a = \underset{a}{\operatorname{argmax}}[\alpha \log P_{qg}(q|p, a) + \beta \log P_{qa}(a|p, q)]$$

At test time, we run the RGX pipeline for each passage without additional training to get fine-grained AEs and corresponding questions. We then take the top-k spans predicted by the QAE model, and only keep the top prediction and those that also appear in the fine-grained AE set. The filtering strategy dramatically reduces the number of potential answer spans, and removes unreasonable spans predicted by the QAE model.

6.4 Experiments

6.4.1 Modules

For our experiments, we train three modules for building the cooperative self-training environment RGX, i.e., the answer entity recognizer (AER), the question generator (QG), and the question-answering extractor (QAE). We used a BERT [Devlin et al., 2018] model for AER, a BART [Lewis et al., 2019a] model for QG, and an ELECTRA [Clark et al., 2020] model for AER and QAE. To compare with the results reported in [Shakeri et al., 2020] and [Bartolo et al., 2021], we also evaluate the performance of training BERT [Devlin et al., 2018] and RoBERTa [Liu et al., 2019] models on the synthetic QA data generated by RGX.

6.4.2 Data

For our experiments, we leverage Natural Questions [Kwiatkowski et al., 2019] and SQuAD v1.1 [Rajpurkar et al., 2016] as the seed corpora for pretraining all modules. To evaluate the performance of the proposed RGX on question answering tasks with different difficulty levels, we conduct experiments on both SQuAD v1.1 [Rajpurkar et al., 2016] and MRQA [Fisch et al., 2019] out-of-domains (BioASQ, TextbookQA, RACE, RelationExtraction, DuoRC, and DROP). In the following sections, we use the term SQuAD to represent the SQuAD v1.1 corpus. For self-training, we sample 3,000 passages from the training set of each corpus for data synthesis.

The SQuAD v1.1 is the easiest QA corpus used in this paper. The dataset contains $107,785$ question-answer pairs on 536 articles, which are split into passages. Each question is labeled with an answer that can be extracted from the given passage.

The Natural Questions dataset is a large-scale corpus designed for open-domain

Dataset	Num. Synthetic QA
BioASQ	123121
TextbookQA	133773
RACE	115847
RelExt.	52142
DuoRC	250698
DROP	100394

Table 6.1: Number of synthetic QA of each MRQA domain.

question answering. The dataset is more challenging than SQuAD. All questions are collected from human search queries and are annotated with long and abstractive answers. Some of the questions are also labeled with a short answer for learning answer-span extraction or reading comprehension. Focusing on the machine reading comprehension task, we select $106,926$ questions labeled with both long and short answers from the dataset for experiments.

For each target domain in MRQA, we collect the corresponding training data and sample 3,000 passages for QA synthesis. The number of synthetic QAs varies based on the length of input passages, and is shown in Table 6.1.

6.4.3 Answer Entity Recognition Details

In this section, we describe additional details of the AER methods. All AER models are pretrained on the Natural Questions corpus. To solve the sparsity problem, in other words, the passages are long but not all potential question-answer pairs are annotated, we train all following AER models by using the sentence containing the annotated answer entities as inputs, instead of the whole passage. If a sentence in the passage does not contain an annotated answer entity, we do not use it for training.

We introduce two types of AER methods, tagging-based AER (AER-tag) and extraction-based AER (AER-Search and AER-Coop). We next describe their training and how we use the trained model to recognize answer entities in our experiments.

AER-Tag

Training We apply a BIO tagging model for answer entity recognition in the AER-Tab method. We train the model to classify all tokens in the input sentence into three classes,

- B(egin) - the first token of the annotated answer entity
- I(nsize) - other tokens of the annotated answer entity
- O(utside) - tokens that are not a part of the annotated answer entity

Evaluation Given an input passage, we run the trained BIO tagging model on each of its sentences and greedily predict answer entities. There might be more than one answer entities predicted in each sentence, and we only use the answer entities start with a predicted B tag.

AER-LM

Training For the AER-LM method, we need to pretrain an extraction-based AER model. We also take a sentence of L tokens containing an annotated answer entity as an example. Using an extraction model, which is similar to the question answering model, we train the model to predict the start and end location of the annotated answer entity. The model outputs a start score and an end score for each token, and predicts the start/end locations by selecting the tokens that are assigned with highest scores. The model is trained with cross-entropy loss, by regarding the extraction task as two L-class classification tasks.
Evaluation For evaluation, we first run the model on each sentence of the input passages and calculate the start and end scores for each token. For each span $(x_i, x_{i+1}, \dots, x_j)$ that is not longer than L_{span} tokens, we calculate the span score with

$$s_{ij} = s_{st}^i + s_{ed}^j \tag{6.2}$$

where s_{st}^i is the start score of the first token of span (i, j), and s_{ed}^j is the end score of the last token of the span. In practice, we set $L_{span} = 10$.

To re-rank all possible answer entities, we select the top $N_0 = 40$ spans according to s_{ij} for each passage. For all selected answer entities, we generate questions with a pretrained question generator and collect the generation perplexity of the questions. We select $N_{search} = 5$ question-answer pairs with the lowest perplexities for the final question-answering fine-tuning.

AER-Coop

In AER-Coop, we use the same extraction training method applied in AER-Search, and we also use the s_{ij} scores to select the top $N_0 = 40$ preliminary answer entities for further search. The difference is that we search for final answer entities cooperatively with the pretrained question generator and question answering extractor.

With the question generator and question answering extractor, we re-rank the recognized answer entities with the following score

$$s^c_{ij} = \gamma \cdot I_c - p \tag{6.3}$$

where γ is a large, positive coefficient, p is the perplexity of generated question based on span (i, j), and $I_c = 1$ if the generated question is correctly answered, and otherwise $I_c = 0$.

Answer Entity Overlapping

We found that the extraction-based AER model leads to overlapping problems, since a large start or end score assigned to a token leads to many candidate answer entities that start or end at the token. In practice, if an answer entity is selected by the AER-Search and AER-Coop method, we no longer consider any other answer entities that overlap with the selected ones.

6.4.4 Implementation Details

Pretraining We pretrain the AER, QG, and QAE models on NaturalQuestions and SQuAD (i.e., the seed) corpora. For NaturalQuestions, we only use the data points containing a short answer. For Cooperative training, we follow the steps described in Section 6.3.2 for the cooperative training phase.

Self-training We apply self-training for QG and QAE by finetuning the models on selected synthetic QA pairs using the same method as pretraining. The AER model is fixed after pretraining. The QAE model is finetuned using the official Huggingface [Wolf et al., 2019] training scripts for question answering.

Hyper-parameters There are three phases of model training in this work: pretraining on the Natural Question corpus, cooperative adaptation with reinforcement learning on the target corpora, and final fine-tuning on the target corpora. We adopt most of the hyper-parameters reported in the original BERT [Devlin et al., 2018], BART [Lewis et al., 2019a], and ELECTRA [Clark et al., 2020] papers. We select the final fine-tuning learning rates from $\{3e-5, 4e-5, 5e-5\}$ and report the highest performance. All the other hyper-parameters are the same as reported in the corresponding papers. For all the phases, we fix $eps = 1e-6$ and $s_w = 2000$, where s_w is the number of warm-up steps, and we apply no weight decays. In the following sections, we describe the details of each training phase. We use BART-large (406M parameters) and ELECTRA-large (335M parameters) models for our experiments. We run our experiments on 2 Tesla V100 GPUs. Training the QAE models on augmented data takes around 4 hours.

For the maximum mutual information inference process shown in the equation below,

$$a = \underset{a}{\operatorname{argmax}}[\alpha \log P_{qg}(q|p, a) + \beta \log P_{qa}(a|p, q)]$$

we fix $\beta = 1$. We use an adaptive α value by comparing the synthetic question generated by the QG model and the input question. For each answer entity a, we calculate

$$\alpha = \mathtt{max}(1 - \mathtt{abs}(\frac{q_{input}}{q_{gen}} - 1), 0.1)$$

This value normalizes the question probability $p(q|p, a)$ estimated by the QG model, since generated questions from some answer entities is easier than other spans in the same passage, which makes the QG model assign all natural questions a relatively low perplexity.

6.4.5 Experimental Results

We assess the performance of RGX with both semi-annotated and zero-annotated evaluation on unseen domains. In our semi-annotated setting, we use the annotated answer entities in the target corpora but utilize QG to generate questions for obtaining the training question-answer pairs. The labeled questions are not used. We employ no annotation from the target corpora for the out-of-domain task but automatically construct the question-answer training pairs with entities and questions inferred by AER and QG on the corpora.

Semi-annotated Evaluation

The model performance with the pretrained QA model, RGX, and SOTA trained with full-supervision is shown in Table 6.2.

Models	**EM**	**F1**
Source domain: NQ, Target domain: SQuAD		
ELECTRA-large (NaturalQuestions)	67.8	80.3
RGX	83.1	90.7
–w/o Coop. ST	81.2	89.1
ELECTRA-large (SQuAD)	89.7	94.9

Table 6.2: The performance of the question answering models in the semi-annotated setting. RGX stands for our cooperative training approach, and Coop. ST stands for cooperative self-training.

Table 6.2 shows that RGX yields improvement over the pretrained model, approaching the SOTA performance of the fully trained ELECTRA-large-discriminator model. The experiment result suggests that the cooperative learning strategy improves the question generation model.

Models		ELECTRA	QAGen2S	SynQA	RGX	-MMI	-EM	-CST
BioASQ	EM	41.9	43.2	-	**50.3**	49.7	48.2	45.4
	F1	59.0	64.1	-	**70.1**	69.1	67.9	66.4
TQA	EM	31.9	39.9	-	**49.9**	49.4	47.4	41.9
	F1	41.5	51.7	-	**60.9**	60.6	59.8	53.8
RACE	EM	32.4	33.7	-	**40.3**	39.7	38.3	35.1
	F1	43.4	45.5	-	**52.4**	51.5	50.5	47.2
RelExt.	EM	67.7	71.6	-	**76.1**	75.4	74.1	72.7
	F1	81.8	84.4	-	**87.2**	86.7	86.2	85.4
DuoRC	EM	40.0	43.8	-	**47.8**	46.9	46.6	45.5
	F1	48.5	53.2	-	**58.4**	57.5	56.9	54.9
DROP	EM	**39.3**	24.2	-	27.6	27.1	26.1	24.6
	F1	**51.1**	37.1	-	42.1	41.7	40.9	37.9
Avg.	EM	42.2	42.7	-	**48.7**	46.8	45.8	44.2
	F1	54.2	56.0	-	**61.9**	61.2	60.4	57.6

Table 6.3: The QA performance evaluation on the out-of-domains of the MRQA benchmark by pretaining on NaturalQuestions. RGX stands for our method, MMI stands for maximum mutual information inference, EM stands for question selection with EM, and CST stands for the cooperative self training.

Models		ELECTRA	QAGen2S	SynQA	RGX	-MMI	-EM	-CST
BioASQ	EM	58.7	56.8	55.1	**60.3**	59.2	52.1	57.5
	F1	73.1	71.7	68.7	**74.8**	73.6	64.0	72.1
TQA	EM	43.0	48.0	41.4	**51.2**	50.1	50.6	48.6
	F1	53.6	56.5	50.2	**61.2**	60.4	58.9	57.0
RACE	EM	38.3	43.4	40.2	44.9	**46.3**	35.4	43.8
	F1	52.5	54.9	54.2	**58.7**	57.6	48.3	55.2
RelExt.	EM	79.0	73.4	78.9	**79.2**	78.9	75.6	74.3
	F1	88.4	84.8	**88.6**	**88.6**	88.5	85.9	85.3
DuoRC	EM	53.1	53.3	51.7	**57.4**	56.2	55.6	53.9
	F1	64.2	64.6	62.1	**66.2**	65.7	64.9	65.3
DROP	EM	48.3	42.2	**64.9**	47.6	46.9	40.7	43.0
	F1	60.8	54.5	**73.0**	60.9	60.6	53.2	55.1
Avg.	EM	53.4	52.8	55.3	**56.8**	56.3	51.7	53.5
	F1	65.4	64.5	66.1	**68.4**	67.7	62.5	65.0

Table 6.4: The QA performance evaluation on out-of-domain subsets of the MRQA benchmark by pretraining on SQuAD v1.1.

	QAGen2S	**SynQA**	**RGX**
Pretraining	XQ	SQ+AQA	XQ
Synthesis	Target	Wikipedia	Target
Finetuning	XQ+Syn	SQ+AQA+Syn	XQ+Syn
AER Model	None	None	ELECTRA
Coop. ST	No	No	Yes
QA Num.	1M	1.5M	0.3M

Table 6.5: Comparison of different self-training methods. XQ stands for "NaturalQuestions or SQuAD".

Out-of-domain Evaluation

We also evaluate the models in unseen domains, where we do not use any annotated QA for finetuning. We train the QAE models based on the synthetic training data and evaluate the models on the target domains. We compare RGX with latest self-trainig QA methods, QAGen2S [Shakeri et al., 2020] and SynQA [Bartolo et al., 2021]. Since QAGen2S did not report full MRQA results, we implemented our own version. We first present the RGX performance and the results reported by the authors QAGen2S and SynQA, and then conduct ablation study by training different language models on RGX synthetic QA data.

The full evaluation results on MRQA out-of-domains are shown in Tables 6.3 and 6.4, and the experiment setting comparison is shown in Table 6.5. The results show that the models trained with the RGX framework achieve significantly higher EM and F1 scores on most domains, comparing to both pretrained QA models and self-training baselines. The results showed that the RGX model achieves 7.7 and 3.0 average F1 improvement over ELECTRA, the SOTA pretrained language model for QA, by pretraining on NQ and SQuAD respectively. The improvement over previous SOTA self-training QA methods, QAGen2S and SynQA, is also significant on both pretraining corpora, although SynQA applies complicated adversarial QA annotation. The largest gain we got is adapting NQ model to TextbookQA domain, increasing 18.0 EM and 19.4 F1 scores. Note that our model still outperforms all baselines without MMI. The performance on the DROP benchmark drops since DROP requires multi-step reasoning, but the synthetic generation model tends to generate safe question-answer

Models	**EM**	**F1**
Source domain: NQ, Target domain: SQuAD		
Pretrained NQ	67.8	80.3
RGX + NER	27.4	35.4
RGX + AER-Tag	71.4	82.4
RGX + AER-LM	72.7	85.9
RGX + AER-EM	79.2	89.4
Supervised ELECTRA-large	89.7	94.9

Table 6.6: Comparison of different AER strategies. NER stands for the BERT named entity recognition model trained on the CONLL 2003 shared task.

	ELECTRA		**Top-k+MMI**		**AER+MMI**	
	EM	F1	EM	F1	EM	F1
BioASQ	58.7	73.1	57.8	72.9	59.9	74.0
TextbookQA	43.0	54.6	44.6	54.9	45.3	55.4
RACE	38.3	52.5	38.1	52.4	39.7	54.1
RelExt	79.0	88.4	78.6	88.3	79.2	88.6
DuoRC	53.1	64.2	52.6	64.3	53.8	65.1
DROP	48.3	60.8	46.7	60.8	49.7	61.5

Table 6.7: Comparison between maximum mutual information inference performance grounded on AER results and top-k ($k = 20$) predictions of the QA model.

pairs. We also found that without selecting harder questions with SEM in RGX, the performance is significantly lower. These facts indicate that the QA model needs hard training examples for better performance, and explains the good performance of SynQA on DROP. For the same reason, the performance drop led by removing EM from RGX is significantly larger when the QG model is pretrained on SQuAD, since SQuAD questions are more coherent with the context than NQ, and selecting simple questions for RGX training will encourage the model to generate trivial questions, which is harmful for the QA training.

Models	Mean Len.	Std Len.	Vocab
Ground-truth	11.29	3.72	988703
Semi-anno. RGX	10.54	1.91	923191
–w/o Coop. ST	10.49	2.48	919105
Zero-anno. RGX	10.53	1.94	873300
–w/o Coop. ST	10.57	2.63	789924
–w/o AER	10.60	1.87	743454
–w/o EM	10.18	1.62	692301

Table 6.8: The vocabulary sizes and lengths of Annotated and generated questions on SQuAD under both semi- and zero-annotated settings in unseen domains

Domain	**RGX w/o Coop. ST**		**RGX**	
	Hit	BLEU	Hit	BLEU
BioASQ	68.1	5.9	75.8	12.7
TextbookQA	43.7	7.5	58.2	13.2
RACE	8.3	5.2	12.3	6.8
RelExt.	47.4	2.8	54.2	3.3
DuoRC	53.5	6.7	60.0	7.5
DROP	73.5	12.3	75.3	9.3

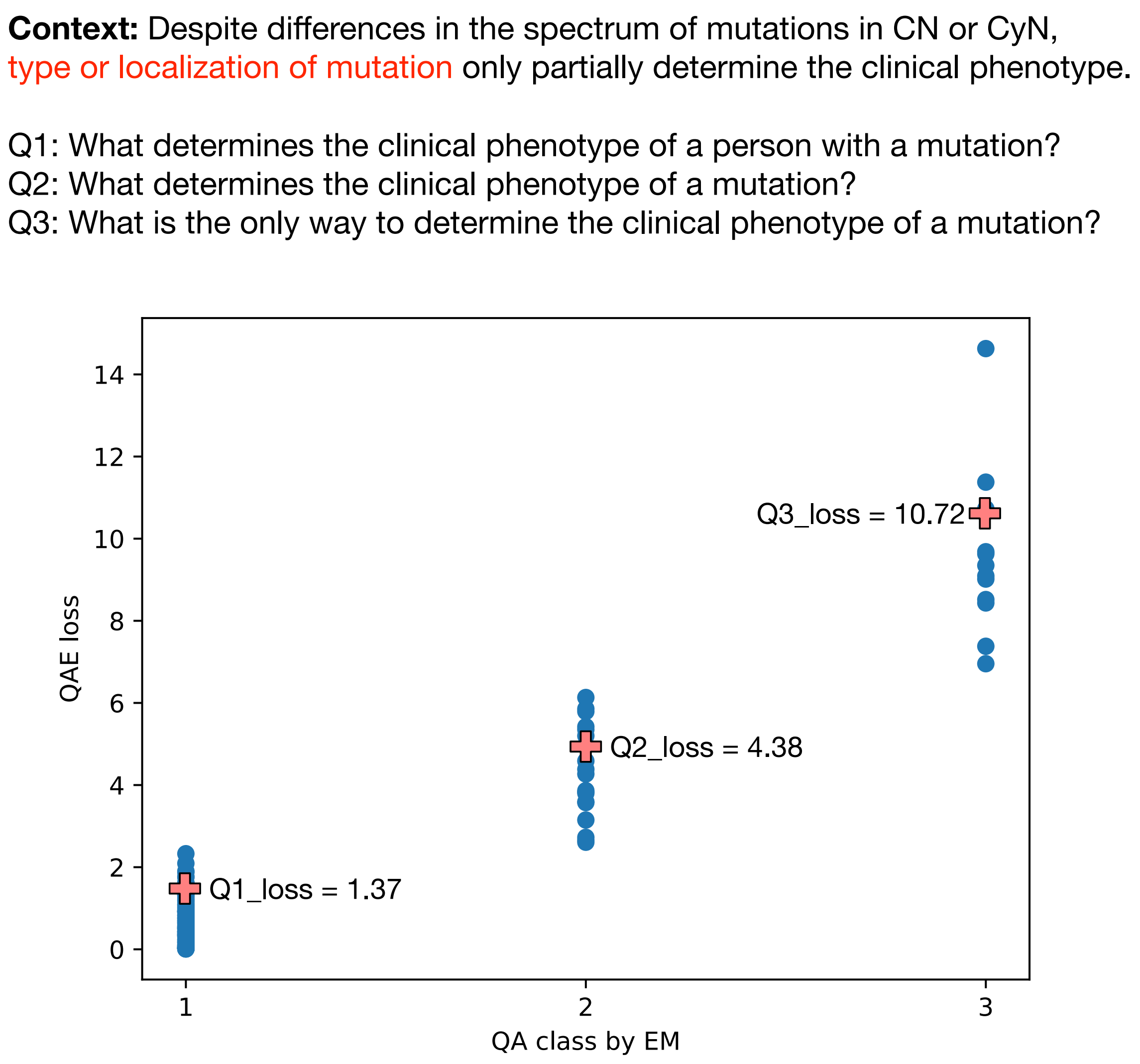
Table 6.9: Evaluation of the answer hit rates and question BLEU scores of the synthetic data.

6.4.6 Analysis

Answer Entity Recognition

We first compare the performance of different AER models and strategies by setting NQ as the source domain and SQuAD 1.1 as the target domain in Table 6.6. The results showed that the choice of AER model and strategy significantly influences the final QA performance. The low performance of the NER model trained on CONLL shared task suggests the importance of our AER module. We notice that the improvement from the cooperative learning over the pretrained models is higher in the zero-annotation setting than the semi-annotated task. The observation indicates that the model trained with RGX is more robust against the automatically recognized answer entities.

The AER method also enables and improves the maximum mutual information


Context: Despite differences in the spectrum of mutations in CN or CyN, type or localization of mutation only partially determine the clinical phenotype.

Q1: What determines the clinical phenotype of a person with a mutation?
Q2: What determines the clinical phenotype of a mutation?
Q3: What is the only way to determine the clinical phenotype of a mutation?

Figure 6-5: Generated questions about the same answer entity classified into different types by EM.

(MMI) inference in test time. Tables 6.3 and 6.4 shows that MMI achieves the best performance, and we also show that the MMI accuracy is hurt without AER. Table 6.7 shows that MMI grounded on AER constantly outperform the ELECTRA model, but grounding on top-k seriously hurts the EM scores. Some invalid answer predictions leads to low question generation perplexities, which makes MMI inference noisy. Table 6.8 shows that the QG model generated more diverse questions based on the AER outputs.

Synthetic QA Selection with EM

Previous experiments showed that selecting non-trivial synthetic QA pairs is essential for RGX to achieve high performance. Tables 6.3 and 6.4 shows that the performance of cooperative self-trained RGX is much lower than the pretrained baseline without EM. If selecting QA pairs with low perplexities instead of EM, the QA diversity is significantly lower as shown in Table 6.8, thus makes the QAE model overfit to simple training cases and hurts the QA accuracy. We show questions about the same answer entity being classified into simple, challenging, and difficult types by EM in Figure 6-5. The data points in the plot represents the losses of synthetic QA pairs and the predicted QA type. Based on the highlighted answer entity, question 1 and 2 are predicted as correct questions, while question 3, which has a relatively high QAE loss, is regarded as a wrong question. Note that we only generate one question for each span recognized by the AER model, but different questions might be re-directed to the same AE after QAE fine-graining.

Cooperative Self-training

We found that the cooperative self-training method improves the domain adaptation ability of self-trained QA models by increasing both accuracy and diversity of QA synthesis.

Accuracy We also evaluate the quality of the generated QA pairs without a downstream task by assessing the answer entity hit rate and the BLEU scores of generated questions using the evaluation sets of each domain. The results are shown in Table 6.9, indicating that RGX find mores human-annotated answer entities, and the generated questions have higher BLEU scores on all domains. The evaluation results show that the synthetic QA pairs generated by RGX covers more human annotated answer entities, and the generated questions are more similar to human annotations than the pretrained question generation model.

Diversity We compare the lengths and vocabulary sizes of the questions and summarize the statistics in Table 6.8, which shows that the ground-truth questions are longer

Architecturally, the school has a Catholic character. Atop the Main Building's gold dome is a golden statue of the Virgin Mary. Immediately in front of the Main Building and facing it, is a copper statue of Christ with arms upraised with the legend "Venite Ad Me Omnes". Next to the Main Building is the Basilica of the Sacred Heart. Immediately behind the basilica is the Grotto, a Marian place of prayer and reflection. It is a replica of the grotto at Lourdes, France where the Virgin Mary reputedly appeared to Saint Bernadette Soubirous in 1858. At the end of the main drive (and in a direct line that connects through 3 statues and the Gold Dome), is a simple, modern stone statue of Mary.

Annotated	Pretrained	RGX
Saint Bernadette Soubirous	**a Marian place of prayer and reflection**	**a Marian place of prayer and reflection**
To whom did the Virgin Mary allegedly appear in 1858 in Lourdes France?	what is the grotto at st bernadette's?	what is the grotto in st bernadette school?
a copper statue of Christ	**the grotto at Lourdes, France**	**Venite Ad Me Omnes**
What is in front of the Notre Dame Main Building?	where is the grotto located at st bernadette school?	what is the message on the statue in front of st bernadette school?
the Main Building	**Immediately behind the basilica is the Grotto**	**1858**
The Basilica of the Sacred heart at Notre Dame is beside to which structure?	what is the grotto in st peter's school?	when was the grotto at lourdes built?
a Marian place of prayer and reflection	**copper statue of Christ with arms upraised**	**a simple, modern stone statue of Mary**
What is the Grotto at Notre Dame?	what is it a statue of christ?	what is the statue at st bernadette school?
a golden statue of the Virgin Mary	**a replica**	**the grotto at Lourdes, France**
What sits on top of the Main Building at Notre Dame?	is the grotto at st bernadette school in paris a replica of which European landmark?	what is the replica of st bernadette's school in paris?

Figure 6-6: An example of a passage in the training set of the SQuAD corpus. We list the annotated question-answer pairs, and the question-answer pairs generated by the models pretrained on NQ and finetuned by RGX. The bold texts are annotated or recognized answer entities. Adapting from NQ is difficult since the questions in NQ do not strictly coherent with a given context.

and more diverse in vocabulary than the generated ones. However, the cooperative self-training, together with AER and EM, improves the vocabulary diversity. We observe a correlation between the vocabulary size and the QA performance reported in Table 6.2 and 6.6, presumably because the QAE model requires diverse knowledge for training. Thus, we believe generating more diverse QA pairs with good quality will be a critical next step to improve RGX.

Case Study An example of a SQuAD [Rajpurkar et al., 2016] passage is shown in Table **??**. We list the annotated and generated question-answer pairs by different models. The table shows that the models can recognize reasonable answer entities other than the annotated ones, and RGX generates more natural QAs.

6.5 Chapter Summary

In this chapter, we propose a cooperative self-training framework, RGX, consisting of an answer entity Recognizer, a question Generator, and an answer eXtractor, for question generation and answering. We also introduce in the framework an expectation-maximization method that measures the quality of generated questions for reinforced finetuning of the question generation models. Experiments show that RGX significantly outperforms pretrained and self-trained model baselines while adapted to unseen domains, suggesting that RGX is a promising framework for making extractive question answering methods more scalable and less dependent on human annotation.

Chapter 7

Entailment Self-training for Language Task Adaptation

7.1 Introduction

In previous chapters, we have explored self-training methods for conversational tasks, including dialog response & action selection, and question answering. In this chapter, we seek to extend the self-training algorithm for more general natural language understanding tasks, including language inference, sentiment analysis, and fact-checking. To generalize our self-training method to different tasks, we formulate all tasks as language entailment predictions.

Entailment has been an important topic in logic and linguistics research for decades [Routley and Meyer, 1973]. Given two propositions p and q, we say p entails q ($p \Rightarrow q$) if p is true, then q must be true. In the context of large-scale self-supervised language model pretraining [Devlin et al., 2018, Raffel et al., 2019, He et al., 2020], entailment is currently considered as one of many regular downstream natural language understanding tasks, namely natural language inference (NLI) [Bowman et al., 2015], where p and q are natural sentences and the task is to predict if p and q has an entail, neutral, or contradict relation. However many if not all, other NLU tasks can also be formulated as NLI tasks. For example, verifying the correctness of an answer to a question can be solved by predicting the entailment between the QA pair and a given passage [Rajpurkar et al., 2016]. To predict if two questions are asking for the same answer [Wang et al., 2017], we can evaluate the entailment between the following texts: "The answer to question A" and "The answer to question B". Even for the tasks where only one input is required instead of a text pair, for example, sentiment analysis [Socher et al., 2013], we can construct an entailment prediction task for the following sentence pair: "The user commented: x", and "The user likes the product". If the result is entailment, then x is a positive review, otherwise, x expresses a negative attitude.

The previous studies and chapters in this thesis have explored generalizing models to different domains but within the same task. In this chapter, we propose an entailment self-training method that adapts to different NLU tasks with data synthesis and entailment prediction. We evaluate the method on two different tasks. Firstly, we re-use the RGX model proposed in the previous chapter for QA generation to solve a fact checking problem. We then evaluate the performance and robustness of few-shot task adaptation using our entailment self-training method.

7.2 QA-based Fact Checking

Fact checking is an important technique to identify misinformation by comparing an input claim with a trusted knowledge base [Nadeem et al., 2019]. Previous end-to-end fact checking models [Mohtarami et al., 2018] are black boxes that encode input claims and long grounded documents as a batch and output a single prediction. This line of research is limited by the difficulty of data annotation. Labeling misinformation is costly and the size of current publicly available fact checking corpora is limited. For example, the current largest fact checking corpus, FEVER [Thorne et al., 2018], only contains 2,582 grounded documents for training. Meanwhile, SQuAD v1.1 [Rajpurkar et al., 2016], a popular QA benchmark, contains over 18k training passages, and the MNLI benchmark contains 39k premise texts [Bowman et al., 2015]. This motivates us to adapt question answering models that are trained on a much larger training set to the fact checking task. Experiments show that with this method, we can conduct both supervised and unsupervised task adaptations, while both the performance and interpretability of the QA-based fact-checking method are better than supervised end-to-end fact-checking models.

7.2.1 QA-based Fact-checking Pipeline

Based on the document retrieval results provided by the FAKTA system [Nadeem et al., 2019], we propose a 3-step pipeline for QA-based fact-checking as follows,

- Generate and cluster question-answer pairs for input claims (Q, A_c)
- Answer the generated question grounded on retrieved documents (Q, A_d)
- Predict the fact-checking result by calculating the entailment of two QA pairs (Q, A_c) and (Q, A_d)

where Q stands for generated questions, A_c stands for answer entities recognized in input claims, and A_d are predicted answers to Q grounded on retrieved documents. The entire pipeline is shown in Figure 7-1. Given an input text claim, it is first processed by an Answer Entity Recognizer component to detect and extract entities in the claim. The entities and the claim are given as inputs to a Question Generator component to generate question-answer pairs from the claim. The resulting questions are then categorized based on their corresponding answers in the claim using the following text-based similarity metrics: Longest common subsequence (LCS) [Irving and Fraser, 1992], n-gram [Barrón-Cedeno et al., 2010], Levenshtein distance [Navarro, 2001] as well as Cosine [Qian et al., 2004] and Jaccard [Jaccard, 1901] similarity metrics. This categorization reduces the impact of poorly generated questions on the final performance of the system. After categorization, the top K retrieved documents (obtained via FAKTA) and the generated questions are passed to an Answer Extractor component to predict the answers for the questions from the documents. These answers in conjunction with answers from the input claim are processed in an Answer Matching component to measure the extent of matching for each question category. Finally, the claim is predicted to be True if at least one matched answer can be found in each question category, otherwise it is predicted to be False.

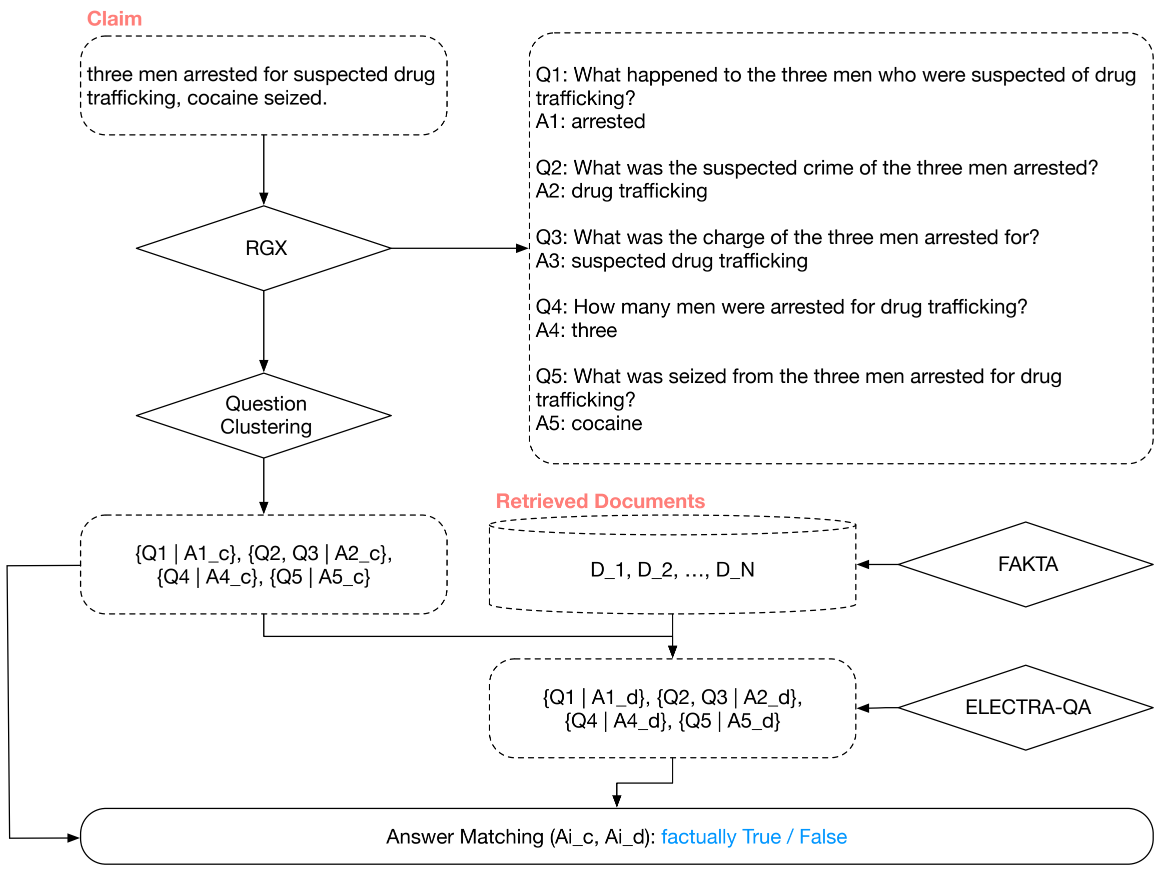

Claim
three men arrested for suspected drug trafficking, cocaine seized.
RGX
Q1: What happened to the three men who were suspected of drug trafficking?
A1: arrested
Q2: What was the suspected crime of the three men arrested?
A2: drug trafficking
Q3: What was the charge of the three men arrested for?
A3: suspected drug trafficking
Q4: How many men were arrested for drug trafficking?
A4: three
Q5: What was seized from the three men arrested for drug trafficking?
A5: cocaine
Question Clustering
{Q1 | A1_c}, {Q2, Q3 | A2_c}, {Q4 | A4_c}, {Q5 | A5_c}
Retrieved Documents
D_1, D_2, ..., D_N
FAKTA
{Q1 | A1_d}, {Q2, Q3 | A2_d}, {Q4 | A4_d}, {Q5 | A5_d}
ELECTRA-QA
Answer Matching (Ai_c, Ai_d): factually True / False

Figure 7-1: The architecture of the QA-based fact-checking system.

As in the previous chapter, we employ ELECTRA [Clark et al., 2020] for the answer entity recognizer and answer extractor components, and BART [Lewis et al., 2019a] for the question generator component. We leverage the SQuAD v1.1 [Rajpurkar et al., 2016] dataset as the seed corpus for pretraining these components. The dataset contains 107,785 question-answer pairs on 18,856 passages. Each question is labeled with an answer that can be extracted from the given passage. Our full fact checking pipeline is an aggregation of FAKTA and the QA-based systems. It uses an unsupervised combination of the systems in which the aggregation pipeline would relies on the FAKTA output if the probability score for its prediction is higher than a threshold (we use the threshold 0.5), otherwise it relies on the prediction from QA-based system.

We also explore a supervised combination of the FAKTA and QA-based systems using a SVM classifier with a rich set of features including text-based features (e.g., n-grams), FAKTA-based features (e.g., agree, disagree and discuss scores for top

K retrieved documents, max, min and mean of the stance scores, and related and unrelated scores for the documents), and QA-based features (e.g., QA-based output, fraction of matched answers over all). However, the unsupervised combination performs better than the supervised combination based on our experiments.

7.2.2 Improved RGX for Fact Checking

We use the RGX framework proposed in the previous chapter for fact checking as shown in Figure 7-1, but the previous RGX model is not enough for the fact checking system. For example in the claim "three men arrested for suspected drug trafficking, cocaine seized", a fact checker might need to go through all entities in the claim, including "three", "men", "drug trafficking", and "cocaine". However, the RGX model does not guarantee coverage of all potential answer entities. This might lead to inaccurate fact checking because an entity will not be checked if no grounded QA-pair is generated.

To improve the coverage of answer entities, we propose a hierachical RGX model that generates more verified QA pairs. In the standard RGX model, we call a QA pair certified if the QA model successfully extracts the answer entity that is used to generate the question. Let a be a answer span, $G(\cdot)$ stand for the question generation model, $A(\cdot)$ stand for the question answering model, and c stands for a context passage, we call a QA pair "verified" if

$$a = A(q, c)$$

and

$$q = G(a, c)$$

Because of the difficulty of the QA task, the answer entity cannot be successfully extracted even when a correct question is generated. To simplify the answer extraction task, we can shorten the context c to provide more concentrated information for the answering model to extract. To achieve this goal, we change the AER model with a constituency parser. We regard all nounes and spans that are recognized by the constituency parser as answer entities. Furthermore, we also use the same constituency parser to split the context into sentences and spans. For each potential answer entity, we run the RGX framework on all spans that are on its path to the root node. The process starts from the root node, which is the context itself, and goes down to the direct parent node of the target answer entity. The loop stops when the answer entity is verified, otherwise there is no question assigned to the answer entity if no QA pairs can be verified on any level. The new method is illustrated in Figure 7-2.

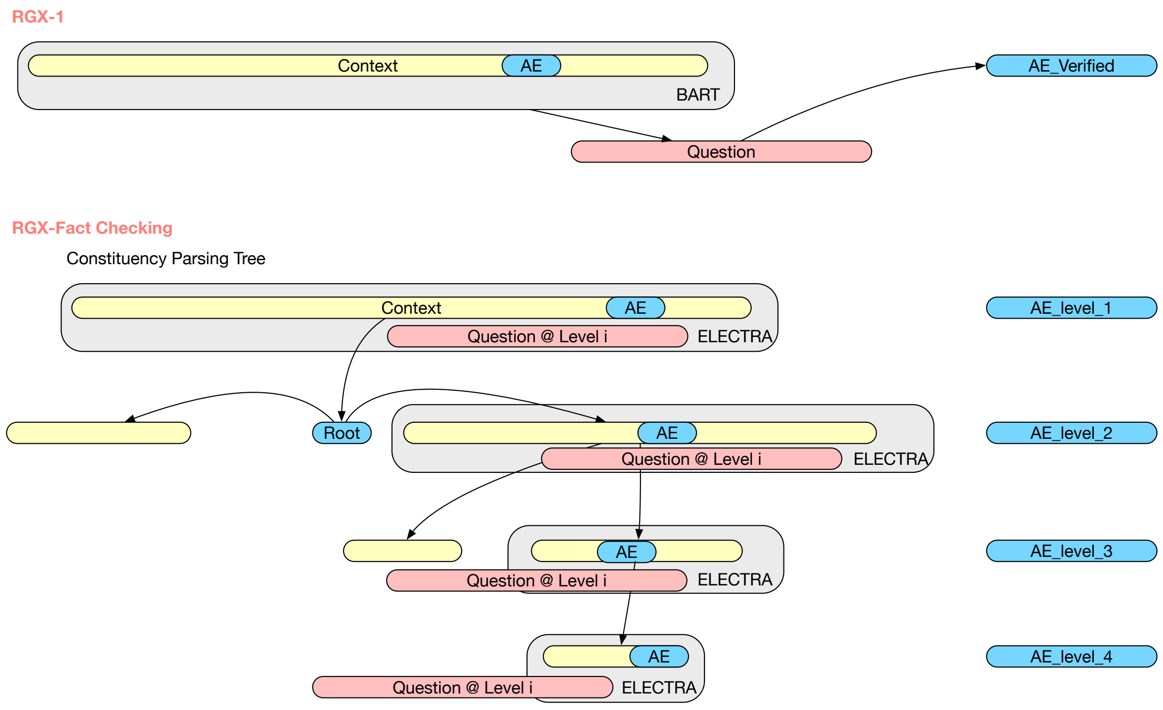


Figure 7-2: The architecture of hierarchical RGX for fact checking.

In the figure, the upper part is “RGX-1”, standing for the original RGX model for comparison, while the lower part is the hierarchical RGX model. If a span stands for a reasonable answer entity, it is more possible to be verified with a question in the hierarchical RGX pipeline, leading to more comprehensive fact checking.

7.2.3 Fact Checking Results

We evaluated the system on a corpus consisting of three different sets of claims which are relevant to Singapore: WildForums which contains 50 claims collected

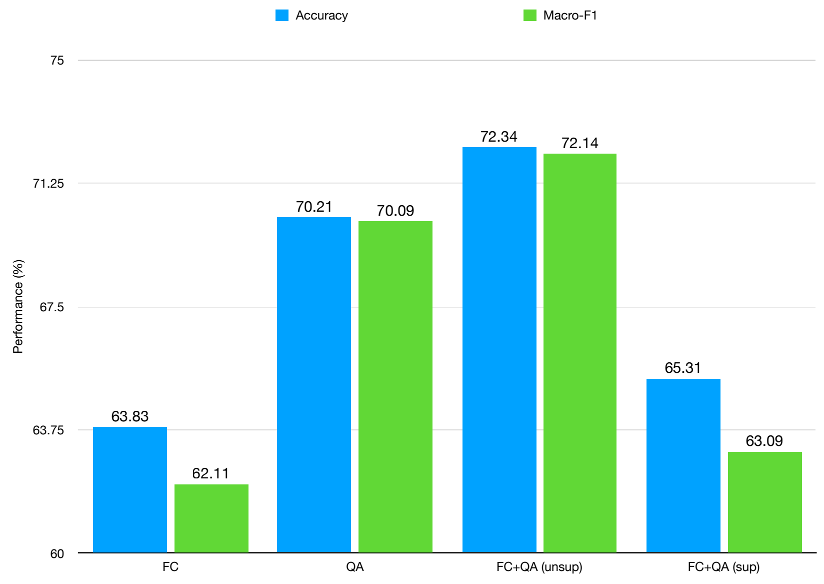


Figure 7-3: Results of QA-based fact checking system on the Wild-Forums data.

from Hardwarezone[1] and Reddit, Wild-Synthetics which contains 36 claims created synthetically, and factChecker contains 50 claims collected from Black Dot Research[2] and AFP Factcheck[3]. We use the following evaluation measures

- Accuracy: The fraction of correctly classified examples.
- Macro-F_1: The average of F_1 scores computed separately for each class.

Figures 7-3, 7-4 and 7-5 show the performance of our fact checking models on the Wild-Forums, Wild-Synthetics, and factChecker test datasets respectively. In the Figures, blue and green colors indicate accuracy and macro-F1, and the performance is shown for FAKTA (FC in the Figures), the QA-based fact checking model (QA), and the supervised and unsupervised combination of FAKTA and QA-based models indicated by FC+QA(sup) and FC+QA(unsup) in the figures. The results show that the performance for the QA-based model is significantly higher than FAKTA across datasets. This is because, in contrast to FAKTA, the QA-based model performs fine-grained fact checking at sub-claim level and remedies minor linguistic manipulations in the input claim. The highest performance achieved using QA-based model on the

[1]`https://www.hardwarezone.com.sg`
[2]`https://blackdotresearch.sg`
[3]`https://factcheck.afp.com`

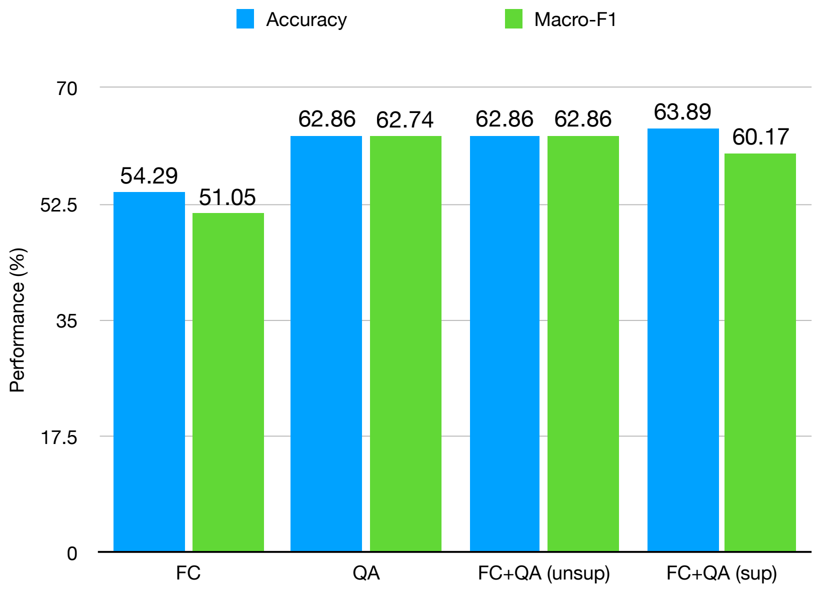


Figure 7-4: Results of QA-based fact checking system on the Wild-Synthetics data.

factChecker dataset, and the model also obtained significantly high performance on the other two datasets. The unsupervised combination of the two models obtained the highest performance on the Wild-Forums and Wild-Synthetics datasets. Overall, the results indicate (a) the QA-based model performs better than FAKTA across datasets, and (b) on average, the QA-based model and its unsupervised combination with FAKTA result in higher performance on our three datasets.

7.3 Self-training for Robust Language Understanding

Although achieving state-of-the-art performance in different natural language understanding tasks [Devlin et al., 2018, Liu et al., 2019, Yang et al., 2019, Clark et al., 2020, He et al., 2020, Joshi et al., 2020], large-scaled pretrained language models are still challenged by difficult evaluation examples crafted by adversarial attacks or model-in-loop adversarial data annotation [Wang et al., 2021, Jin et al., 2020, Bartolo et al., 2020, Zang et al., 2019, Garg and Ramakrishnan, 2020, Li et al., 2020]. The performance of a finetuned language model on an adversarial evaluation benchmark can be much lower than evaluating on a standard benchmark even in the same domain. However, experiments also showed that finetuning models on adversarial data annotation can

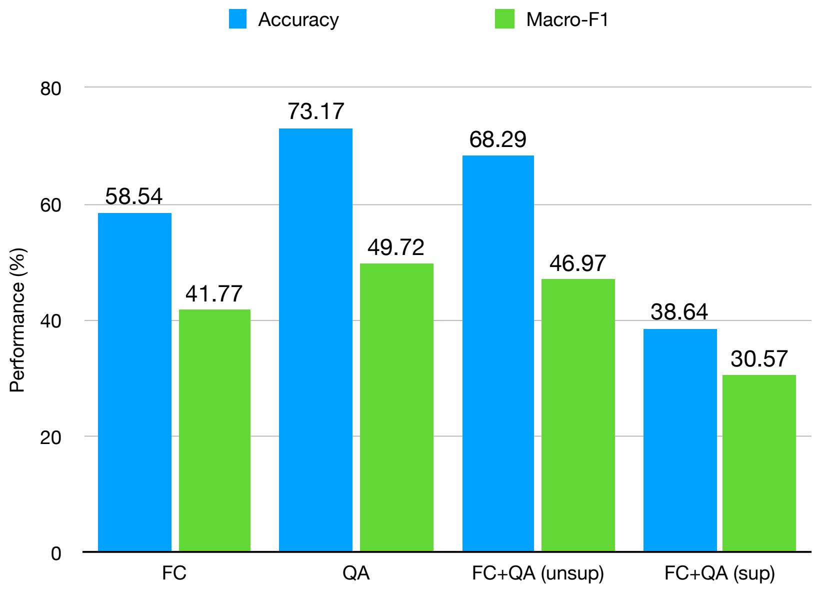


Figure 7-5: Results of QA-based fact checking system on the FactChecker data.

significantly mitigate the performance gap [Bartolo et al., 2020].

One reason that causes this problem is the difference in data distribution between the training and evaluation splits, even though they are in the same knowledge domain. For example, SQuAD [Rajpurkar et al., 2016] and AdversarialQA [Bartolo et al., 2020] are both question answering corpora based on annotated Wikipedia passages, but the performance of fine-tuning on the SQuAD training set and evaluating on the test set of AdversarialQA is significantly lower than fine-tuning on the AdversarialQA training set. Given this fact, training on normal corpora and evaluating on adversarial evaluation benchmarks is an out-of-domain adaptation problem where the model needs to generalize to different data distributions. Since there are no adversarial training cases available in most cases, and the adversarial evaluation set lies in the same domain as the training set, it is difficult to solve the problem with regular few-shot and self-supervised transfer learning techniques to solve the problem [Tan et al., 2018, Noroozi et al., 2018].

We propose to improve the adversarial robustness of neural language models with the self-training method [Zoph et al., 2020], which is a special transfer learning method based on synthetic training data. Recent studies [Zoph et al., 2020, Zou et al., 2019] have compared self-training with self-supervised pretraining as a strategy of using

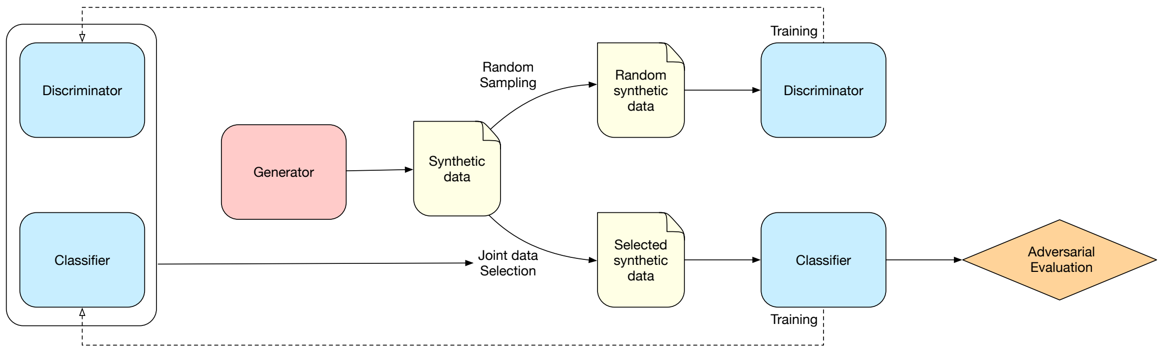


Figure 7-6: The pipeline of the adversarial self-training algorithm includes four steps: 1. synthetic data generation, 2. discriminator training using randomly sampled synthetic and human-labeled training data, 3. data selection with the discriminator and weighting with the pretrained classifier, and 4. updating both generator and classifier with selected and weighted synthetic data.

additional unlabeled human-generated data to create new real-data, pseudo-label training cases. In contrast to previous research, we propose a method that only uses a restricted amount of real data as the initial training set and generates synthetic data-label pairs for self-training. Given a training set of a specific task, we pretrain a synthetic data generator and a task solver model, for example, a classifier for a task in GLUE [Wang et al., 2018] and AdversarialGLUE [Wang et al., 2021]. We also randomly initialize a discriminator as GANs for predicting the source of textual inputs. The models are trained in a multi-epoch iterative pipeline, each epoch of which contains four steps: 1) generate synthetic data based on given labels, 2) update the task model with high-quality synthetic selected by the discriminator and the pretrained task model, 3) sample synthetic and real data to update the discriminator, and 4) update the text generator with high-quality synthetic data selected by the updated discriminator. The proposed self-training pipeline is illustrated in Figure 7-6.

We designed experiments to compare the traditional pretraining-fine-tuning approach and our self-training pipeline by evaluating the learned models on natural language understanding benchmarks, GLUE, and AdversarialGLUE. We found the pretrained data generators cannot improve the training of task models since the synthetic data they generate is less diverse and natural than human-generated texts. However, after adversarial training, the generator is encouraged to output training cases on which the pretrained task model fails to make confident predictions. By running this pipeline iteratively, our system challenges the task model with training cases of gradually increased difficulty and diversity. Experiments show that the self-training pipeline significantly boosted the performance of task models in both regular and adversarial evaluation settings. On the SST-2 task of the AdversarialGLUE benchmark, the performance improvement is as high as 10%.

7.3.1 Related Work

Pretraining large-scale neural language models in a self-supervised manner on large corpora and fine-tuning on task-specific training data has been a popular method recently for both language understanding [Devlin et al., 2018, Liu et al., 2019, Yang et al., 2019, Clark et al., 2020, He et al., 2020, Joshi et al., 2020] and generation [Brown et al., 2020b, Lewis et al., 2019a, Raffel et al., 2019, Zhang et al., 2019]. Although achieving state-of-the-art performance in a wide range of tasks, recent studies have found that the pretraining-fine-tuning strategy relies on task-specfic data annotation and the performance is sensitive to adversarial data examples [Blum and Mitchell, 1998, Wang et al., 2021, Jin et al., 2020, Bartolo et al., 2020, Zang et al., 2019, Garg and Ramakrishnan, 2020, Li et al., 2020].

Besides self-supervised pretraining, another strategy to improve models with unlabeled data is self-training [Zoph et al., 2020, Xie et al., 2020, Noroozi et al., 2018, Zou et al., 2019, He et al., 2019, Sachan and Xing, 2018, Shakeri et al., 2020, Bartolo et al., 2021]. Different from pretraining with data augmentation or constructing task-like pretraining cases [Glass et al., 2019], self-training models learn from synthetic task-specific training cases. A typical training case can be modeled as a data-label pair. [Zoph et al., 2020, Xie et al., 2020, Noroozi et al., 2018, Zou et al., 2019] explored training with additional real data and pseudo labels, while [He et al., 2019, Sachan and Xing, 2018, Shakeri et al., 2020, Bartolo et al., 2021] introduced training cases consist of both synthetic data and pseudo labels. These studies suggest that self-training benefits both training robustness and generalization across domains.

Generative adversarial networks (GANs) [Goodfellow, 2016] have been widely applied to generate synthetic data with similar distributions to human-produced data. Because of their discrete nature, a similar idea can only be implemented with a policy gradient to train language generators [Yu et al., 2017]. A previous study also showed that language generators can be optimized for non-differentiable objects with policy gradients [Rennie et al., 2017]. The previous chapter on RGX also found fine-tuning question generators with selected synthetic with weighted self-training benefits self-training for question answering.

7.3.2 Method: Adversarial Self-training

In this chapter, we proposed a multi-agent architecture that shares similarities with GANs despite a different focus and downstream tasks. Our architecture involves three modules: a generator, a discriminator, and a task handler, usually a classifier for GLUE-like tasks. We call the method adversarial self-training (AdvST) since the pipeline shares the generator-discriminator components of the generative adversarial network (GAN) [Goodfellow et al., 2014] architecture.

Modules

Generator ($\mathbb{G}$) We train sequence-to-sequence text generators [Lewis et al., 2019a, Raffel et al., 2019] to produce the synthetic data. Given the training set $\{(x_0^i, x_1^i, y^i)|i \in [0, N]\}$ of a target task, we train a generation model

$$x_a^i = \mathbb{G}(x_b^i, l^i; \theta_{\mathbb{G}})$$

where x_a and x_b are different textual inputs of the task and l^i is the given label. The generator learns to generate a textual input given another input and the label. For example in the natural language inference task (MNLI), a generation model learns to generate the hypothesis given the premise and a "entail" label. For the classification tasks that only take one textual input, for example, sentiment analysis (SST-2), we train the model to generate texts only based on given labels [Wang et al., 2018].

Discriminator ($\mathbb{D}$) As in [Goodfellow et al., 2014], the discriminator is used to distinguish the synthetic texts from human-generated texts. The model learns to assign input texts or text pairs probabilities of being generated by humans,

$$p_h = \mathbb{D}(x_0^1, x_1^1; \theta_{\mathbb{D}})$$

For the SST-2 task, the model only processes one textual input. During the training, we randomly sample synthetic training cases and human-annotated training cases, and train the discriminator model on a mixed training set consisting of sampled training cases.

Classifier ($\mathbb{C}$) We train the classifier for the target task. In our pipeline, the classifier is pretrained on human-labeled training data, and fine-tuned on a new data set constructed by mixing synthetic and the original training samples. In AdversarialGLUE, the classifier acts as a regular classification model that takes sentences or text pairs and outputs predicted classes. The classifier is evaluated on different benchmarks after finetuning.

7.3.3 Synthetic Data Generation Pipeline

We propose a cooperative adversarial self-training framework (CAST) that jointly fine-tunes synthetic data generators and task-specific classifiers. To enable the pipeline, we first pretrain the generator and the classifier with maximum likelihood learning on the training set of a given task as shown in Section 7.3.2. The generator will be fine-tuned for generating better training cases, while the pretrained classifier is used for providing classification entropy in later steps. The classification entropy makes a crucial fine-tuning signal for the generator.

With the pretrained models, we fine-tune the modules for synthetic data generation in an iterative pipeline with multiple epochs. Each epoch of the pipeline contains three steps, including synthetic data generation, discriminator training, and generator finetuning with selective, weighted self-training. The pipeline of our method is illustrated in Figure 7-6.

Step 1. Synthetic data generation. Consider a training set consisting of N

training samples and M labels, we generate M synthetic texts for each training case. In other words, we generate N synthetic texts for each label. As a result, we generate $N \cdot M$ synthetic texts for further selection using the pretrained generation model. For the tasks that only have one input, we generate texts only based on given labels, while for tasks that require models to process model pairs $\{(x_1^i, x_2^i)|_{i=0}^{N-1}\}$, we generate M first-sentences for each second-sentences x_2^i for a fair comparison across different tasks.

Step 2. Discriminator training. We train a discriminator to distinguish between synthetic and human-generated training cases. To train the model, given a human-labeled training set containing N training cases and a synthetic corpus with M data points ($M > N$), we randomly sample N cases from the synthetic set and construct a new training set by mixing the sampled synthetic training cases with the human-generated cases. We then fine-tune a pretrained trained language model on the new training set with a binary classification task.

Step 3. Generator fine-tuning. We classify the synthetic dataset with the trained discriminator and assign "synthetic" and "human-generated" probabilities $p_{syn}^{\mathbb{D}}$ and $p_{real}^{\mathbb{D}}$ to each synthetic case. Since the discriminator is trained with cases sampled from the same synthetic set, only a small subset of the synthetic cases will receive $p_{real}^{\mathbb{D}} > 0.5 > p_{syn}^{\mathbb{D}}$. In addition, we apply the pretrained classifier $\mathbb{C}$ to assign probabilities to each class for synthetic cases. For the i-th synthetic training case (x_1^i, x_2^i, y^i), we denote the probability of the labeled class with $p_i^{\mathbb{C}} = p(y^i|x_1^i, x_2^i; \theta_{\mathbb{C}})$.

The generator is fine-tuned with a selective training set consists of synthetic training cases that satisfy $p_{real}^{i,\mathbb{D}} > 0.5 > p_{syn}^{i,\mathbb{D}}$. We define the loss function of the i-th selected training case as

$$l_i = -p_{real}^{i,\mathbb{D}} \cdot (1 - p_i^{\mathbb{C}}) \cdot \log(x_1^i|x_2^i, y^i; \theta_{\mathbb{G}}) \tag{7.1}$$

Our self-training design assigns larger loss signals to the training cases that have higher $p_{real}^{i,\mathbb{D}}$ and lower $p_i^{\mathbb{C}}$. In other words, a training case receives a strong training signal if the discriminator believes it is a human-generated data point and the classifier is not too confident about the label so that the generator is guided to produce more natural and difficult synthetic training cases.

Iterative fine-tuning. After fine-tuning the generator, we can go back to step 1 and re-do the entire pipeline for more epochs. At epoch t, we further fine-tune the discriminator and generator trained in epoch $t-1$. After the iterative fine-tuning, both generator and discriminator are fixed for the final synthetic data generation for fine-tuning the classifier.

Classifier fine-tuning

After the generator and discriminator fine-tuning is finished, we re-do the synthetic data generation step described above and select human-like synthetic data points $\{(x_1^i, x_2^i, y^i) | p_{real}^{i,\mathbb{D}} > p_{syn}^{i,\mathbb{D}}\}$ using the trained discriminator $\mathbb{D}$. We construct a new training set by mixing and randomly shuffling the selected training cases and the given training set. A pretrained language model is fine-tuned on the newly constructed training set for further evaluation. In this work, we evaluate the fine-tuning performance of different pretrained language models under fully-supervised and few-shot settings and evaluate the models on both standard and adversarial evaluation sets to understand the effect of the proposed method.

7.3.4 Experiments

We evaluate our method on GLUE [Wang et al., 2018] and AdversarialGLUE [Wang et al., 2021] benchmarks. We evaluate the generalization ability of the fine-tuned classifier model to adversarial evaluation examples and assess the influence of our model on the standard evaluation set. We compare the performance of our model on SST-2, QQP, QNLI, and RTE tasks where both standard and adversarial evaluation sets are provided [Wang et al., 2021]. We fine-tune DeBERTa [He et al., 2020] on GLUE training sets, which is the state-of-the-art model on AdversarialGLUE according to [Wang et al., 2021], and evaluate by averaging three separate experiments. The performance is shown in Tables 7.1 and 7.2. We compare our method with different baseline data augmentation methods, including SSMBA [Ng et al., 2020] and SeqGAN

Dataset	Accuracy	Baseline	SSMBA	SeqGAN	AdvST
SST-2	Avg.	55.10	59.04	56.08	**62.33**
	Std.	4.51	3.65	4.68	**2.55**
RTE	Avg.	63.51	69.53	71.19	**73.66**
	Std.	10.73	3.58	4.34	**2.67**
QQP	Avg.	55.98	59.86	60.72	**64.75**
	Std.	7.06	6.34	5.23	**4.37**
QNLI	Avg.	51.89	58.11	57.65	**62.33**
	Std.	5.50	4.25	3.81	**2.61**

Table 7.1: The performance of different training methods on AdversarialGLUE.

Dataset	Accuracy	Baseline	SSMBA	SeqGAN	AdvST
SST-2	Avg.	95.75	95.43	**96.44**	96.32
	Std.	0.55	1.02	0.52	**0.27**
RTE	Avg.	78.04	80.59	79.06	**80.86**
	Std.	3.84	1.53	**0.72**	1.26
QQP	Avg.	90.56	89.72	**90.85**	90.35
	Std.	**0.20**	0.72	0.65	0.79
QNLI	Avg.	92.98	92.69	92.94	**93.26**
	Std.	0.77	0.94	1.27	**0.43**

Table 7.2: The performance of different training methods on GLUE.

[Yu et al., 2017].

The experimental results shown in the tables are summarized based on 5 separate runs. Our method, adversarial self-training (AdvST), achieved the best performance on adversarial tasks, achieving the highest performance and lowest standard deviation, indicating that our method is effective in improving the robustness and stability of language model fine-tuning. Our method also outperforms SSMBA and SeqGAN, which are strong data augmentation and language generation methods. Our method outperforms the baseline Deberta model by more than 10% on adversarial RTE, QQP, and QNLI tasks.

On the evaluation set of the standard GLUE benchmark, we found that SeqGAN sometimes outperforms the AdvST model. The reason is that our AdvST method rewards generating training cases with high entropy, which is not necessary for standard GLUE evaluation. Out of 4 tasks, SeqGAN achieves the highest performance on SST-2 and QQP, and AdvST outperforms all baseline models on RTE and QNLI. While both AdvST and SeqGAN outperform SSMBA, all data augmentation methods outperform the baseline model on standard GLUE tasks not as significantly as in the adversarial experiments because the data annotation in standard GLUE is regular. The result indicates that our method, as well as other data augmentation methods, are more effective when the evaluation data is more difficult.

7.4 Chapter Summary

In this chapter, we demonstrate generative self-training methods for different natural language understanding tasks. We applied an improved RGX model with a higher answer entity hit rate for both supervised and unsupervised fact checking tasks, achieving as much as 20% improvement on some evaluation tasks. In addition to question answering, we develop a generative data augmentation method, AdvST, to improve both the accuracy and robustness of NLU models. Experiments show that the proposed method can improve the standard model fine-tuning method by higher than 10% accuracy.

Chapter 8

Conclusions

In this thesis, we have proposed several different self-training methods for natural language processing (NLP), showing that the training of neural language models can be improved by different strategies of synthetic data generation, including soft pseudo label generation, prototypical label embedding generation, and synthetic training text generation.

In Chapter 3, we proposed a self-training method based on pseudo-labels for interpretable dialog response retrieval. We train the dialog encoder to output both sequence-level coherence scores and token-level evidence scores. Since there are no annotated keywords as evidence, we proposed a pseudo-labeling method and encouraged the model to learn from the generated evidence labels. Experiments show that the joint retrieval-extraction (REX) encoder improves both dialog response selection performance compared with existing state-of-the-art models, but also improves the interpretability by providing the evidence keywords that make the retrieved utterances good responses. REX significantly outperforms the cross-encoder baseline and achieves the new SOTA performance on the DSTC7 Track 1 challenge, without increasing the number of trainable parameters. Our analysis suggests that the proposed unsupervised extraction training leads to more improvement than simple poolings. The visualizations of the extraction results demonstrate that the model attends to evidence keywords helping determine whether the candidate is a good response, and thus enhance interpretability.

In Chapter 4, we proposed a novel dialog management model, prototypical Q network, for supervised and few-shot dialog policy learning using prototypical label embeddings pretrained in a self-supervised manner. We apply this model in the area of automatic conversational diagnosis. Experiments showed that the ProtoQNs outperform the DQN model in both supervised and few-shot settings. In the supervised setting, ProtoQNs achieve results comparable to SOTA without using domain-specific features. As for the few-shot experiment, ProtoQN learns new diseases using a few training samples without forgetting previously learned symptom knowledge and achieves the SOTA performance. The model also shows less degradation as we inject noise into a conversation. Our study suggests that modeling real conversations directly reinforces simulator-based dialog policy learning. Embeddings of dialog actions are shareable among tasks (diseases, in our case) and benefit the fast adaptation to new ones. Here we show promising results in the medical domain. In the future, we will investigate more adaptive models as well as different domains and corpora toward the goal of modeling new dialog tasks better and with fewer examples.

In Chapter 6, we extended self-training methods from the pseudo label and label embeddings to training with textual synthetic data points, which requires task-aware natural language generation models. We proposed a cooperative self-training framework, RGX, consisting of an answer entity Recognizer, a question Generator, and an answer eXtractor, for question generation and answering. We also introduced in the framework an expectation-maximization method that measures the quality of generated questions for reinforced finetuning of the question generation models. Experiments show that RGX significantly outperforms pretrained and self-trained model baselines while adapted to unseen domains, suggesting that RGX is a promising framework for making extractive question answering methods more scalable and less dependent on human annotation.

In Chapter 7, we explored synthetic data generation and self-training methods for different NLU tasks, including fact checking, sentiment analysis, and other language entailment tasks. Experiments showed that the synthetic QA pairs can be used for unsupervised fact checking, which outperforms supervised baselines. We also proposed adversarial self-training, which is also a generative data augmentation method that learns to augment the original training set by learning adversarial and prediction entropy signals. The proposed method significantly outperforms all baseline models, including standard model finetuning, masked language model-based data augmentation method, and existing adversarial text generation methods.

The long-term goal of this research is automatically generating natural training data for different tasks and modalities. Systems in this direction will benefit tasks where high-quality, human-annotated training data is difficult to obtain because of different limitations, including costs, expertise, privacy, and other reasons. With such systems, users of machine learning models do not have to conduct costly data annotation or send sensitive data to third parties. It is also interesting to evaluate if self-training models can mitigate biases in human data annotation. In the future, we will propose and analyze self-training methods in different domains and tasks, to build fully automatic, private machine learning pipelines.

Bibliography

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

[Barrón-Cedeno et al., 2010] Barrón-Cedeno, A., Rosso, P., Agirre, E., and Labaka, G. (2010). Plagiarism detection across distant language pairs. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 37–45.

[Bartolo et al., 2020] Bartolo, M., Roberts, A., Welbl, J., Riedel, S., and Stenetorp, P. (2020). Beat the ai: Investigating adversarial human annotation for reading comprehension. *Transactions of the Association for Computational Linguistics*, 8:662–678.

[Bartolo et al., 2021] Bartolo, M., Thrush, T., Jia, R., Riedel, S., Stenetorp, P., and Kiela, D. (2021). Improving question answering model robustness with synthetic adversarial data generation. *arXiv preprint arXiv:2104.08678*.

[Bender et al., 2003] Bender, O., Och, F. J., and Ney, H. (2003). Maximum entropy models for named entity recognition. In Daelemans, W. and Osborne, M., editors, *Proceedings of CoNLL-2003*, pages 148–151. Edmonton, Canada.

[Bengio, 2012] Bengio, Y. (2012). Deep learning of representations for unsupervised and transfer learning. In *Proceedings of ICML workshop on unsupervised and transfer learning*, pages 17–36. JMLR Workshop and Conference Proceedings.

[Bengio et al., 2000] Bengio, Y., Ducharme, R., and Vincent, P. (2000). A neural probabilistic language model. *Advances in Neural Information Processing Systems*, 13.

[Bengio et al., 2003] Bengio, Y., Ducharme, R., Vincent, P., and Jauvin, C. (2003). A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155.

[Blum and Mitchell, 1998] Blum, A. and Mitchell, T. (1998). Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 92–100.

[Bowman et al., 2015] Bowman, S. R., Angeli, G., Potts, C., and Manning, C. D. (2015). A large annotated corpus for learning natural language inference. *arXiv preprint arXiv:1508.05326.*

[Brown et al., 2020a] Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. (2020a). Language models are few-shot learners.

[Brown et al., 2020b] Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. (2020b). Language models are few-shot learners. *arXiv preprint arXiv:2005.14165.*

[Chen et al., 2017a] Chen, D., Fisch, A., Weston, J., and Bordes, A. (2017a). Reading wikipedia to answer open-domain questions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1870–1879.

[Chen et al., 2017b] Chen, D., Fisch, A., Weston, J., and Bordes, A. (2017b). Reading Wikipedia to answer open-domain questions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1870–1879, Vancouver, Canada. Association for Computational Linguistics.

[Chen and Wang, 2019] Chen, Q. and Wang, W. (2019). Sequential attention-based network for noetic end-to-end response selection. *arXiv preprint arXiv:1901.02609.*

[Clark et al., 2020] Clark, K., Luong, M.-T., Le, Q. V., and Manning, C. D. (2020). Electra: Pre-training text encoders as discriminators rather than generators. *arXiv preprint arXiv:2003.10555.*

[Dai et al., 2019] Dai, Z., Yang, Z., Yang, Y., Carbonell, J. G., Le, Q., and Salakhutdinov, R. (2019). Transformer-xl: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2978–2988.

[De Vries et al., 2017] De Vries, H., Strub, F., Chandar, S., Pietquin, O., Larochelle, H., and Courville, A. (2017). Guesswhat?! visual object discovery through multi-modal dialogue. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5503–5512.

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

[Dinan et al., 2019] Dinan, E., Logacheva, V., Malykh, V., Miller, A., Shuster, K., Urbanek, J., Kiela, D., Szlam, A., Serban, I., Lowe, R., et al. (2019). The second conversational intelligence challenge (convai2). *arXiv preprint arXiv:1902.00098.*

[Dua et al., 2019] Dua, D., Wang, Y., Dasigi, P., Stanovsky, G., Singh, S., and Gardner, M. (2019). DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

[Duan et al., 2017] Duan, N., Tang, D., Chen, P., and Zhou, M. (2017). Question generation for question answering. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 866–874.

[Fazel-Zarandi et al., 2017] Fazel-Zarandi, M., Li, S.-W., Cao, J., Casale, J., Henderson, P., Whitney, D., and Geramifard, A. (2017). Learning robust dialog policies in noisy environments. In *Workshop on Conversational AI: Today's Practice and Tomorrow's Potential, NeurIPS*.

[Fei et al., 2020] Fei, H., Ren, Y., and Ji, D. (2020). Retrofitting structure-aware transformer language model for end tasks. *arXiv preprint arXiv:2009.07408*.

[Feng et al., 2020] Feng, F., Yang, Y.-F., Cer, D. M., Arivazhagan, N., and Wang, W. (2020). Language-agnostic bert sentence embedding. *ArXiv*, abs/2007.01852.

[Finn et al., 2017] Finn, C., Abbeel, P., and Levine, S. (2017). Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pages 1126–1135. PMLR.

[Fisch et al., 2019] Fisch, A., Talmor, A., Jia, R., Seo, M., Choi, E., and Chen, D. (2019). Mrqa 2019 shared task: Evaluating generalization in reading comprehension. In *Proceedings of the 2nd Workshop on Machine Reading for Question Answering*, pages 1–13.

[Gao et al., 2019] Gao, S., Sethi, A., Agarwal, S., Chung, T., and Hakkani-Tur, D. (2019). Dialog state tracking: A neural reading comprehension approach. In *Proceedings of the 20th Annual SIGdial Meeting on Discourse and Dialogue*, pages 264–273, Stockholm, Sweden. Association for Computational Linguistics.

[Garg and Ramakrishnan, 2020] Garg, S. and Ramakrishnan, G. (2020). Bae: Bert-based adversarial examples for text classification. *arXiv preprint arXiv:2004.01970*.

[Glass et al., 2019] Glass, M., Gliozzo, A., Chakravarti, R., Ferritto, A., Pan, L., Bhargav, G., Garg, D., and Sil, A. (2019). Span selection pre-training for question answering. *arXiv preprint arXiv:1909.04120*.

[Goodfellow, 2016] Goodfellow, I. (2016). Nips 2016 tutorial: Generative adversarial networks. *arXiv preprint arXiv:1701.00160*.

[Goodfellow et al., 2014] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. *Advances in neural information processing systems*, 27:2672–2680.

[Graves et al., 2014] Graves, A., Wayne, G., and Danihelka, I. (2014). Neural turing machines. *arXiv preprint arXiv:1410.5401.*

[Gu et al., 2018] Gu, J.-C., Ling, Z.-H., Ruan, Y.-P., and Liu, Q. (2018). Building sequential inference models for end-to-end response selection. *arXiv preprint arXiv:1812.00686.*

[Gunasekara et al., 2019] Gunasekara, C., Kummerfeld, J. K., Polymenakos, L., and Lasecki, W. (2019). DSTC7 task 1: Noetic end-to-end response selection. In *Proceedings of the First Workshop on NLP for Conversational AI.*

[Guu et al., 2020] Guu, K., Lee, K., Tung, Z., Pasupat, P., and Chang, M.-W. (2020). Realm: Retrieval-augmented language model pre-training. *arXiv preprint arXiv:2002.08909.*

[Havrylov and Titov, 2017] Havrylov, S. and Titov, I. (2017). Emergence of language with multi-agent games: Learning to communicate with sequences of symbols. In *Advances in neural information processing systems*, pages 2149–2159.

[He et al., 2019] He, J., Gu, J., Shen, J., and Ranzato, M. (2019). Revisiting self-training for neural sequence generation. *arXiv preprint arXiv:1909.13788.*

[He et al., 2020] He, P., Liu, X., Gao, J., and Chen, W. (2020). Deberta: Decoding-enhanced bert with disentangled attention. *arXiv preprint arXiv:2006.03654.*

[Henderson et al., 2019a] Henderson, M., Casanueva, I., Mrkšić, N., Su, P.-H., Wen, T.-H., and Vulić, I. (2019a). Convert: Efficient and accurate conversational representations from transformers. *arXiv preprint arXiv:1911.03688.*

[Henderson et al., 2019b] Henderson, M., Vulić, I., Gerz, D., Casanueva, I., Budzianowski, P., Coope, S., Spithourakis, G., Wen, T.-H., Mrkšić, N., and Su, P.-H. (2019b). Training neural response selection for task-oriented dialogue systems. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5392–5404, Florence, Italy. Association for Computational Linguistics.

[Hochreiter and Schmidhuber, 1997] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

[Humeau et al., 2019a] Humeau, S., Shuster, K., Lachaux, M.-A., and Weston, J. (2019a). Poly-encoders: Transformer architectures and pre-training strategies for fast and accurate multi-sentence scoring. *arXiv preprint arXiv:1905.01969.*

[Humeau et al., 2019b] Humeau, S., Shuster, K., Lachaux, M.-A., and Weston, J. (2019b). Poly-encoders: Transformer architectures and pre-training strategies for fast and accurate multi-sentence scoring. *arXiv: Computation and Language.*

[Irving and Fraser, 1992] Irving, R. W. and Fraser, C. B. (1992). Two algorithms for the longest common subsequence of three (or more) strings. In *Annual Symposium on Combinatorial Pattern Matching*, pages 214–229. Springer.

[Jaccard, 1901] Jaccard, P. (1901). Étude comparative de la distribution florale dans une portion des alpes et des jura. *Bull Soc Vaudoise Sci Nat*, 37:547–579.

[Jia et al., 2021] Jia, R., Lewis, M., and Zettlemoyer, L. (2021). Question answering infused pre-training of general-purpose contextualized representations. *arXiv preprint arXiv:2106.08190*.

[Jin et al., 2020] Jin, D., Jin, Z., Zhou, J. T., and Szolovits, P. (2020). Is bert really robust? a strong baseline for natural language attack on text classification and entailment. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 8018–8025.

[Joshi et al., 2020] Joshi, M., Chen, D., Liu, Y., Weld, D. S., Zettlemoyer, L., and Levy, O. (2020). Spanbert: Improving pre-training by representing and predicting spans. *Transactions of the Association for Computational Linguistics*, 8:64–77.

[Kakade, 2001] Kakade, S. M. (2001). A natural policy gradient. *Advances in neural information processing systems*, 14:1531–1538.

[Karpukhin et al., 2020a] Karpukhin, V., Oğuz, B., Min, S., Lewis, P., Wu, L., Edunov, S., Chen, D., and Yih, W.-t. (2020a). Dense passage retrieval for open-domain question answering. *arXiv preprint arXiv:2004.04906*.

[Karpukhin et al., 2020b] Karpukhin, V., Oguz, B., Min, S., Lewis, P., Wu, L., Edunov, S., Chen, D., and Yih, W.-t. (2020b). Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781, Online. Association for Computational Linguistics.

[Kembhavi et al., 2017] Kembhavi, A., Seo, M., Schwenk, D., Choi, J., Farhadi, A., and Hajishirzi, H. (2017). Are you smarter than a sixth grader? textbook question answering for multimodal machine comprehension. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[Kingma and Ba, 2015] Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. In Bengio, Y. and LeCun, Y., editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

[Klein and Nabi, 2019] Klein, T. and Nabi, M. (2019). Learning to answer by learning to ask: Getting the best of gpt-2 and bert worlds. *arXiv preprint arXiv:1911.02365*.

[Koch et al., 2015] Koch, G., Zemel, R., and Salakhutdinov, R. (2015). Siamese neural networks for one-shot image recognition. In *ICML deep learning workshop*, volume 2. Lille.

[Kwiatkowski et al., 2019] Kwiatkowski, T., Palomaki, J., Redfield, O., Collins, M., Parikh, A., Alberti, C., Epstein, D., Polosukhin, I., Devlin, J., Lee, K., et al. (2019). Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466.

[Lai et al., 2017] Lai, G., Xie, Q., Liu, H., Yang, Y., and Hovy, E. (2017). RACE: Large-scale reading comprehension dataset from examinations. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing.*

[Lazaridou et al., 2016] Lazaridou, A., Peysakhovich, A., and Baroni, M. (2016). Multi-agent cooperation and the emergence of (natural) language. *arXiv preprint arXiv:1612.07182.*

[Lee et al., 2020] Lee, D. B., Lee, S., Jeong, W. T., Kim, D., and Hwang, S. J. (2020). Generating diverse and consistent qa pairs from contexts with information-maximizing hierarchical conditional vaes. *arXiv preprint arXiv:2005.13837.*

[Lee et al., 2019] Lee, K., Chang, M.-W., and Toutanova, K. (2019). Latent retrieval for weakly supervised open domain question answering. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6086–6096.

[Lee et al., 2017] Lee, K., He, L., Lewis, M., and Zettlemoyer, L. (2017). End-to-end neural coreference resolution. *arXiv preprint arXiv:1707.07045.*

[Lester et al., 2021] Lester, B., Al-Rfou, R., and Constant, N. (2021). The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691.*

[Levy et al., 2017] Levy, O., Seo, M., Choi, E., and Zettlemoyer, L. (2017). Zero-shot relation extraction via reading comprehension. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017).*

[Lewis et al., 2019a] Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., and Zettlemoyer, L. (2019a). Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461.*

[Lewis et al., 2019b] Lewis, P., Denoyer, L., and Riedel, S. (2019b). Unsupervised question answering by cloze translation. *arXiv preprint arXiv:1906.04980.*

[Lewis et al., 2021] Lewis, P., Wu, Y., Liu, L., Minervini, P., Küttler, H., Piktus, A., Stenetorp, P., and Riedel, S. (2021). Paq: 65 million probably-asked questions and what you can do with them. *arXiv preprint arXiv:2102.07033.*

[Li and Jurafsky, 2016] Li, J. and Jurafsky, D. (2016). Mutual information and diverse decoding improve neural machine translation. *arXiv preprint arXiv:1601.00372.*

[Li et al., 2020] Li, L., Ma, R., Guo, Q., Xue, X., and Qiu, X. (2020). Bert-attack: Adversarial attack against bert using bert. *arXiv preprint arXiv:2004.09984.*

[Li and Liang, 2021] Li, X. L. and Liang, P. (2021). Prefix-Tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (ACL/IJCNLP 2021)*.

[Lin, 1992] Lin, L.-J. (1992). Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine learning*, 8(3-4):293–321.

[Lipton et al., 2016] Lipton, Z. C., Gao, J., Li, L., Li, X., Ahmed, F., and Deng, L. (2016). Efficient exploration for dialogue policy learning with bbq networks & replay buffer spiking. *arXiv preprint arXiv:1608.05081*, 3.

[Liu et al., 2018] Liu, B., Tur, G., Hakkani-Tur, D., Shah, P., and Heck, L. (2018). Dialogue learning with human teaching and feedback in end-to-end trainable task-oriented dialogue systems. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2060–2069.

[Liu et al., 2020] Liu, D., Gong, Y., Fu, J., Yan, Y., Chen, J., Lv, J., Duan, N., and Zhou, M. (2020). Tell me how to ask again: Question data augmentation with controllable rewriting in continuous space. *arXiv preprint arXiv:2010.01475*.

[Liu et al., 2021] Liu, P., Yuan, W., Fu, J., Jiang, Z., Hayashi, H., and Neubig, G. (2021). Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing.

[Liu et al., 2019] Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. (2019). Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

[Luo et al., 2019a] Luo, H., Jiang, L., Belinkov, Y., and Glass, J. (2019a). Improving neural language models by segmenting, attending, and predicting the future. *arXiv preprint arXiv:1906.01702*.

[Luo et al., 2019b] Luo, H., Mohtarami, M., Glass, J. R., Krishnamurthy, K., and Richardson, B. (2019b). Integrating video retrieval and moment detection in a unified corpus for video question answering. In *INTERSPEECH*, pages 599–603.

[Mazaré et al., 2018] Mazaré, P.-E., Humeau, S., Raison, M., and Bordes, A. (2018). Training millions of personalized dialogue agents. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2775–2779, Brussels, Belgium. Association for Computational Linguistics.

[Merity et al., 2017] Merity, S., Keskar, N. S., and Socher, R. (2017). Regularizing and optimizing lstm language models. *arXiv preprint arXiv:1708.02182*.

[Mikolov et al., 2013a] Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013a). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

[Mikolov et al., 2013b] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013b). Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 26:3111–3119.

[Mnih et al., 2013] Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. (2013). Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

[Mohtarami et al., 2018] Mohtarami, M., Baly, R., Glass, J., Nakov, P., Màrquez, L., and Moschitti, A. (2018). Automatic stance detection using end-to-end memory networks. *arXiv preprint arXiv:1804.07581*.

[Mordatch and Abbeel, 2018] Mordatch, I. and Abbeel, P. (2018). Emergence of grounded compositional language in multi-agent populations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

[Nadeem et al., 2019] Nadeem, M., Fang, W., Xu, B., Mohtarami, M., and Glass, J. (2019). Fakta: An automatic end-to-end fact checking system. *arXiv preprint arXiv:1906.04164*.

[Navarro, 2001] Navarro, G. (2001). A guided tour to approximate string matching. *ACM computing surveys (CSUR)*, 33(1):31–88.

[Ng et al., 2020] Ng, N., Cho, K., and Ghassemi, M. (2020). Ssmba: Self-supervised manifold based data augmentation for improving out-of-domain robustness. *arXiv preprint arXiv:2009.10195*.

[Noroozi et al., 2018] Noroozi, M., Vinjimoor, A., Favaro, P., and Pirsiavash, H. (2018). Boosting self-supervised learning via knowledge transfer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9359–9367.

[Papineni et al., 2001] Papineni, K. A., Roukos, S., and Ward, R. T. (2001). Natural language task-oriented dialog manager and method. US Patent 6,246,981.

[Peng et al., 2017] Peng, B., Li, X., Li, L., Gao, J., Celikyilmaz, A., Lee, S., and Wong, K.-F. (2017). Composite task-completion dialogue policy learning via hierarchical deep reinforcement learning. *arXiv preprint arXiv:1704.03084*.

[Pennington et al., 2014] Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

[Peters et al., 2018a] Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018a). Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

[Peters et al., 2018b] Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018b). Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association* *for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

[Puri et al., 2020] Puri, R., Spring, R., Patwary, M., Shoeybi, M., and Catanzaro, B. (2020). Training question answering models from synthetic data. *arXiv preprint arXiv:2002.09599*.

[Qian et al., 2004] Qian, G., Sural, S., Gu, Y., and Pramanik, S. (2004). Similarity between euclidean and cosine angle distance for nearest neighbor queries. In *Proceedings of the 2004 ACM symposium on Applied computing*, pages 1232–1237.

[Qian and Yu, 2019] Qian, K. and Yu, Z. (2019). Domain adaptive dialog generation via meta learning. *arXiv preprint arXiv:1906.03520*.

[Radford et al., 2019a] Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. (2019a). Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

[Radford et al., 2019b] Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. (2019b). Language models are unsupervised multitask learners.

[Raffel et al., 2019] Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. (2019). Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*.

[Rajpurkar et al., 2016] Rajpurkar, P., Zhang, J., Lopyrev, K., and Liang, P. (2016). Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.

[Ramadan et al., 2018] Ramadan, O., Budzianowski, P., and Gasic, M. (2018). Large-scale multi-domain belief tracking with knowledge sharing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 432–437.

[Rennie et al., 2017] Rennie, S. J., Marcheret, E., Mroueh, Y., Ross, J., and Goel, V. (2017). Self-critical sequence training for image captioning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7008–7024.

[Riemer et al., 2018] Riemer, M., Cases, I., Ajemian, R., Liu, M., Rish, I., Tu, Y., and Tesauro, G. (2018). Learning to learn without forgetting by maximizing transfer and minimizing interference. *arXiv preprint arXiv:1810.11910*.

[Rosenberg et al., 2005] Rosenberg, C., Hebert, M., and Schneiderman, H. (2005). Semi-supervised self-training of object detection models.

[Routley and Meyer, 1973] Routley, R. and Meyer, R. (1973). The semantics of entailment. In *Studies in Logic and the Foundations of Mathematics*, volume 68, pages 199–243. Elsevier.

[Sachan and Xing, 2018] Sachan, M. and Xing, E. (2018). Self-training for jointly learning to ask and answer questions. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 629–640.

[Saha et al., 2018] Saha, A., Aralikatte, R., Khapra, M. M., and Sankaranarayanan, K. (2018). DuoRC: Towards complex language understanding with paraphrased reading comprehension. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*.

[Sanh et al., 2021] Sanh, V., Webson, A., Raffel, C., Bach, S. H., Sutawika, L., Alyafeai, Z., Chaffin, A., Stiegler, A., Scao, T. L., Raja, A., et al. (2021). Multitask prompted training enables zero-shot task generalization. *arXiv preprint arXiv:2110.08207*.

[Scheffler and Young, 2002] Scheffler, K. and Young, S. (2002). Automatic learning of dialogue strategy using dialogue simulation and reinforcement learning. In *Proceedings of the second international conference on Human Language Technology Research*, pages 12–19. Morgan Kaufmann Publishers Inc.

[Shakeri et al., 2020] Shakeri, S., Santos, C. N. d., Zhu, H., Ng, P., Nan, F., Wang, Z., Nallapati, R., and Xiang, B. (2020). End-to-end synthetic data generation for domain adaptation of question answering systems. *arXiv preprint arXiv:2010.06028*.

[Shazeer and Stern, 2018] Shazeer, N. and Stern, M. (2018). Adafactor: Adaptive learning rates with sublinear memory cost. In *International Conference on Machine Learning*.

[Shen et al., 2017] Shen, Y., Lin, Z., Huang, C.-W., and Courville, A. (2017). Neural language modeling by jointly learning syntax and lexicon. *arXiv preprint arXiv:1711.02013*.

[Shen et al., 2018] Shen, Y., Lin, Z., Jacob, A. P., Sordoni, A., Courville, A., and Bengio, Y. (2018). Straight to the tree: Constituency parsing with neural syntactic distance. *arXiv preprint arXiv:1806.04168*.

[Shen et al., 2020] Shen, Y., Tay, Y., Zheng, C., Bahri, D., Metzler, D., and Courville, A. (2020). Structformer: Joint unsupervised induction of dependency and constituency structure from masked language modeling. *arXiv preprint arXiv:2012.00857*.

[Shuster et al., 2020a] Shuster, K., Smith, E. M., Ju, D., and Weston, J. (2020a). Multi-modal open-domain dialogue. *arXiv preprint arXiv:2010.01082*.

[Shuster et al., 2020b] Shuster, K., Urbanek, J., Dinan, E., Szlam, A., and Weston, J. (2020b). Deploying lifelong open-domain dialogue learning. *arXiv preprint arXiv:2008.08076*.

[Silver et al., 2016] Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al. (2016). Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484.

[Silver et al., 2014] Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., and Riedmiller, M. (2014). Deterministic policy gradient algorithms.

[Snell et al., 2017] Snell, J., Swersky, K., and Zemel, R. (2017). Prototypical networks for few-shot learning. In *Advances in neural information processing systems*, pages 4077–4087.

[Socher et al., 2013] Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A. Y., and Potts, C. (2013). Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

[Tan et al., 2018] Tan, C., Sun, F., Kong, T., Zhang, W., Yang, C., and Liu, C. (2018). A survey on deep transfer learning. In *International conference on artificial neural networks*, pages 270–279. Springer.

[Tang et al., 2017] Tang, D., Duan, N., Qin, T., Yan, Z., and Zhou, M. (2017). Question answering and question generation as dual tasks. *arXiv preprint arXiv:1706.02027*.

[Thorne et al., 2018] Thorne, J., Vlachos, A., Christodoulopoulos, C., and Mittal, A. (2018). Fever: a large-scale dataset for fact extraction and verification. *arXiv preprint arXiv:1803.05355*.

[Tsatsaronis et al., 2012] Tsatsaronis, G., Schroeder, M., Paliouras, G., Almirantis, Y., Androutsopoulos, I., Gaussier, E., Gallinari, P., Artieres, T., Alvers, M. R., Zschunke, M., et al. (2012). BioASQ: A challenge on large-scale biomedical semantic indexing and question answering. In *2012 AAAI Fall Symposium Series*.

[Vaswani et al., 2017] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

[Vinyals et al., 2016] Vinyals, O., Blundell, C., Lillicrap, T., Wierstra, D., et al. (2016). Matching networks for one shot learning. In *Advances in neural information processing systems*, pages 3630–3638.

[Wang et al., 2018] Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., and Bowman, S. R. (2018). Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*.

[Wang et al., 2021] Wang, B., Xu, C., Wang, S., Gan, Z., Cheng, Y., Gao, J., Awadallah, A. H., and Li, B. (2021). Adversarial glue: A multi-task benchmark for robustness evaluation of language models. *arXiv preprint arXiv:2111.02840*.

[Wang et al., 2017] Wang, Z., Hamza, W., and Florian, R. (2017). Bilateral multi-perspective matching for natural language sentences. *arXiv preprint arXiv:1702.03814*.

[Wei et al., 2018] Wei, Z., Liu, Q., Peng, B., Tou, H., Chen, T., Huang, X., Wong, K.-F., and Dai, X. (2018). Task-oriented dialogue system for automatic diagnosis. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 201–207.

[Wen et al., 2017a] Wen, T.-H., Miao, Y., Blunsom, P., and Young, S. (2017a). Latent intention dialogue models. In *International Conference on Machine Learning*, pages 3732–3741. PMLR.

[Wen et al., 2017b] Wen, T.-H., Vandyke, D., Mrkšić, N., Gasic, M., Barahona, L. M. R., Su, P.-H., Ultes, S., and Young, S. (2017b). A network-based end-to-end trainable task-oriented dialogue system. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 438–449.

[Williams, 2014] Williams, J. D. (2014). Web-style ranking and slu combination for dialog state tracking. In *15th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, page 282.

[Wolf et al., 2019] Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., et al. (2019). Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.

[Wu et al., 2020] Wu, C.-S., Hoi, S., Socher, R., and Xiong, C. (2020). Tod-bert: Pre-trained natural language understanding for task-oriented dialogues. *ArXiv*, abs/2004.06871.

[Xie et al., 2020] Xie, Q., Luong, M.-T., Hovy, E., and Le, Q. V. (2020). Self-training with noisy student improves imagenet classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10687–10698.

[Xu et al., 2019] Xu, L., Zhou, Q., Gong, K., Liang, X., Tang, J., and Lin, L. (2019). End-to-end knowledge-routed relational dialogue system for automatic diagnosis. *arXiv preprint arXiv:1901.10623*.

[Xu et al., 2020] Xu, S., Semnani, S. J., Campagna, G., and Lam, M. S. (2020). Autoqa: From databases to qa semantic parsers with only synthetic training data. *arXiv preprint arXiv:2010.04806*.

[Yang et al., 2017] Yang, X., Chen, Y.-N., Hakkani-Tür, D., Crook, P., Li, X., Gao, J., and Deng, L. (2017). End-to-end joint learning of natural language understanding and dialogue manager. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5690–5694. IEEE.

[Yang et al., 2019] Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R., and Le, Q. V. (2019). Xlnet: Generalized autoregressive pretraining for language understanding. *arXiv preprint arXiv:1906.08237*.

[Young et al., 2010] Young, S., Gašić, M., Keizer, S., Mairesse, F., Schatzmann, J., Thomson, B., and Yu, K. (2010). The hidden information state model: A practical framework for pomdp-based spoken dialogue management. *Computer Speech & Language*, 24(2):150–174.

[Young et al., 2013] Young, S., Gašić, M., Thomson, B., and Williams, J. D. (2013). Pomdp-based statistical spoken dialog systems: A review. *Proceedings of the IEEE*, 101(5):1160–1179.

[Yu et al., 2017] Yu, L., Zhang, W., Wang, J., and Yu, Y. (2017). Seqgan: Sequence generative adversarial nets with policy gradient. In *Proceedings of the AAAI conference on artificial intelligence*, volume 31.

[Zang et al., 2019] Zang, Y., Qi, F., Yang, C., Liu, Z., Zhang, M., Liu, Q., and Sun, M. (2019). Word-level textual adversarial attacking as combinatorial optimization. *arXiv preprint arXiv:1910.12196*.

[Zhang et al., 2018] Zhang, S., Dinan, E., Urbanek, J., Szlam, A., Kiela, D., and Weston, J. (2018). Personalizing dialogue agents: I have a dog, do you have pets too? In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2204–2213, Melbourne, Australia. Association for Computational Linguistics.

[Zhang et al., 2019] Zhang, Y., Sun, S., Galley, M., Chen, Y.-C., Brockett, C., Gao, X., Gao, J., Liu, J., and Dolan, B. (2019). Dialogpt: Large-scale generative pre-training for conversational response generation. *arXiv preprint arXiv:1911.00536*.

[Zhao and Eskenazi, 2016] Zhao, T. and Eskenazi, M. (2016). Towards end-to-end learning for dialog state tracking and management using deep reinforcement learning. *arXiv preprint arXiv:1606.02560*.

[Zhao et al., 2019] Zhao, T., Xie, K., and Eskenazi, M. (2019). Rethinking action spaces for reinforcement learning in end-to-end dialog agents with latent variable models. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1208–1218.

[Zoph et al., 2020] Zoph, B., Ghiasi, G., Lin, T.-Y., Cui, Y., Liu, H., Cubuk, E. D., and Le, Q. (2020). Rethinking pre-training and self-training. *Advances in neural information processing systems*, 33:3833–3845.

[Zou et al., 2019] Zou, Y., Yu, Z., Liu, X., Kumar, B., and Wang, J. (2019). Confidence regularized self-training. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5982–5991.