University of California
Santa Barbara

Label-efficient Learning in Natural Language Processing

A dissertation submitted in partial satisfaction
of the requirements for the degree

Doctor of Philosophy
in
Computer Science

by

Shiyang Li

Committee in charge:

Professor Xifeng Yan, Chair
Professor William Wang
Professor Lei Li

March 2023

The Dissertation of Shiyang Li is approved.

Professor William Wang

Professor Lei Li

Professor Xifeng Yan, Committee Chair

March 2023

Label-efficient Learning in Natural Language Processing

To my family, for their unconditional love and endless support.

Acknowledgements

My deepest gratitude goes to my advisor, Xifeng Yan, without whom this dissertation is not possible. He brought me to UCSB, and provided me with insightful guidance and endless support in both research and life. His advice, encouragement, criticism and vision deeply shaped my research taste, my way of thinking and much more, and the things I learned from him will continue to be one of the most valuable assets I own.

I would like to thank my committee member William Wang and Lei Li. They provided me with invaluable comments and feedback in every stage of my PhD study. In addition, I want to give special acknowledgements to Yu-Xiang Wang for voluntarily organizing seminars and providing constructive and invaluable feedback in the first year of my PhD study, from which I learned a lot.

I also would like to express my great gratitude to my wonderful mentors and collaborators: Semih Yavuz, Kazuma Hashimoto, Caiming Xiong, Jianshu Chen, Yelong Shen, Yifan Gao, Qingyu Yin, Haoming Jiang, Zheng Li, Chao Zhang, Bing Yin, Na Zhang, Zhiyu Chen, Wenhu Chen, Xiaoyong Jin, Hanwen Zha, Keqian Li, Yao Xuan, Xiyou Zhou, Hong Wang, Jing Qian, Zekun Li, Weizhi Wang and Xinlu Zhang.

What's more, I would like to thank my friends, who brought me unforgettable memory and provided me with various support during my Ph.D. study. Here I want to thank: Chong Liu, Lianke Qin, Guyue Huang, Fuheng Zhao, Zheng Wang, Xin Dong, Lu Han, Jinglei Yang, Yujie Lu, Yue Wei, Ming Yin, Zichang He, Kaiqi Zhang, Yingrui Yang, Kaiqi Zhang, Xuandong Zhao, Buyun Li, Mengye Liu and Wanrong Zhu.

Finally, I would like to thank my parents and sister for their unconditional love and endless support.

Curriculum Vitæ

Shiyang Li

Education

2018-2023 Ph.D. in Computer Science
University of California, Santa Barbara
Advisor: Xifeng Yan

2014-2018 B.Eng. in Computer Science and Technology
Harbin Institute of Technology

Work Experience

Applied Scientist Intern, Amazon — Sep. 2022-Dec. 2022
Location: Remote in Oxnard, CA, USA
Topic: Graph Reasoning for Question Answering

Software Engineer Intern, Meta — Jun. 2022-Sep. 2022
Location: Seattle, WA, USA
Topic: Zero-shot Unseen Class Generalization

Research Intern, Microsoft — Jun. 2021-Sep. 2021
Location: Remote in Santa Barbara, CA, USA
Topic: Online Knowledge Adaptation for Pre-trained Language Model

Research Intern, Salesforce — Jun. 2020-Dec. 2020
Location: Remote in Santa Barbara, CA, USA
Topic: Data Generation for Evaluating and Augmenting Dialogue State Trackers

Research Intern, Tencent AI Lab — Jun. 2019-Sep. 2019
Location: Bellevue, WA, USA
Topic: Knowledge Base Enhanced Pre-training for Commonsense Reasoning

Research Intern, SenseTime — Nov. 2017-Aug. 2018
Location: Beijing, China
Topic: Semantic Segmentation for Autonomous Driving Systems

Publications

17. Time Series as Images: Vision Transformer for Irregularly Sampled Time Series
Zekun Li, **Shiyang Li** and Xifeng Yan
Under Review

16. Graph Reasoning for Question Answering with Triplet Retrieval
Shiyang Li, Yifan Gao, Haoming Jiang, Qingyu Yin, Zheng Li, Xifeng Yan, Chao Zhang and Bing Yin
Under Review

15. Limitations of Language Models in Arithmetic and Symbolic Induction
Jing Qian, Hong Wang, Zekun Li, **Shiyang Li**, Xifeng Yan
Under Review

14. Explanations from Large Language Models Make Small Reasoners Better
Shiyang Li, Jianshu Chen, Yelong Shen, Zhiyu Chen, Xinlu Zhang, Zekun Li, Hong Wang, Jing Qian, Baolin Peng, Yi Mao, Wenhu Chen and Xifeng Yan
Under Review

13. Improving Medical Predictions by Irregular Multimodal Electronic Health Records Modeling
Xinlu Zhang*, **Shiyang Li***, Zhiyu Chen, Xifeng Yan and Linda Petzold (*Equal Contribution)
Under Review

12. ConvFinQA: Exploring the Chain of Numerical Reasoning in Conversational Finance Question Answering
Zhiyu Chen, **Shiyang Li**, Charese Smiley, Zhiqiang Ma, Sameena Shah and William Yang Wang
EMNLP 2022

11. Dialogic: Controllable Dialogue Simulation with In-Context Learning
Zekun Li, Wenhu Chen, **Shiyang Li**, Hong Wang, Jing Qian and Xifeng Yan
Findings of EMNLP 2022

10. Making Something out of Nothing: Building Robust Task-oriented Dialogue Systems from Scratch
Zekun Li, Hong Wang, Alon Albalak, Yingrui Yang, Jing Qian, **Shiyang Li** and Xifeng Yan
Alexa Prize TaskBot Challenge Proceedings 2022

9. Domain Adaptation for Trauma Mortality Prediction in EHRs with Feature Disparity
Xinlu Zhang, **Shiyang Li**, Zhuowei Cheng, Rachael Callcut, Linda Petzold
BIBM 2021

8. Task-adaptive Pre-training and Self-training are Complementary for Natural Language Understanding
Shiyang Li, Semih Yavuz, Wenhu Chen and Xifeng Yan
Findings of EMNLP 2021

7. CoCo: Controllable Counterfactuals for Evaluating Dialogue State Trackers
   Shiyang Li*, Semih Yavuz*, Kazuma Hashimoto, Jia Li, Tong Niu, Nazneen Rajani, Xifeng Yan, Yingbo Zhou and Caiming Xiong (*Equal Contribution)
   ICLR 2021

6. TabFact: A Large-scale Dataset for Table-based Fact Verification
   Wenhu Chen, Hongmin Wang, Jianshu Chen, Yunkai Zhang, Hong Wang, **Shiyang Li**, Xiyou Zhou and William Yang Wang
   ICLR 2020

5. Teaching Pretrained Models with Commonsense Reasoning: A Preliminary KB-Based Approach
   Shiyang Li, Jianshu Chen and Dian Yu
   KR2ML Workshop at NeurIPS 2019

4. Enhancing the Locality and Breaking the Memory Bottleneck of Transformer on Time Series Forecasting
   Shiyang Li, Xiaoyong Jin, Yao Xuan, Xiyou Zhou, Wenhu Chen, Yu-Xiang Wang and Xifeng Yan
   NeurIPS 2019

3. HierCon: Hierarchical Organization of Technical Documents based on Concepts
   Keqian Li, **Shiyang Li**, Semih Yavuz, Hanwen Zha, Yu Su and Xifeng Yan
   IEEE ICDM 2019

2. Multi-step Deep Autoregressive Forecasting with Latent States
   Xiaoyong Jin, **Shiyang Li**, Yunkai Zhang and Xifeng Yan
   ICML 2019 Workshop on Time Series

1. Towards Understanding Acceleration Tradeoff between Momentum and Asynchrony in Distributed Nonconvex Stochastic Optimization
   Tianyi Liu, **Shiyang Li**, Jianping Shi, Enlu Zhou and Tuo Zhao
   NeurIPS 2018

Abstract

Label-efficient Learning in Natural Language Processing

by

Shiyang Li

Natural language is one of the most common mediums for people to communicate and natural language processing (NLP) is a subfield in artificial intelligence that enables computers to understand and reason human languages. NLP has many important applications in our daily life, including Amazon Alexa, Apple Siri, and Google Translate. However, state-of-the-art NLP models often require a large amount of labeled data to learn. This labeled-data-hungry property brings challenges when applying these methods in many real-world applications since collecting large-scale labeled data is cost-expensive, time-consuming and sometimes even not possible due to privacy restrictions.

In this dissertation, we aim at alleviating the labeled-data-hungry issue in NLP by leveraging additional unlabeled and synthesized data. We begin with leveraging unlabeled data to improve model performance for natural language understanding tasks. We show that two major semi-supervised approaches, namely task-adaptive pre-training and self-training, are complementary and their performance gains can be strongly additive. Then we move our focus on utilizing synthesized data to facilitate model learning. In this line of work, we first develop an algorithm focusing on how to leverage a structured knowledge base (KB) to teach the common sense reasoning capability of pre-trained language models (PLMs). Specifically, we use KB to construct various logical forms and utilize rules to convert these logical forms into multiple-choice question-answer pairs requiring commonsense logical reasoning to refine PLMs. Next, we aim at evaluating and augmenting the dialogue state tracking module, a core component for task-oriented dialogue systems. In particular, we first train a PLM as data generator and then

leverage it to generate additional dialogue data with their labels to evaluate and augment state-of-the-art dialogue state tacking models. Finally, we target improving the reasoning capability of small language models (SLMs) to reduce their gap with large language models (LLMs). Specifically, we utilize LLM prompting to generate explanations and they are used as additional supervision signals to improve SLMs, which are more favorable due to low storage and compute cost in real-world applications.

In the end, we summarize the key findings of this dissertation to improve label efficiency via additional unlabeled and synthesized data, and discuss possible future directions towards the goal of more label-efficient learning in NLP.

Contents

Chapter 1

Introduction

Natural language is one of the most common mediums for people to communicate with each other and serves as the basis of human interaction. When people express themself with natural language, they implicitly or explicitly connect it with abstract or concrete objects, which requires complex understanding and reasoning for efficient and effective communication. On the other hand, there are many scenarios where automatic systems are needed to process natural language. For example, monolingual users in social media require machine translation systems to understand posts with other languages; Social media platforms require systems to detect hate speech automatically for social good; Business' customer service needs intelligent systems to automatically answer users' questions and solve their problems timely to provide better user experience.

Natural language processing (NLP) is a subfield in artificial intelligence that aims at enabling computers to understand and reason complex human language. Due to recent advances in representation learning for NLP [1, 2, 3, 4], many NLP systems [5, 6] have achieved better or comparable performance compared to human in specific tasks [7, 8]. However, state-of-the-art NLP models often require a large amount of labeled data to learn. As an example, Figure 1.1 shows three state-of-the-art NLP models, namely BERT [2], GPT [9] and XLNet [10], have

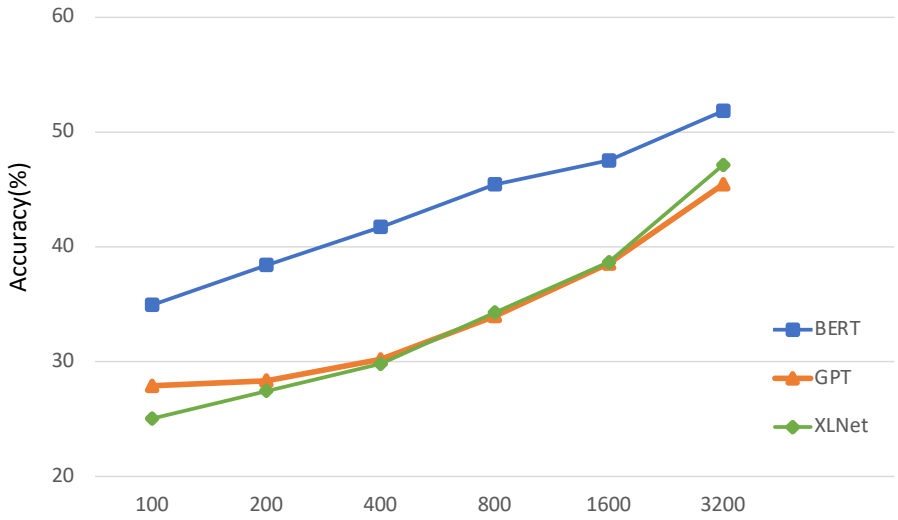

Figure 1.1: Results of finetuning BERT, GPT and XLNet with different sizes of labeled data on CommonsenseQA dataset [8]. Details are deferred into Section 3.4.1

much worse performance when their labeled training data sizes are small in CommonsenseQA, a question answering dataset requiring common sense reasoning ability [8]. On the other hand, collecting large-scale labeled data requires lots of human efforts, which is not only cost-expensive but also time-consuming, and is even not possible due to data privacy constraints in many cases [11] , e.g. medical applications, posing challenges when applying state-of-the-art NLP methods in many real-world applications.

1.1 Improving Label Efficiency via Unlabeled Data

To address aforementioned labeled-data-hungry issue, semi-supervised learning [12] provides a plausible solution by making effective use of freely massive unlabeled data [13, 14]. The goal of semi-supervised learning algorithms is to integrate both labeled and unlabeled data to boost final model performance. Semi-supervised learning has achieved remarkable successes in a variety of tasks [13, 14, 11]. Task-adaptive pre-training (TAPT) [15] and Self-training (ST) [16] have emerged as the major semi-supervised approaches to improve natural language under-

standing (NLU) tasks. In TAPT, pre-trained language models (PLMs) continue self-supervised learning, e.g. mask language modeling [2], on unlabeled data to learn task-specific linguistic characteristics and has shown to be consistently effective across multiple tasks [15]. On the other hand, ST [16] utilizes unlabeled data different from the paradigm of TAPT. In ST, a teacher model is first trained on labeled data and then used to generate pseudo labels for unlabeled data. After that, both labeled and pseudo-labeled data are combined together to train a student model for better performance. The student model can be again assigned as a teacher model and the teacher-student framework can be iterated over multiple rounds. Due to different paradigms of utilizing unlabeled data in TAPT and ST, it is unclear if these two major semi-supervised approaches are complementary for NLU tasks.

In this dissertation, we show that these two major semi-supervised approaches, TAPT and ST, do not always outperform each other, and they are complementary and their performance gains can be strongly additive across various settings. These findings can help future NLP studies build a strong semi-supervised baseline to better learn information buried under freely massive unlabeled data and bridge the gap between the state-of-the-art methods and real-world applications.

1.2 Improving Label Efficiency via Synthesized Data

Another promising direction to alleviate labeled-data-hungry issue is to synthesize high-quality data for model learning. Automatic data synthesis is often faster, cheaper and more scalable than human annotations. In the following section, we detail our proposed three individual approaches to synthesize high-quality data for improving model learning in different scenarios.

Teaching PLMs with commonsense reasoning via data synthesis from KB. Commonsense reasoning is important for many tasks, e.g. text understanding, and is seen as one of the milestones of artificial general intelligence [17]. However, the commonsense reasoning ability of state-of-the-art PLMs, e.g. BERT, GPT and XLNet, are still limited [18, 8]. On the other hand, there exist large-scale commonsense knowledge bases(KBs), e.g. ConceptNet [19], which explicitly record relations between entities and can capture rich world knowledge. Therefore, a key question is how to inject commonsense reasoning knowledge from KBs into PLMs. To achieve this goal, we develop an algorithm that utilizes KB to construct various logical forms and generates multiple-choice question-answer pairs requiring commonsense logical reasoning. These generated multiple-choice question-answer pairs are used as additional pre-training data to refine common sense reasoning capability of PLMs. We show that our proposed method can consistently and significantly improve the commonsense reasoning capability of PLMs, especially in few-shot settings.

Evaluating and augmenting dialogue state tracking via data synthesis from PLM. Task-oriented dialogue systems aim at assisting users to finish specific goals, e.g. restaurant booking. A key component of task-oriented dialogue systems is dialogue state tracking module, which recognizes the user's goal represented as slot-value pairs, e.g. $\{$(*restaurant-food*, *Chinese*),(*restaurant-people*, *6*)$\}$, and often requires a large amount of labeled data to achieve state-of-the-art performance [20]. On the other hand, collecting a large-scale corpus of task-oriented dialogues with human annotations is expensive and time-consuming [21]. To bridge this gap, we first train a model as data generator and then leverage it to generate additional labeled dialogue data. After that we filter out degraded dialogue data and utilize generated labeled data after filtering to evaluate and augment state-of-the-art dialogue state tracking models. We show that our method can consistently improve state-of-the-art dialogue state tacking models in various settings.

Improving SLM reasoning via explanation synthesis from LLM. Large language models (LLMs) [22, 23, 24, 25, 26, 27] have achieved remarkable successes in reasoning tasks [28]. However, their strong reasoning abilities only emerge when they scale to dozens or hundreds of billions of parameters [29], making it expensive to deploy them in a large scale for real-world applications [28]. On the other hand, small language models (SLMs) have low cost in both storage and computation but worse performance on complicated reasoning tasks [29, 30], especially when training data sizes are limited [31]. To bridge this gap, we propose to utilize explanations generated from LLM [28, 32] to improve SLM reasoning capability. Specifically, we first utilize several examples with human-written explanations as demonstrations for LLM and then generate explanations for the whole *training* set. Then we adopt a multi-task learning setup to utilize the LLM-generated explanations to facilitate SLM to acquire strong reasoning power along with explanation generation ability. We show that our framework can consistently and significantly improve the reasoning capability of SLMs, especially in few-shot settings.

1.3 Summary

In this section, we briefly summarize our contributions toward the goal of label-efficient learning in NLP. Our contributions fall into two categories (1) label-efficient learning via making use of freely massive unlabeled data (2) label-efficient learning via data synthesis.

For label-efficient learning via unlabeled data, in [33], we study two major semi-supervised approaches for NLU tasks, TAPT and ST, and find that they are complementary for NLU tasks and their performance gains can be strongly additive across various settings. We hope these findings can help future NLP studies build a strong semi-supervised baseline to better learn information within unlabeled data.

For label-efficient learning via data synthesis, we study several different techniques to synthesize data for improving model learning. In [31], we propose to leverage KB to generate

large-scale commonsense reasoning question answering dataset to refine PLMs. In [34], we propose to train a PLM to synthesize labeled data, which is used to evaluate and augment dialogue state tracking module. In [35], we propose a framework to leverage explanations generated from LLM to improve SLM reasoning capability.

The rest of this dissertation is organized as follows. Chapter 2 studies the complementarity of TAPT and ST for NLU tasks. Chapter 3 presents our method to leverage KB to improve commonsense reasoning capability of PLMs. Chapter 4 discusses our approach for evaluating and augmenting dialogue state tracking module. Chapter 5 introduces our framework to utilize explanations from LLM to improve reasoning ability of SLMs. Chapter 6 summarizes key findings of our work and discusses possible future directions.

Chapter 2

Complementarity of Pre-training and Self-training with Unlabeled Data for NLU

2.1 Introduction

Deep neural networks [36] often require large amounts of labeled data to achieve state-of-the-art performance [37]. However, acquiring high-quality annotations is both time-consuming and cost-expensive, which inspires research on methods that can exploit unlabeled data to improve performance [38]. Pre-trained language models like BERT [2], RoBERTa [39] and T5 [40] can learn general language understanding abilities from large-scale unlabeled corpora and have reduced this annotation cost. In this paradigm, large neural networks are first pre-trained on massive amounts of unlabeled data in a self-supervised manner and then finetuned on large amount of labeled data for specific downstream tasks, which has led to large improvements for natural language understanding on standard benchmarks [41, 7]. However, their success still relies on large amount of data during finetuning stage. For example, [20] shows that BERT only achieves 6.4% joint goal accuracy with 1% finetuning data for dialogue state tracking task, a core component of task-oriented dialogue systems, making it far behind its full counterpart

45.6%. This data-intensive finetuning poses several challenges for many real-world applications, where collecting large amount of labeled data is not only cost-expensive and time-consuming, but also infeasible sometimes due to data access and privacy constraints [11].

Semi-supervised learning [12] provides a plausible solution to address aforementioned data hungry issue by making effective use of freely available unlabeled data. One of the most popular semi-supervised learning algorithms is self-training [16]. In self-training, a teacher model is first trained on available labeled data and then used to generate pseudo labels for unlabeled data. The original hand-annotated labeled data and the pseudo-labeled data are combined together to train a student model. The student model is assigned as a teacher model in next round and the teacher-student training procedure is repeated until convergence or reaching maximum rounds. Self-training utilizes unlabeled data in a *task-specific* way during pseudo labeling process [42] and has been successfully applied to a variety of tasks, including image recognition [37, 43], automatic speech recognition [44], text classification [45, 46], sequence labeling [11] and neural machine translation [38].

Recently, task-adaptive pre-training (TAPT) [15] has been further proposed, which can adapt pre-trained language models, e.g. BERT and RoBERTa, to unlabeled in-domain training set to improve performance [15]. The intuition of TAPT is that datasets for specific tasks may only contain a subset of the text within the broader domain and continuing pretraining on the task dataset itself or other relevant data can be useful [15]. TAPT tends to adapt its linguistic representation by utilizing the unlabeled data in a *task-agnostic* way [42]. With the recent success of task-adaptive pre-training and self-training in natural language understanding (NLU), a research question arises: *Are task-adaptive pre-training (TAPT) and self-training (ST) complementary for natural language understanding (NLU)?*

In this chapter, we show that TAPT and ST can be complementary with simple TFS protocol by following **T**APT $\rightarrow$ **F**inetuning $\rightarrow$ **S**elf-training process (TFS). TFS protocol follows three steps: (1) TAPT on unlabeled corpus drawn from a task (2) Standard supervised finetuning

on labeled data inheriting parameters from TAPT as initialization to train a teacher model (3) Teacher model generates pseudo labels for *the same* unlabeled corpus in (1) and trains a student model in a self-training framework until convergence or reaching maximum rounds as shown in Figure 2.1. The first step utilizes unlabeled corpus in a task-agnostic way to learn general linguistic representations while the third one utilizes unlabeled corpus in a task-specific way during pseudo-labeling process. Therefore, unlabeled data are utilized *twice* through two different ways by taking advantages of TAPT and ST. TFS can effectively utilize unlabeled data to achieve strong combined gains of TAPT and ST consistently across six datasets covering sentiment classification, paraphrase identification, natural language inference, named entity recognition and dialogue slot classification. We further investigate various semi-supervised settings and consistently show that gains from TAPT and ST can be strongly *additive* by following TFS procedure.

2.2 Related Work

Pre-training. Unsupervised or self-supervised pre-training have achieved remarkable successes in natural language processing [2, 39, 3, 40, 22]. However, these models are pre-trained on a very large general domain corpus, e.g. Wikipedia, and may limit their performance on a specific task due to distribution shift [47, 20, 15]. To better handle aforementioned issue, domain-adaptive pre-training (DAPT) by continuing pre-training of existing language models, e.g. BERT and RoBERTa, on a large corpus of unlabeled domain-specific text data has been proposed and achieved great successes in specific domains [15, 47, 20]. [47] proposes BioBERT by continuing pre-training of BERT on biomedical domain corpus and outperforms BERT in biomedical text mining significantly. Following a similar idea, [20] proposes ToD-BERT by continuing pre-training of BERT on nine dialogue datasets for NLU tasks in task-oriented dialogue systems and achieves great successes in various few-shot NLU tasks in dialogue domain.

[15] takes one step further and continues pre-training of language models on a much smaller amount of unlabeled data but drawn from the same distribution for a given task (TAPT), which not only can achieve competitive results with DAPT but also is complementary with it.

Self-training. Self-training as one of the earliest and simplest semi-supervised learning has recently shown state-of-the-art performance for tasks like image classification [37, 48], object detection [43] and can perform at par with fully supervised models while using much less labeled training data. On natural language processing, [46] applies self-training for few-shot text classification and incorporates uncertainty estimation of the underlying neural network for unlabeled data selection. [11] improves self-training with meta-learning by adaptive sample reweighting to mitigate error propagation from noisy pseudo-labels for named entity recognition and slot tagging in task-oriented dialog systems. [38] injects noise to the input space as a noisy version of self-training for neural sequence generation and obtains state-of-the-art performance for tasks like neural machine translation. [45] utilizes information retrieval to retrieve task-specific in-domain data from a large amount of web sentences for self-training. Beyond these applications of self-training, [49] further theoretically proves that self-training and input-consistency regularization will achieve high accuracy in regard to ground-truth labels under certain assumptions.

There also exist works combing pre-training with self-training. [42] first conducts self-supervised pre-training with SimCLR [50] on ImageNet [51] in a task-agnostic way, then finetunes pre-trained models on limited labeled data and finally conducts self-training/knowledge distillation [52] via the same unlabeled examples as pre-training in a task-specific way. Such a framework enables models to make use of data twice in both pre-training and self-training/ knowledge distillation stage. [53] follows this framework on speech recognition and achieves state-of-the-art performance only with very limited labeled data. However, it's unclear that language models like BERT that have already been pre-trained in a very large general corpus can

benefit this framework or not since [42] and [53] conduct pre-training from scratch. In addition, they only conduct self-training in one round, making it unclear whether iterative self-training without pre-training can achieve comparable results in the end. A recent work [45] does both continuing pre-training and self-training in retrieved data from open domains but only observes gains for self-training while our work utilizes existing in-domain unlabeled data and finds that both TAPT and self-training are effective.

2.3 Algorithms

2.3.1 Problem setup

Denote $D_l = \{x_i, y_i\}^N$ to be a set of N labeled instances, where x_i is a sequence of m tokens: $x_i = \{x_{i1}, x_{i2}, ..., x_{im}\}$ with y_i being its label. Also, consider $D_u = \{x_j\}^M$ to be a set of M unlabeled instances drawn from the same distribution of $\{x_i\}^N$, where $M \gg N$. Assuming that we can only access a small amount of labeled data along with a much larger amount of unlabeled data, our goal is to fully leverage unlabeled data D_u to improve model performance.

2.3.2 Task-adaptive Pre-training (TAPT)

One simple yet effective way to improve BERT-like models with unlabeled data is task-adaptive pre-training (TAPT). The approach of TAPT is quite straightforward – simply continuing pre-training BERT-like models with masked language modeling (MLM) [2] on unlabeled text data for a specific given task [15].

Specifically, during MLM process, a proportion of randomly sampled tokens in the input are masked out with special token `[MASK]`. We conduct dynamical token masking during pre-training following [39, 20]. The training objective of MLM is the cross entropy loss to

reconstruct masked tokens:

$$\mathcal{L}_{\text{mlm}} = -\sum_{j=1}^{M}\sum_{k=1}^{m} \mathbb{1} * \log(\ p(x_{jk})), \tag{2.1}$$

where $\mathbb{1}$ is 1 if x_{jk} is masked out in the input, otherwise 0.

2.3.3 Self-training (ST)

Self-training begins with a *teacher* model p^t trained on the labeled data D_l. The teacher model is used to generate pseudo labels for unlabeled data D_u. The augmented data $D_l \cup D_u$ is then used to train a student model p^s. Specifically, $\forall x_j \in D_u$, we use teacher model to generate its soft label and then student model is trained with standard cross-entropy loss for labeled data and KL divergence for unlabeled data, which can be formulated as:

$$\begin{aligned}\mathcal{L}_{\text{st}} = &-\sum_{(x_i,y_i)\in D_l} \log(y_i|p^s(x_i)) \\ &-\sum_{x_j\in D_u} \text{KL}(p^t(x_j)||p^s(x_j)),\end{aligned} \tag{2.2}$$

where teacher model p^t is fixed in the current round. After training of student model with objective $\mathcal{L}_{\text{st}}$, it is assigned as a new teacher model in the next round and the teacher-student training procedure is repeated until convergence or reaching maximum rounds.

2.3.4 TAPT $\rightarrow$ Finetuning $\rightarrow$ Self-training (TFS)

Although TAPT has been proven effective to utilize unlabeled data, it's *task-agnostic* in the sense that it's unaware of specific tasks, e.g. sentence classification or name entity recognition. This paradigm learns general linguistic representations buried under unlabeled data, which are not directly tailored to a specific task. Utilizing data in a *task-agnostic* may lose the information

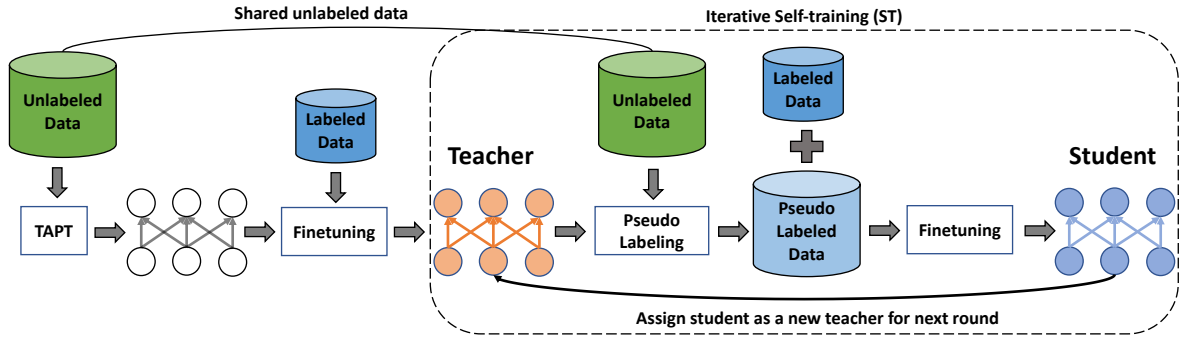

Figure 2.1: The overall pipeline of TFS. It has three steps (1) TAPT on unlabeled corpus drawn from a task (2) Train a teacher model on labeled data with TAPT as initialization (3) Teacher generates pseudo labels from *share* unlabeled corpus with (1) and trains a student model with both labeled and pseudo labeled data in an iterative self-training framework.

of unlabeled data key to the task at hand. On the contrary, self-training utilizes unlabeled data in a *task-specific* way. Pseudo labels are obtained through trained models and task-specific information can be encoded into pseudo labels. However, this method may only work well when a considerable portion of the predictions on unlabeled data are correct [38], otherwise early mistakes made by *teacher* model p^t due to limited labeled data can reinforce itself by generating incorrect labels for unlabeled data and re-training on this data can even result in a worse student model p^s in the next round [54].

TFS protocol by following **T**APT $\rightarrow$ **F**inetuning $\rightarrow$ **S**elf-training (TFS) process can take advantages of TAPT and ST but at the same time avoid their weakness. The overall pipeline of TFS is shown in Figure 2.1. TFS first utilizes unlabeled data in a *task-agnostic* way by

Algorithm 1 TFS Protocol

Input: Labeled corpus D_l, unlabeled corpus D_u and initialized model p_θ

1: Update model p_θ with TAPT on unlabeled corpus D_u by Equation 2.1
2: Train a teacher model p_τ initialized with p_θ by finetuning on labeled corpus D_l
3: **repeat**
4: Apply p_τ to the unlabeled corpus D_u to obtain $\hat{D}_u := \{(x_j, p_\tau(x_j)) | \forall x_j \in D_u\}$
5: Train a student model p_τ on $D_l \cup \hat{D}_u$ by Equation 2.2
6: Assign p_τ as a teacher for the next round
7: **until** Convergence or maximum rounds are reached

TAPT to have a better initialization for finetuning in next step and then finetunes a teacher model initializing its parameters from TAPT on labeled data in a standard supervised way. These two steps can build a better teacher model, avoid early mistakes and generate more accurate predictions for students, which is key to the success of self-training. The unlabeled data is leveraged again during self-training process in a *task-specific* way to further boost the performance of models at hand. We summarize the workflow of TFS in Algorithm 1.

2.4 Experiments

Here we conduct comprehensive experiments and analysis on different NLU datasets to demonstrate the effectiveness of TFS.

2.4.1 Experimental Setup

We use six popular large-scale datasets covering sentiment classification, paraphrase identification, natural language inference, named entity recognition and dialogue slot classification as follows.

(1) **SST-2** consists of sentences from movie reviews and human annotations of their sentiment [55] . The task is to predict the sentiment of a given sentence to be positive or negative [41].

(2) Both **QNLI** [41] and **MNLI** [56] are natural language inference datasets. QNLI is adapted from the SQuAD [57] question answering dataset and the task is to predict whether the context sentence includes the answer to a given question [41], which can be regarded as a binary classification problem. MNLI is slightly different from QNLI as it has multiple genres. Specifically, a premise sentence and a hypothesis sentence are given for each example in MNLI, and the task is to predict whether the given premise entails (*entailment*), contradicts (*contradiction*) the given hypothesis, or neither of them (*neutral*) [41].

(3) **QQP** is a paraphrase identification dataset [58]. The goal is to determine if two questions

Dataset	Task	Train size	Number of classes	Evaluation metrics
SST-2	Sentiment analysis	67,349	2	Accuracy
QNLI	Natural language inference	104,743	2	Accuracy
MNLI	Natural language inference	100,000*	3	Accuracy
QQP	Paraphrase identification	100,000*	2	F1
CoNLL 2003	Named entity recognition	14,041	9	F1
MultiWOZ 2.1	Slot classification	56,557	30	Micro-F1

Table 2.1: Dataset summary for evaluation. * are datasets that we randomly sample 100K instances from original training sets due to the high cost of iterative self-training.

asked on Quora are semantically equivalent [41], which can also be formulated as a binary classification problem.

(4) **CoNLL 2003** is a name entity recognition dataset and the task is to recognize four types of named entities: *persons*, *locations*, *organizations* and *miscellaneous* entities, where *miscellaneous* type does not fall into any of the previous three categories [59].

(5) **MultiWOZ 2.1** is a large-scale multi-domain dialogue dataset with human-human conversations [60]. We convert each dialogue into turns and the task is to predict whether a slot, e.g. restaurant name, is mentioned in a turn and can be cast as a multi-label binary slot classification problem [34].

SST-2, QNLI, MNLI and QQP datasets are from GLUE benchmark [1] and we only report their results on development sets as extensive experiments don't allow us to submit predictions on their test sets to the official leaderboard due to submission limitations [2]. Note that for both MNLI and QQP, we randomly downsample their training sets into 100K and development sets into 5K otherwise iterative self-training in various semi-supervised setups can be too costly and for MNLI, we report results on the matched development set. On both CoNLL 2003 and MultiWOZ 2.1, we report results of their test sets. For SST-2, MNLI and QNLI, we use standard accuracy metric and for QQP and CoNLL 2003 we report their F1 scores. For MultiWOZ 2.1, we report micro-F1. We summarize details of each dataset including task, full training data size,

[1] We only consider datasets with training data size larger than 10K in GLUE benchmark.

[2] See more about FAQ 1 at `https://gluebenchmark.com/faq`

number of classes and their evaluation metric in Table 2.1.

TAPT. We use `BERT-base` and `BERT-large` as our backbone to leverage both labeled and unlabeled data. Both labeled and unlabeled data are used for TAPT in our implementation so that we can use the same checkpoint for different data split and labeled data size without repeating costly pre-training process on the same dataset. During TAPT process, we use MLM objective with random token masking probability 0.15 for each training set listed in Table 2.1 following previous work [2, 39].

Finetuning. We follow standard supervised fine-tuning paradigm [2] by adding a linear projection layer with weight $W \in \mathbb{R}^{K \times I}$ on top of BERT in labeled data for each dataset listed in Table 2.1, where K is the number of classes and I is the dimensionality of representations of BERT. Specifically, for SST-2, QNLI, MNLI and QQP, we pass the representation of `[CLS]` token H^{CLS} to a linear layer followed by a `Softmax` function. Models are trained with cross-entropy loss between the predicted distributions $\texttt{Softmax}(W(H^{CLS}))$ and their ground truth labels. For CoNLL 2003 name entity recognition task, we feed the representation of each token into a linear layer followed by a `Softmax` function. Models are trained with average cross-entropy loss between the predicted distributions and their labels over all tokens [3]. For multi-label binary slot classification task on MultiWOZ 2.1, we pass the representation of `[CLS]` token H^{CLS} into a linear layer followed by a `Sigmoid` function. Models are trained with mean binary cross-entropy loss between the predicted distributions $\texttt{Sigmoid}(W(H^{CLS}))$ and their ground truth labels.

Self-training. We use the finetunned models with labeled data as *teachers* to generate pseudo soft labels on unlabeled data following [45]. Pseudo labeled data are combined with original labeled data to train student models by optimizing objective function in Equation 2.2. In the first

[3] We only calculate loss of the first token for words with multiple tokens after tokenization.

round, *students* utilize the same pre-trained checkpoints as their teachers and in the following rounds, *students* inherit parameters from *teachers*. We set maximum rounds as 3 since we observe that setting a much larger round brings the same results or very marginal gains on both SST-2 and CoNLL 2003.

2.4.2 Main results

Dataset	Model	FT	TAPT	ST	TFS
SST-2	$\texttt{BERT}_{\texttt{base}}$	$87.3_{1.5}$	$88.5_{0.7}$ (+1.2)	$88.4_{1.0}$ (+1.1)	$\mathbf{89.4}_{0.8}$ (+**2.1**)
	$\texttt{BERT}_{\texttt{large}}$	$89.0_{4.2}$	$90.7_{0.7}$ (+1.7)	$90.1_{4.2}$ (+1.1)	$\mathbf{91.4}_{0.4}$ (+**2.4**)
QNLI	$\texttt{BERT}_{\texttt{base}}$	$79.1_{0.8}$	$82.0_{0.5}$ (+2.9)	$80.2_{0.7}$ (+1.1)	$\mathbf{83.1}_{0.6}$ (+**4.0**)
	$\texttt{BERT}_{\texttt{large}}$	$82.6_{0.4}$	$83.2_{0.6}$ (+0.6)	$83.7_{0.4}$ (+1.1)	$\mathbf{84.4}_{0.6}$ (+**1.8**)
MNLI	$\texttt{BERT}_{\texttt{base}}$	$57.3_{1.9}$	$58.8_{1.3}$ (+1.5)	$59.2_{2.1}$ (+1.9)	$\mathbf{60.9}_{1.4}$ (+**3.6**)
	$\texttt{BERT}_{\texttt{large}}$	$66.4_{2.6}$	$67.6_{1.5}$ (+1.2)	$68.7_{2.2}$ (+2.3)	$\mathbf{69.4}_{1.4}$ (+**3.0**)
QQP	$\texttt{BERT}_{\texttt{base}}$	$71.3_{0.8}$	$74.3_{0.8}$ (+3.0)	$72.3_{0.6}$ (+1.0)	$\mathbf{75.1}_{0.9}$ (+**3.8**)
	$\texttt{BERT}_{\texttt{large}}$	$73.1_{1.7}$	$75.1_{0.9}$ (+2.0)	$74.2_{1.8}$ (+1.1)	$\mathbf{76.1}_{0.9}$ (+**3.0**)
CoNLL 2003	$\texttt{BERT}_{\texttt{base}}$	$78.8_{1.1}$	$79.3_{1.6}$ (+0.5)	$81.6_{1.1}$ (+2.8)	$\mathbf{82.2}_{1.3}$ (+**3.4**)
	$\texttt{BERT}_{\texttt{large}}$	$76.3_{2.4}$	$79.8_{1.0}$ (+3.5)	$79.4_{2.4}$ (+3.1)	$\mathbf{82.2}_{1.1}$ (+**5.9**)
MultiWOZ 2.1	$\texttt{BERT}_{\texttt{base}}$	$75.6_{0.7}$	$79.8_{0.5}$ (+4.2)	$76.6_{0.8}$ (+1.0)	$\mathbf{80.2}_{0.4}$ (+**4.6**)
	$\texttt{BERT}_{\texttt{large}}$	$77.7_{0.4}$	$81.4_{0.2}$ (+3.7)	$78.7_{0.6}$ (+1.0)	$\mathbf{81.8}_{0.3}$ (+**4.1**)

Table 2.2: Results comparison (%) of finetuned baselines on labeled data (FT), TAPT, ST and TFS of $\texttt{BERT}_{\texttt{base}}$ and $\texttt{BERT}_{\texttt{large}}$ on six different datasets with 1% labeled data. Mean results along with their standard deviation in the subscript are listed and values inside the parentheses are gains over FT.

In this section, we simulate data scarcity scenarios for these mentioned datasets in Table 2.1 for both $\texttt{BERT}_{\texttt{base}}$ and $\texttt{BERT}_{\texttt{large}}$. Specifically, for each dataset we randomly sample 1% training data as labeled corpus and left 99% as unlabeled data. Both labeled and unlabeled corpus is used as the input of TAPT while only unlabeled corpus is used for self-training. For all datasets, we randomly choose three data splits and have three different runs for each of them except $\texttt{BERT}_{\texttt{large}}$ on CoNLL 2003 and MultiWOZ 2.1 to combat their instability by leveraging their results on development sets. In these two datasets, we use ten different runs for each

data split on $\texttt{BERT}_{\texttt{large}}$ and report corresponding test set results based on top three runs on development sets. Results are summarized in Table 2.2.

Comparison between TAPT and ST. TAPT and ST in both $\texttt{BERT}_{\texttt{base}}$ and $\texttt{BERT}_{\texttt{large}}$ models consistently outperform finetuned baseline results across six different datasets, demonstrating their effectiveness as semi-supervised methods on NLU tasks. However, TAPT has inconsistent results compared to ST. TAPT outperforms ST in SST-2, QQP and MultiWOZ 2.1 datasets but underperforms ST or only achieves comparable results for QNLI, MNLI and CoNLL 2003. These results indicate that they learn different representations from unlabeled data since they utilize data from different perspectives.

TFS shows strong *additive* gains over individual TAPT and ST. On QNLI, TFS on $\texttt{BERT}_{\texttt{base}}$ improves 4% accuracy over finetuned baselines (FT), equal to the sum of gains from TAPT (+2.9%) and ST (+1.1%), and on $\texttt{BERT}_{\texttt{large}}$ improves 1.8% accuracy, even slightly larger than the sum of gains from TAPT (+0.6%) and ST (+1.1%). On MNLI, TFS on $\texttt{BERT}_{\texttt{base}}$ improves 3.6% accuracy, larger than the sum of gains from TAPT (+1.5%) and ST (+1.9%). Similar results on $\texttt{BERT}_{\texttt{base}}$ also hold for CoNLL 2003. For results of other settings, improvements of TFS can also be well approximated by simply adding gains from corresponding TAPT and ST over FT. These consistent and significant results show that TAPT and ST are complementary to each other and TFS can effectively add their gains.

2.4.3 Varying size of labeled data

We have demonstrated the effectiveness of TFS on both $\texttt{BERT}_{\texttt{base}}$ and $\texttt{BERT}_{\texttt{large}}$ in six different datasets with 1% training data in section 2.4.2. We further explore different sizes of labeled data on six datasets in Table 2.1 with $\texttt{BERT}_{\texttt{base}}$ model.

Specifically, on relatively simple dataset SST-2, we vary labeled data ratio as $\{0.1\%, 0.2\%,$

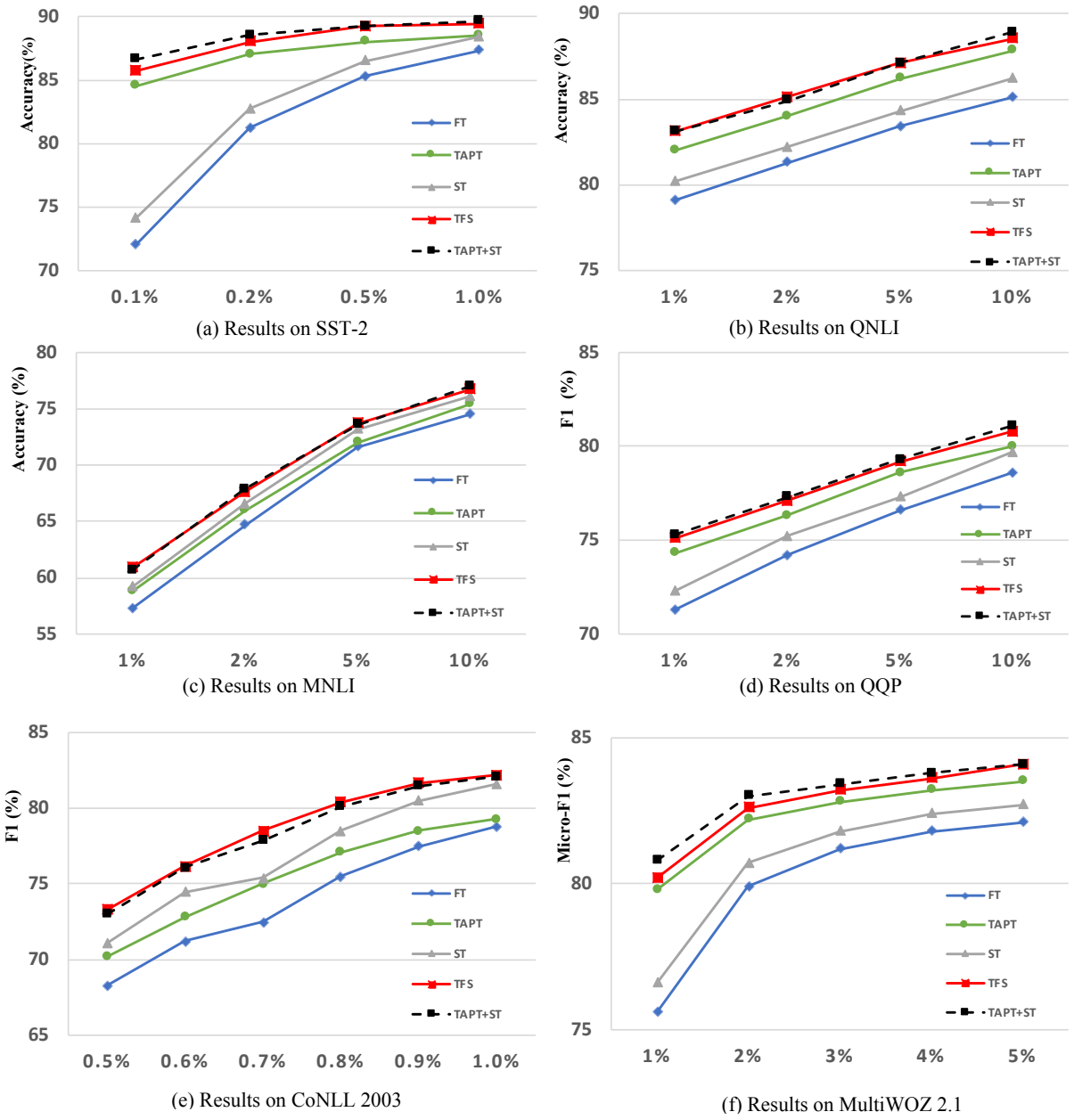

Figure 2.2: Results comparison (%) of FT, TAPT, ST and TFS with different sizes of labeled data on six datasets. `TAPT+ST` is ***not*** a method but references to demonstrate additive gains of TFS.

$0.5\%, 1.0\%\}$ and on more difficult QNLI, MNLI and QQP datasets, we vary their labeled ratios as $\{1\%, 2\%, 5\%, 10\%\}$. For CoNLL 2003, we explore labeled data ratio in $\{0.5\%, 0.6\%, 0.7\%, 0.8\%\, 0.9\%, 1.0\%\}$ and for MultiWOZ 2.1, we set labeled data ratio as $\{1\%, 2\%, 3\%, 4\%, 5\%\}$.

Following previous settings, for each labeled data ratio among these six datasets, we randomly select 3 data splits and each data split has three different runs. Final average results for each data ratio are reported over these nine runs. To better measure additive property of TFS, we introduce `TAPT+ST` in our results for references, which directly adds performance gains of TAPT and ST on FT.

Results of six different datasets among different sizes of labeled data are summarized in Figure 2.2. TAPT outperforms ST in SST-2, QNLI, QQP and MutiWOZ 2.1 datasets in various labeled data setups but underperforms it on MNLI and CoNLL 2003, indicating that TAPT and ST learn different representations from unlabeled data and have their pros and cons. However, TFS consistently and significantly outperforms TAPT and ST in all scenarios among six different datasets and again proves its effectiveness over TAPT and ST alone. For example, in CoNLL 2003 with 0.5% labeled data, TFS has relative 4.4% and 3.1% improvement over TAPT and ST, respectively. More importantly, TFS overall has very similar results with `TAPT+ST` in various labeled data size of different datasets, which further strengthens that TFS protocol can yield strong additive gains over TAPT and ST.

2.4.4 Analysis

Given the promising results in the previous experiments, we aim to answer why TFS outperforms ST consistently and significantly. Indeed, the differences between TFS and ST lie in two aspects: (1) TFS uses initialization from the checkpoint of TAPT rather than original BERT as ST. (2) TFS utilizes pseudo labels generated from `TAPT finetuned models` while ST uses pseudo labels generated from `BERT finetuned models` (FT). To further investigate these two perspectives, we design a variant of original ST, **ST** with **T**APT **I**nitilization (STTI), which utilizes pseudo labels generated by BERT finetuned models as ST but is initialized with the same checkpoints from TAPT as TFS during the first round of self-training. The intermediate

	FT	TAPT	ST	STTI	TFS
Init.	BERT	TAPT*	BERT	TAPT*	TAPT*
Pseud.	-	-	FT	FT	TAPT
Acc. (0.1%)	72.0	84.5	74.1	75.4	**85.7**
Acc. (1.0%)	87.3	88.5	88.4	88.8	**89.4**

Table 2.3: Results comparison of FT, TAPT, ST, STTI and TFS on SST-2 dataset. Rows with *Init.* and *Pseud.* show initialization and pseudo labeler of different models, respectively. Last two rows list accuracy with 0.1% and 1.0% labeled training data of these models. * represents models without finetuning on labeled data.

variant can help us better understand what makes TFS work. We run experiments on SST-2 with 0.1% and 1.0% labeled data for $\texttt{BERT}_{\texttt{base}}$ to compare ST, STTI and TFS. The results of STTI are obtained by running over the same three data splits as ST and TFS, and having three different runs for each data split. Results are averaged and summarized in Table 2.3.

Importance of initialization. Table 2.3 shows that STTI consistently outperforms ST in both 0.1% and 1% labeled setup. Comparing its difference with ST, we can conclude that its improvement over ST comes from its TAPT initialization. Results of MNLI and CoNLL 2003 in Figure 2.2 (c) and (e) also validate the importance of initialization. In these two datasets, although ST can consistently generate more accurate labels than TAPT finetuned models, meaning that it can match TAPT finetuned performance during self-training process, it still underperforms TFS in the end. These results again indicate the importance of initialization. Without TAPT as initialization, even if ST itself can outperform TAPT finetuned models, who are teachers of TFS in self-training process, but still at its end will be left behind of TFS.

Importance of pseudo label correctness. Table 2.3 also shows that STTI underperforms TFS in both 0.1% and 1% labeled setup although both of them inherit the same parameters from TAPT. These results indicate that beyond initialization, accurate pseudo labels also matter for self-training process. STTI takes pseudo labels generated from BERT finetuned baselines (FT)

that have more errors while TFS utilizes more accurate pseudo labels generated from TAPT finetuned models. Suffering from more incorrect pseudo labels in the beginning of self-training process, STTI may converge to a worse local optima than that of TFS. This is even more severe when labeled data is 0.1% and FT is left far behind of TAPT finetuned models, causing that STTI has 10.3% accuracy gap compared to TFS. These results prove the importance of accurate pseudo labels for self-training.

Combining these findings, we argue that TFS can outperform ST at least for two reasons: (1) it has a better initialization from TAPT compared to ST from BERT (2) it utilizes more accurate pseudo labels from TAPT finetuned models than ST.

2.5 Conclusion

In this chapter, we demonstrate that TAPT and ST are complementary for NLU tasks with TFS by following TAPT $\rightarrow$ Finetuning $\rightarrow$ Self-training process. Our extensive experiments in various semi-supervised setups across six popular datasets show that they are not only complementary but also strongly *additive* with TFS protocol. We further show that TFS outperforms ST through (1) a better initialization from TAPT (2) more accurate predictions from TAPT finetuned models. We hope that TFS could serve as an important semi-supervised baseline for future NLP studies.

Chapter 3

Teaching PLMs with Commonsense Reasoning via Data Synthesis from KB

3.1 Introduction

Recently, pretrained language models [2, 9, 10] have achieved great successes on various natural language understanding tasks, and they are also believed to master a certain level of commonsense reasoning abilities [61, 62, 63]. Equipping machines with commonsense reasoning ability has been seen as one of the key milestones of artificial general intelligence [17]. However, the commonsense reasoning ability of these state-of-the-art pretrained models is still far away from that of humans [18, 8]. One probable reason is that these models are learned from massive amounts of *unstructured* texts with various language model (LM) objectives (e.g., masked language model [2]). That is, the commonsense reasoning capability is never explicitly taught to the pretrained models, but is implicitly acquired through modeling input texts via LM objectives. In this chapter, we focus on how to *explicitly* teach the pretrained models the commonsense reasoning ability.

There are several challenges in explicitly injecting commonsense reasoning capability

into pretrained models. First, it is generally hard to exploit direct supervision signals for commonsense reasoning from unstructured texts, and it is also expensive, if ever possible, to create a large amount of human-labeled data for learning the commonsense reasoning ability. Second, the pretrained models do not have explicit symbolic reasoning operations; instead, the reasoning is performed implicitly through the neural network operations such as self-attention, and any knowledge relevant to reasoning is stored in the network weights. Note that the weights are only learned to fit certain input-output pairs, where the inputs to the model are natural language sentences, and the outputs are certain items to predict (e.g., masked tokens, next sentence indicator). That is, any reasoning ability has to be acquired implicitly by processing unstructured input texts during pretraining, and it is more difficult to directly supervise the reasoning path for a pretrained model.

To address these challenges, we propose a simple yet effective method to teach pretrained models with explicit commonsense reasoning abilities. The key idea is to exploit the *structured* knowledge in commonsense knowledge bases (e.g., ConceptNet [19]) to generate multiple-choice questions that require commonsense reasoning. Specifically, we sample subgraphs from KB to generate various logical forms and then use text templates to generate natural language questions and candidate answers. As a result, we automatically generate a large-scale multiple-choice question answering dataset with 167 million questions that ask about specific logical relations between different entities/concepts. These questions will be used as the additional training data to further *refine* the pretrained models, which force them to learn the commonsense reasoning ability in order to answer correctly. These training inputs are already in the natural language form, which is consistent with the input of pretrained models. Therefore, it allows the model to *continually* adjust its pretrained weights so that it can master more commonsense reasoning abilities; it naturally combines the power of pretrained weights from unstructured texts and the new information from structured knowledge in KB. Our experimental results show that the proposed approach consistently outperforms the baselines on commonsense reasoning

tasks, especially in few-shot learning settings. In addition, we examine which logical relations are more "commonsense" and find that only a few simple ones are most relevant. This work is as a preliminary attempt to integrate structured commonsense knowledge into pretrained models with promising results. As we shall see, the *structured* knowledge in KB allows us to *systematically* construct the logical relations that we want to teach the models. We hope that our work could inspire more research towards combining structured knowledge and pretrained models.

3.2 Related Work

Structured knowledge, e.g. Freebase [64] and ConceptNet [19], has been explored for improving question answering and reading comprehension systems. [65] transforms questions into knowledge graphs so that large-scale knowledge bases can be leveraged to provide the missing background to improve question answering. [66] exploits to incorporate knowledge triplets from knowledge bases into answer generation process. [67] proposes a general framework to leverage graph convolutional neural networks [68] to encode external knowledge from knowledge base for question answering systems. [69] utilizes relations and embeddings from ConceptNet to build feature-based classifier for reading comprehension problems. [70] augments the input with relation embedding from ConceptNet to better incorporate commonsense knowledge for commonsense machine comprehension problems. [71] utilizes ConceptNet to pre-train direct and indirect relational functions between concepts so that they could be added to existing neural network models. This line of work rely on knowledge retrieval from knowledge base for specific questions and, are not generic and flexible.

Another line of work directly inject commonsense knowledge from knowledge base into the parameters of pre-trained models so that they can be used as drop-in replacements for existing ones. [72] fine-tunes pre-trained BERT model on questions constructed by predicting the head

or tail mention in a triple while [73] proposes to align triples in knowledge base to Wikipedia sentences to form natural language questions to continue pre-training BERT. Instead, we explore utilizing knowledge base to systematically construct different logical forms and then convert them into question-answer pairs to explicitly teach multiple types of pre-train language models with commonsense reasoning.

3.3 The Proposed Approach

The key idea of our method is to generate multiple-choice questions from different subgraphs in KB, and then we use the generated data to further *refine* the pretrained models. The overall idea of the data generation process is shown in Figure 3.1, which consists of (i) generating different logical forms from a sampled subgraph in KB, (ii) generating multiple-choice questions in natural language form.

3.3.1 Generating multiple-choice questions as the refinement data

Generating logical forms We first sample a subgraph from KB that is in the following form:

$$(A \xrightarrow{R_1} B \xrightarrow{R_2} C) \tag{3.1}$$

where A, B, and C are three different entities in the KB, and R_1 and R_2 represent two different relations in the KB. For each of the above subgraph, we will construct a multiple-choice question related to the entity B in the following manner. First, introduce the following two sets: $\mathcal{R}_1 = \{X \in \Omega : \ A \xrightarrow{R_1} X\}$, $\mathcal{R}_2 = \{X \in \Omega : \ X \xrightarrow{R_2} C\}$, where Ω denotes the entire entity set. Note that the set $\mathcal{R}_1$ represents the set of all (tail) entities that have relation R_1 with A, and $\mathcal{R}_2$ represents the set of all (head) entities that have relation R_2 with entity C. We use the two circles in the Venn diagram (Fig. 3.1) to represent these two sets, respectively.

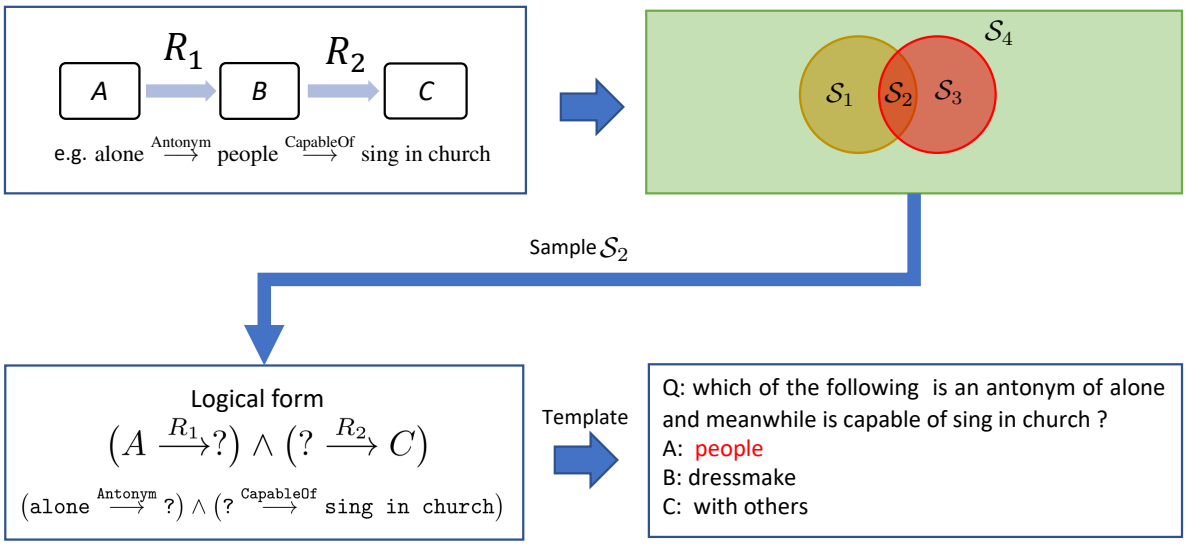

Figure 3.1: The generation of logical forms and multiple-choice questions in our proposed approach. The yellow and the red circles in the Venn diagram represent the sets $\mathcal{R}_1$ and $\mathcal{R}_2$, respectively.

Note from Fig. 3.1 that the entire space could be partitioned into four subsets, denoted as: $\mathcal{S}_1 = \mathcal{R}_1 \cap \mathcal{R}_2^c$, $\mathcal{S}_2 = \mathcal{R}_1 \cap \mathcal{R}_2$, $\mathcal{S}_3 = \mathcal{R}_1^c \cap \mathcal{R}_2$, $\mathcal{S}_4 = \mathcal{R}_1^c \cap \mathcal{R}_2^c$. Each subset represents a certain logical relation. For example, the subset $\mathcal{S}_2 = \mathcal{R}_1 \cap \mathcal{R}_2$ means all the entities that have relation R_1 with A *and* have relation R_2 with C. Using these four subsets, we could compose questions that ask about all different logical relations from the subgraph in (3.1). To see this, note that we could compose a set by either choosing or not choosing each subset $\mathcal{S}_i$, which leads to a total of $2^4 = 16$ subsets. Among them, two trivial cases are excluded: the all-chosen case (full set) and the all-not-chosen set (empty set). Therefore, there are a total of 14 different logical relations about (3.1) that we could ask (see Appendix A.2 for all the 14 logical forms). To have a more concrete example, consider the composed subset $\mathcal{S}_1 \cup \mathcal{S}_3$, then we are examining the logical relation:

$$((A \xrightarrow{R_1} ?) \vee (? \xrightarrow{R_2} C)) \wedge \neg((A \xrightarrow{R_1} ?) \wedge (? \xrightarrow{R_2} C)) \tag{3.2}$$

where $\wedge$ and $\vee$ denotes logical AND and logical OR, respectively, and $\neg$ denotes logical negation (NOT). This approach allows us to *systematically* generate all different types of logical relations pertaining to each sampled subgraph from the KB, which even covers questions about a single relation. For example, the logical form corresponding to $\mathcal{S}_1 \cup \mathcal{S}_2$ is "$A \xrightarrow{R_1} ?$", and the logical form corresponding to $\mathcal{S}_2 \cup \mathcal{S}_3$ is given by "$? \xrightarrow{R_2} C$", which ask the tail entity and head entity, respectively.

Generating multiple-choice questions Now that once we have a logical form in the form of (3.2), we can generate natural language questions that ask about this particular logical relation. We achieve this by using text templates. Specifically, we first create two different types of mapping, namely, *affirmative mapping* and *negative mapping*. The affirmative mapping is used to generate sentences with affirmative questions, while the negative mapping is used for generating negative ones. Consider the following specific example of a logical form (also shown in Figure 3.1):

$$(\texttt{alone} \xrightarrow{\texttt{Antonym}} ?) \wedge (? \xrightarrow{\texttt{CapableOf}} \texttt{sing in church})$$

where the correct answer for the missing entity is people. In the above logical form, the relation CapableOf will be mapped into "is capable of" using affirmative mapping. On the other hand, when there is a negation $\neg$ before the relation CapableOf, it will be mapped into "is not capable of" using a negative mapping. These obtained strings from relations will be put together with the head entities and the tail entities to generate sentences as natural as possible by using a set of simple heuristic rules. For example, the above logical relation will be mapped into the following natural language sentence: "*which of the following is an antonym of alone and meanwhile is capable of sing in church?*" In Appendix A.2, we give examples of the possibly generated questions for all the 14 logical relations.

Generating candidate answers The correct answer is obtained from the particular logical form that we want to examine. For example, if we want to generate a question regarding the logical form (3.2), the set of correct answer is given by $\mathcal{S}_1 \cup \mathcal{S}_3$. On the other hand, for the wrong candidate answers, we firstly choose wrong subset uniformly from $\mathcal{S}_1, \ldots, \mathcal{S}_4$ and then sample an entity from the selected subset.

3.3.2 Refinement: teaching the pretrained models with commonsense reasoning

To teach the pretrained models with commonsense reasoning, we further train the pretrained models on the generated multiple-choice questions to predict the correct answer, which becomes a multi-class classification problem. Afterwards, the model is finetuned on different downstream tasks. We name this step as *refinement* to distinguish it from the *pretraining* and the *finetuning* stages.

3.4 Experiments

In this section, we demonstrate the effectiveness of the proposed method and perform analysis on which logical relations are more "commonsense". First, we briefly describe the experimental setting, and more details could be found in Appendix A.1. We first preprocess ConceptNet and keep 3,098,816 English-only triples. Then, we perform search on these triples and obtain a total of 167,395,947 subgraphs that are in the form of (3.1). These subgraphs would lead to over 167 million multiple-choice questions for further refining the pretrained models. To evaluate the performance, we finetune the refined models on two commonsense reasoning tasks: *CommonsenseQA* [8] and *CosmosQA* [74] (see Appendix A.1 for the descriptions).

Dataset	CommonsenseQA-dev	CosmosQA-dev
BERT	57.33(1.03)	58.34(0.82)
BERT + refine	**59.28(0.43)**	**58.91**(0.44)

Table 3.1: Performance comparison in accuracy (%) on CommonsenseQA and CosmosQA deveopment set with full training data size. "model + refine" denotes our method. All results are averaged over five independent runs, with standard deviations listed inside the parentheses.

3.4.1 Main results

We first evaluate our method on pretrained BERT base model to demonstrate its effectiveness. We refine BERT on our generated multiple-choice questions and then finetune them on both CommonsenseQA and CosmosQA with full training data. We compare the results of our approach to the original BERT without the refinement process. We report mean results and standard deviations with 5 different runs. Results are summarized in Table 3.1. Our approach can consistently improve strong BERT model on both CommonsenseQA and CosmosQA datasets. On CommonsenseQA, our method can improve strong BERT baseline with 1.95% accuracy and on CosmosQA, our method improves 0.57% accuracy. These results demonstrate the effectiveness of our approach in teaching pretrained language model with commonsense reasoning.

We further evaluate our proposed method on pretrained BERT base, GPT and XLNet base under different sizes of fine-tuning data on CommonsenseQA. Specifically, we vary training data size as {100, 200, 400, 800, 1600, 3200, 9741 (full training data size)} . For each data size,

Data size	100 (1.0%)	200 (2.1%)	400 (4.1%)	800 (8.2%)	1600 (16.4%)	3200 (32.9%)	9741 (100%)
BERT [2]	34.94(1.97)	38.41(2.16)	41.73(2.02)	45.44(1.16)	47.53(1.15)	51.84(0.62)	57.33(1.03)
BERT + refine	42.54(1.27)	**44.93**(1.69)	**47.03**(0.27)	**50.58**(0.64)	**53.43**(0.85)	**54.86**(0.75)	59.28 (0.43)
GPT [9]	27.90(1.30)	28.34(1.39)	30.20(1.99)	33.96(2.53)	38.54(1.55)	45.45(0.65)	50.75(1.08)
GPT + refine	37.69(0.27)	38.56(0.37)	40.49(0.50)	42.49(0.44)	44.24(0.58)	46.50(0.35)	51.52(0.62)
XLNet [10]	25.04(0.67)	27.44(1.08)	29.80(2.17)	34.27(0.89)	38.67(1.30)	47.14(1.25)	57.25(1.14)
XLNet + refine	**43.60**(0.15)	43.67(0.24)	45.81(0.23)	47.24(0.47)	50.60(0.53)	53.41(0.32)	**59.31**(0.44)

Table 3.2: Results with different size of fine-tuning data in accuracy (%) on the CommonsenseQA development set. Data size percentages are listed in the parentheses.

we report mean results with 5 different runs along with their standard deviations. Results are summarized in Table 3.2. Our method has consistently and significantly better performance when fine-tuning data sizes are limited across three different models. When training data is extremely low with 100 instances, our approach can improve BERT, GPT and XLNet with 7.60%, 9.79%, 18.56% absolute accuracy, respectively, meaning that the refinement process effectively teaches pretrained models commonsense reasoning even with a few finetuning samples. With full finetuning data, our method also performs consistently better than baselines and improves up to 2.06% absolute accuracy for XLNet. These consistent and significant results across different fine-tuning data sizes and model types demonstrate the effectiveness of our method to teach pre-trained language models with commonsense reasoning.

3.4.2 Analysis

Our method can generate question-answer pairs requiring different logical reasoning capabilities. Hence, a natural question is which logical relations are more "commonsense"? To partially answer this question, we refine the pretrained BERT base model on different subsets of logical relations from all the 14 logical forms in Appendix A.2. For BERT + refine (all), we sample all valid logical forms according to a uniform distribution. For BERT + refine (1,2,5), BERT + refine (2,4,5), and BERT + refine (4,7,9), logical forms are uniformly sampled over (#1, #2, #5), (#2, #4, #5), and (#4 , #7, #9), respectively. We conduct refining experiments

Method	BERT	BERT + refine (1,2,5)	BERT + refine (2,4,5)	BERT + refine (4,7,9)	BERT + refine (all)
CommonsenseQA	57.33(1.03)	59.10(0.43)	58.18(0.41)	56.42(0.82)	59.28(0.43)
CosmosQA	58.34(0.82)	59.28(0.81)	59.76(0.75)	58.86(0.74)	58.91(0.44)
Average	57.84	59.19	58.97	57.64	59.10

Table 3.3: The relevance of different logical relations to commonsense. "BERT + refine (1,2,5)" means the pretrained BERT model is refined on the logical forms #1, #2, and #5 defined in Appendix A.2. "Average" denotes the averaged results on CommonsenseQA and CosmosQA under each setting.

on CommonsenseQA and CosmosQA datasets, train models five times for each setting and report their mean values and standard deviations. Results are shown in Table 3.3. We observe that relatively simple logical relations (#1, #2, #5) (i.e., simple logical `AND` and single relation reasoning) are more relevant to commonsense; refining on just three of them achieves similar performance to refining all (BERT + refine (all)). On the other hand, the logical forms (#4, #7, #9), which require more logical compositions and negations, are less commonsense; refining on them does not improve the baseline BERT model. This is consistent with our intuition that commonsense should be something relatively straightforward.

3.5 Conclusion

In this chapter, we propose a simple yet effective method to use structured knowledge (i.e., ConceptNet) to enhance the commonsense reasoning abilities of pretrained language models. The structured knowledge in KB allows us to construct various logical forms, and then generate multiple-choice questions that require commonsense logical reasoning. Experimental results demonstrate that, when refined on these training examples, these models consistently improve their performance, especially in few-shot learning settings. Further analysis shows that simple logical relations are more relevant to commonsense. Exploring methods than can generate more diverse natural language questions instead of relying on patterns is left as future work.

Chapter 4

Evaluating and Augmenting Dialogue State Tracking via Data Synthesis from PLM

4.1 Introduction

Task-oriented dialogue (TOD) systems have recently attracted growing attention and achieved substantial progress [75, 76, 77, 78, 79], partly made possible by the construction of large-scale datasets [80, 81, 82]. Dialogue state tracking (DST) is a backbone of TOD systems, where it is responsible for extracting the user's goal represented as a set of slot-value pairs (e.g., (*area*, *center*), (*food*, *British*)), as illustrated in the upper part of Figure 4.1. The DST module's output is treated as the summary of the user's goal so far in the dialogue and directly consumed by the subsequent dialogue policy component to determine the system's next action and response. Hence, the accuracy of the DST module is critical to prevent downstream error propagation [83], affecting the end-to-end performance of the whole system.

With the advent of representation learning in NLP [84, 2, 3], the accuracy of DST models has increased from 15.8% (in 2018) to 55.7% (in 2020). While measuring the held-out accuracy is often useful, practitioners consistently overestimate their model's generalization [85, 86]

data	attraction-name	hotel-name	restaurant-name	taxi-departure	taxi-destination	train-departure	train-destination
dev	94.5	96.4	97.3	98.6	98.2	99.6	99.6
test	96.2	98.4	96.8	95.6	99.5	99.4	99.4

Table 4.1: The percentage (%) of domain-slot values in dev/test sets covered by training data.

slot name	data	area	book day	book time	food	name	price range
book people	train	1.9	38.8	39.2	2.1	16.4	1.5
	dev	1.9	38.9	38.9	1.9	16.3	2.2
	test	2.7	36.9	37.7	1.6	18.7	2.4

Table 4.2: Co-occurrence distribution(%) of *book people* slot with other slots in *restaurant* domain within the same user utterance. It rarely co-occurs with particulars slots (e.g., *food*), which hinders the evaluation of DST models on realistic user utterances such as "*I want to book a Chinese restaurant for 8 people.*"

since test data is usually collected in the same way as training data. In line with this hypothesis, Table 4.1 demonstrates that there is a substantial overlap of the slot values between training and evaluation sets of the MultiWOZ DST benchmark [80]. In Table 4.2, we observe that the slot co-occurrence distributions for evaluation sets tightly align with that of train split, hinting towards the potential limitation of the held-out accuracy in reflecting the actual generalization capability of DST models. On the other hand, state-of-the-art DST model is still far from perfect even on held-out test set (55.7% in 2020) when comparing it with models for other natural language understanding tasks, e.g. sentiment analysis, which often exceeds 90% accuracy [39]. Inspired by this phenomenon, we aim to address and provide further insights into the following question: *how can we evaluate and augment state-of-the-art DST models with dialogues that have novel and realistic scenarios but are not captured well by a given dataset?*

To address this question, we propose *controllable counterfactuals* (CoCo) as a principled, model-agnostic approach to generate novel conversation scenarios beyond a given dataset. Our approach is inspired by the combination of two natural questions: how would DST systems react to (1) unseen slot values and (2) rare but realistic slot combinations? CoCo first encapsulates these two aspects under a unified concept called *counterfactual goal* obtained by a stochastic

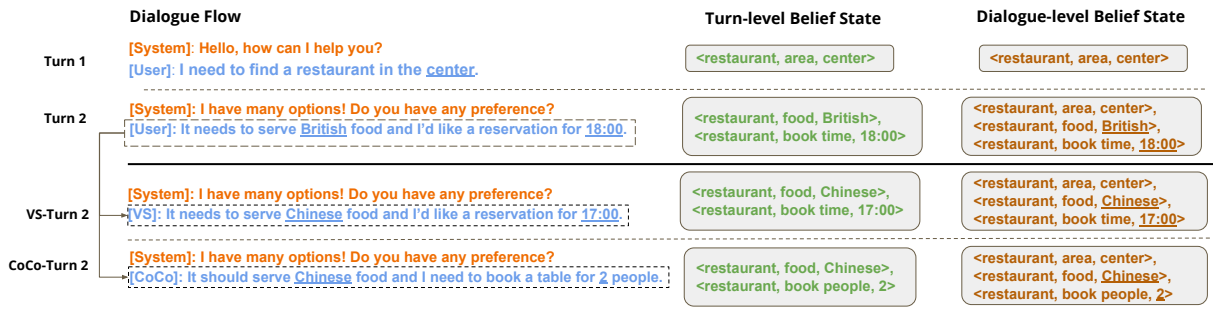

Figure 4.1: The upper left is a dialogue example between *user* and *system* with its turn-level and dialogue-level belief states on the upper right. The lower left are valid user utterance variations generated by VS and CoCo with their corresponding belief states derived from the original ones on the right.

policy of dropping and adding slots to the original turn-level belief state followed by replacing slot values. In the second step, CoCo conditions on the dialogue history and the counterfactual goal to generate *counterfactual conversation*. We cast the actual utterance generation as a conditional language modeling objective. This formulation allows us to plug-in a pretrained encoder-decoder architecture [40] as the backbone that powers the counterfactual conversation generation. We also propose a strategy to filter utterances that fail to reflect the counterfactual goal exactly. We consider *value substitution* (VS), as presented in Figure 4.1, as a special CoCo case that only replaces the slot values in the original utterance without adding or dropping slots. When we use VS as a fall-back strategy for CoCo (i.e., apply VS when CoCo fails to generate valid user responses after filtering), we call it CoCo+.

We first evaluate three strong DST models [87, 88, 89] with our proposed controllable counterfactuals generated by CoCo and CoCo+, and our results show that the performance of each significantly drops (up to 30.8%) compared to their joint goal accuracy on the original MultiWOZ held-out evaluation set. On the other hand, we find that these models are, in fact, quite robust to paraphrasing with back-translation, where their performance only drops up to 2%. We further utilize CoCo+ as a data augmentation method and our results show that it consistently improves the robustness of the investigated DST models on counterfactual

conversations generated by each of VS, CoCo and CoCo+, and also improves the joint goal accuracy of these strong DST models on the original MultiWOZ evaluation set. More notably, when CoCo+ multi-round data augmentation (by sampling multiple counterfactual goals for each turn) is used on the best of three DST models, its performance on the original MultiWOZ evaluation set improves up to 5.49% joint goal accuracy. In addition, human evaluations show that CoCo-generated counterfactual conversations perfectly reflect the underlying user goal with more than 95% accuracy and are found to be quite close to original conversations in terms of their human-like scoring. These results prove our proposed approach's reliability and potential to be used for evaluating and augmenting DST models.

4.2 Related Work

Dialogue State Tracking. DST has been a core component in current state-of-the-art TOD systems. Traditional approaches usually rely on hand-crafted features or domain-specific lexicon [90, 91] and require a predefined ontology, making them hard to extend to unseen values. To tackle this issue, various methods have been proposed. [92] treats DST as a reading comprehension problem and predicts slot values with start and end positions in the dialogue context. [93] proposes DS-DST, a dual-strategy model that predicts values in domains with a few possible candidates from classifiers and others from span extractors. Furthermore, [88] proposes TripPy, a triple copy strategy model, which allows it to copy values from the context, previous turns' predictions and system informs.

An alternative to classification and span prediction is value generation. [87] generates slot values with a pointer generator network [94] without relying on fixed vocabularies and spans. [89] models DST as a conditional generation problem and directly finetunes GPT2 [3] on DST task and achieves state-of-the-art on the MultiWOZ.

Adversarial Example Generation and Data Augmentation in NLP. Adversarial example

generation has been commonly studied in computer vision [95, 96] and has received growing attention recently in NLP domain as well. [97] finds adversarial examples in the embedding space, and then remapped them to the discrete space. [98] proposes a population-based word replacing method and aims to generate fluent adversarial sentences. These methods often edit the original data greedily assuming access to the model's gradients or outputs besides querying the underlying model many times [99]. Alternative line of work investigates generating adversarial examples in a model-agnostic way. [100] proposes to generate adversarial paraphrases of original data with different syntactic structures. [101] automatically generates sentences with key word overlappings of questions in SQuAD [57] to distract computer systems without changing the correct answer or misleading humans.

There also exist many data augmentation studies in NLP. [102] proposes to replace words with their nearest neighbors in embedding space to create new training instances. [103] proposes EDA (easy data augmentation) by synonym replacement, random insertion, swap and deletion of given sentences in training set to improve performance on text classification tasks. [104] instead conducts data augmentation via mixup [105] in embedding space for sentence classification tasks. [106, 107] further utilize pre-train language models to conduct data augmentation for text classification tasks.

Although different methods have been proposed to evaluate or augment NLP models, majority of the prior work in this line focus either on text classification, neural machine translation or reading comprehension problems. Perhaps the most similar existing work with ours are [108] and [109]. [108] focuses on evaluating and improving intent classification and slot tagging in TOD while [109] targets at synthetic competitive negotiation dialogues [110] without DST component. In this work, however, we focus on evaluating and augmenting a core component of state-of-the-art TOD, DST, on the widely used benchmark, MultiWOZ.

4.3 Background

Multi-domain DST task definition. Let $X_t = \{(U_1^{\text{sys}}, U_1^{\text{usr}}), ..., (U_t^{\text{sys}}, U_t^{\text{usr}})\}$ denote a sequence of turns of a dialogue until the t-th turn, where U_i^{sys} and U_i^{usr} $(1 \leq i \leq t)$ denote system and user utterance at the i-th turn, respectively. In multi-domain DST, each turn $(U_i^{\text{sys}}, U_i^{\text{usr}})$ talks about a specific domain (e.g., *hotel*), and a certain number of slots (e.g., *price range*) in that domain. We denote all N possible domain-slot pairs as $S = \{S_1, ...S_N\}$. The task is to track the value for each S_j $(1 \leq j \leq N)$ over X_t (e.g., *hotel-price range*, *cheap*). Belief states can be considered at two granularities: turn-level (L_t) and dialog-level (B_t). L_t tracks the information introduced in the last turn while B_t tracks the accumulated state from the first turn to the last. As illustrated in the upper part of Figure 4.1, when the dialogue flow arrives at the second turn, B_2 becomes {(*restaurant-area*, *center*), (*restaurant-food*, *British*), (*restaurant-book time*, *18:00*)}, while L_2 is {(*restaurant-food*, *British*), (*restaurant-book time*, *18:00*)}, essentially tracking the update to B_t by the last turn.

Problem definition. Given a tuple $< X_t, L_t, B_t >$, our goal is to generate a new user utterance $\hat{U}_t^{\text{usr}}$ to form a novel conversation scenario $\hat{X}_t = \{(U_1^{\text{sys}}, U_1^{\text{usr}}), ..., (U_t^{\text{sys}}, \hat{U}_t^{\text{usr}})\}$ by replacing the original user utterance U_t^{usr} with $\hat{U}_t^{\text{usr}}$. To preserve the coherence of dialogue flow, we cast the problem as generating an alternative user utterance $\hat{U}_t^{\text{usr}}$ conditioned on a modified $\hat{L}_t$ derived from original turn-level belief state L_t in a way that is consistent with the global belief state B_t. This formulation naturally allows for producing a new tuple $< \hat{X}_t, \hat{L}_t, \hat{B}_t >$ controllable by $\hat{L}_t$, where $\hat{B}_t$ is induced by B_t based on the difference between L_t and $\hat{L}_t$. As illustrated in the lower part of Figure 4.1, U_2^{usr} is replaced with the two alternative utterances that are natural and coherent with the dialogue history. Resulting set of $< \hat{X}_t, \hat{L}_t, \hat{B}_t >$ will be used to evaluate and augment DST models when $< X_t, L_t, B_t >$ is from held-out test set and training set, respectively.

Paraphrase baseline with back-translation. Paraphrasing the original utterance U_t^{usr} is a natural way to generate $\hat{U}_t^{\text{usr}}$. With the availability of advanced neural machine translation (NMT) models, round-trip translation between two languages (i.e., back-translation (BT)) has become a widely used method to obtain paraphrases for downstream applications [111]. We use publicly available pretrained *English→German* ($\log(g|e)$) and *German→English* ($\log(e|g)$) NMT models.[1] We translate U_t^{usr} from English to German with a beam size K, and then translate each of the K hypotheses back to English with the beam size K. Consequently, we generate K^2 paraphrase candidates of $\hat{U}_t^{\text{usr}}$ and then rank them according to their round-trip confidence score $\log(g|e) + \log(e|g)$. As paraphrases are expected to preserve the meaning of U_t^{usr}, we set $\hat{L}_t = L_t$ and $\hat{B}_t = B_t$.

4.4 CoCo

As illustrated in Figure 4.2, CoCo consists of three main pillars. We first train a conditional user utterance generation model $p_\theta(U_t^{\text{usr}}|U_t^{\text{sys}}, L_t)$ using original dialogues. Secondly, we modify L_t into a possibly arbitrary $\hat{L}_t$ by our counterfactual goal generator. Given $\hat{L}_t$ and U_t^{sys}, we sample $\hat{U}_t^{\text{usr}} \sim p_\theta(\hat{U}_t^{\text{usr}}|U_t^{\text{sys}}, \hat{L}_t)$ with beam search followed by two orthogonal filtering mechanisms to further eliminate user utterances that fail to reflect the counterfactual goal $\hat{L}_t$.

4.4.1 Value Substitution

A robust DST model should correctly reflect value changes in user utterances when tracking user's goal. However, slot-value combinations, e.g. (*restaurant-book time*, *18:00*), in evaluation sets are limited and even have significant overlaps with training data as shown in Table 4.1, which can hinder evaluations on diverse patterns and at the same time cause overfitting in model training. To alleviate this issue, we propose a Value Substitution (VS) method to generate $\hat{U}_t^{\text{usr}}$.

[1] `https://pytorch.org/hub/pytorch_fairseq_translation`

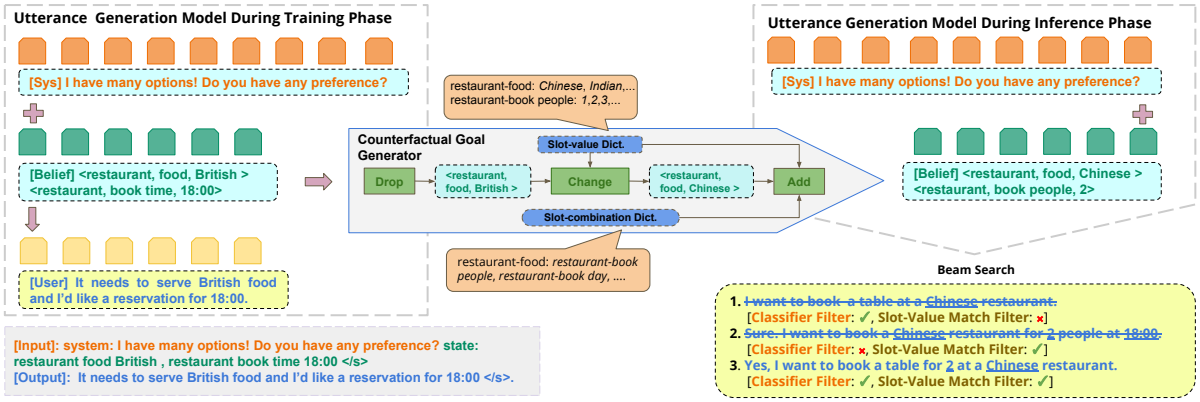

Figure 4.2: The overall pipeline of CoCo. The very left part represents the training phase of utterance generation model, where the concatenation of U_t^{sys} and L_t is processed by the encoder, which the decoder then conditions on to generate the user utterance U_t^{usr}. The input and output of this model is shown within the box at the lower-left. The right part depicts the inference phase, where the counterfactual goal generator first modifies the original belief L_t fed from the left part into a new one $\hat{L}_t$, which is then fed to the trained utterance generator along with the same conversation history to generate $\hat{U}_t^{\text{usr}}$ by beam search followed by filtering undesired utterances. Note that conversational turns in inference phase don't have to originate from training phase.

Specifically, for each value of S_j in L_t, if the value only appears in U_t^{usr} rather than U_t^{sys}, we allow it to be substituted. Otherwise, we keep it as is. This heuristic is based on the following three observations: (1) if the value comes from U_t^{sys}, e.g. TOD system's recommendation of restaurant food, changing it may make the dialogue flow less natural and coherent (2) if it never appears in the dialogue flow, e.g. *yes* of *hotel-parking*, changing it may cause belief state label errors (3) if it only appears in U_t^{usr}, it is expected that changing the value won't cause issues in (1) and (2).

For values that can be substituted, new values are sampled from a *Slot-Value Dictionary*, a predefined value set for each domain-slot. These new values are then used to update their counterparts in U_t^{usr}, L_t and B_t. We defer the details of slot-value dictionary to section 4.4.2. After the update, we get $\hat{U}_t^{\text{usr}}$, $\hat{L}_t$ and $\hat{B}_t$, and can use $< \hat{X}_t, \hat{L}_t, \hat{B}_t >$ to evaluate or augment the performance of DST models depending on the set where it is derived. An example of how VS works is illustrated in the lower part of Figure 4.1. At the second turn, as *British* and *18:00* are in

L_2 and only appear in U_2^{usr} rather than U_2^{sys}, we can replace them with *Chinese* and *17:00* that are sampled from a slot-value dictionary, respectively, to get $\hat{U}_2^{\text{usr}}$, $\hat{L}_2$ and $\hat{X}_2$ without interrupting the naturalness of the dialogue flow.

4.4.2 Controllable Counterfactual Generation

Back-translation (BT) and value-substitution (VS) provide controllability at different granularities. BT only provides syntactic variety while preserving the meaning, hence the belief state. VS can only replace the values of the existing slots in an utterance while still having to exactly retain all the slots. However, neither of them are able to explore conversations with even slightly modified set of slots. We propose a principled approach to unlock the capability of conversation generation that generalizes beyond just transformation of existing utterances. We cast it as a task of generating novel user utterances (U_t^{usr}) from a given conversation history (U_t^{sys}) and a turn-level user goal (L_t).

We propose to tackle this problem with a conditional generation model that utilizes a pretrained encoder-decoder architecture [40, 4] to approximate $p(U_t{}^{\text{usr}}|U_t^{\text{sys}}, L_t)$, where the concatenation of U_t^{sys} and L_t is used as input to the encoder and $U_t{}^{\text{usr}}$ is set to be the target sequence to be generated by the decoder, as illustrated in the lower-left of Figure 4.2. To learn this distribution, we factorize it by chain rule [112] and train a neural network with parameters θ to minimize the aggregated negative log-likelihood $\mathcal{J}_{\text{gen}}$ over each dialogue turn tuple $(U_t^{\text{sys}}, L_t, U_t^{\text{usr}})$ where $U_t^{\text{usr}} = (U_{t,1}^{\text{usr}}, U_{t,2}^{\text{usr}}, \ldots, U_{t,n_t}^{\text{usr}})$ and $U_{t,k}^{\text{usr}}$ is its k-th token: [2]

$$p_\theta(U_t^{\text{usr}}|U_t^{\text{sys}}, L_t) = \prod_{k=1}^{n_t} p_\theta(U_{t,k}^{\text{usr}}|U_{t,<k}^{\text{usr}}, U_t^{\text{sys}}, L_t), \quad \mathcal{J}_{\text{gen}} = -\sum_{k=1}^{n_t} \log p_\theta(U_{t,k}^{\text{usr}}|U_{t,<k}^{\text{usr}}, U_t^{\text{sys}}, L_t) \tag{4.1}$$

Once the parameters θ of the goal-conditioned utterance generation model p_θ are learned from these tuples, it gives us the unique ability to generate novel conversation turns by plugging

[2] More details can be found in Appendix B.3.1.

in an arbitrary but consistent counterfactual goal $\hat{L}_t$ derived from L_t. An example of how the counterfactual goal generator operates is shown in the middle part of Figure 4.2. The counterfactual goal generator has three components, namely operation, slot-value dictionary and slot-combination dictionary.

Operation decides to apply which combination of the following three meta-operations, namely *drop*, *change* and *add* on L_t. *Drop* is used to remove values from a non-empty slot in L_t. *Change* borrows the same operation in VS, to substitute existing values. *Add* allows us to add new domain-slot values into L_t, giving us the power of generating valid but more complicated $\hat{U}_t^{\text{usr}}$.

Slot-Value Dictionary has a pre-defined value set S_j^{val} for each S_j. Once *change* and/or *add* meta-operation is activated for S_j, counterfactual goal generator will randomly sample a value from S_j^{val}.

Slot-Combination Dictionary has a predefined domain-slot set S_j^{add} for each S_j. When *add* meta-operation is activated, counterfactual goal generator will sample a domain-slot from the intersection among all S_j^{add}, where S_j has non-empty values within L_t. Once a new domains-slot is sampled, its value will then be sampled from its corresponding value set as defined in slot-value dictionary.

Given L_t, the counterfactual goal generator first takes L_t as its input, and sequentially applies *drop*, *change* and *add* to output $\hat{L}_t$. Given $\hat{L}_t$ and U_t^{sys}, we can sample $\hat{U}_t^{\text{usr}} \sim p_\theta(\hat{U}_t^{\text{usr}}|U_t^{\text{sys}}, \hat{L}_t)$ with beam search. We use a rule-based method to get $\hat{B}_t$ of $\hat{X}_t$. Specifically, we obtain $\bar{B}_{t-1}$ by calculating the set difference of B_t and L_t. Given $\bar{B}_{t-1}$ and $\hat{L}_t$, we update the domain-slot in $\bar{B}_{t-1}$ if its value in $\hat{L}_t$ is not *none*, otherwise we keep its value as it is in $\bar{B}_{t-1}$ following [113]. After the update, we get $\hat{B}_t$ and use it as the dialogue-level label of $\hat{X}_t$.

4.4.3 Filtering

We have presented methods to generate $\hat{U}_t^{\text{usr}}$, but how do we make sure that the generated utterance correctly reflects the user goal represented by $\hat{L}_t$? To motivate our methods, we take an example generated by beam search located at the lower right of Figure 4.2 for illustration. In this example, the first hypothesis doesn't include value *2* for *restaurant-book people* that is within $\hat{L}_t$. On the contrary, the second hypothesis includes value *18:00* for *restaurant-book time* that is not part of $\hat{L}_t$. We call these two phenomenons *de-generation* and *over-generation*, respectively. Filtering candidates with these issues is thus an important step to make sure (U_t^{sys}, $\hat{U}_t^{\text{usr}}$) perfectly expresses the user goals in $\hat{L}_t$. We propose two filtering methods, namely *slot-value match filter* and *classifier filter*, to alleviate *de-generation* and *over-generation* issues, respectively.

Slot-Value Match Filter. To tackle with *de-generation* issue, we choose a subset of values in $\hat{L}_t$ (values that should only appear in $\hat{U}_t^{\text{usr}}$ rather than U_t^{sys}) to eliminate candidates that fail to contain all the values in the subset.[3] In Figure 4.2, the first hypothesis from the beam search output will be eliminated by this filter because it does not include the value *2* for *restaurant-book people* in $\hat{L}_t$.

Classifier Filter. As shown in Table 4.2, the slot *restaurant-book people* frequently appears together with *restaurant-book time* in the data used to train our generation model $p_\theta(\hat{U}_t^{\text{usr}}|U_t^{\text{sys}}, \hat{L}_t)$, which may cause the resulting generation model to fall into *over-generation* issue. To deal with this *over-generation* problem, we propose to use a N-way multi-label classifier to eliminate such candidates. We employ BERT-base [2] as its backbone:

$$H_t^{\texttt{CLS}} = \texttt{BERT}([\texttt{CLS}] \oplus [X_{t-1}] \oplus [\texttt{SEP}] \oplus [U_t^{\text{sys}}] \oplus [U_t^{\text{usr}}]) \in \mathbb{R}^{d_{\text{emb}}} \tag{4.2}$$

[3]For *hotel-parking* and *hotel-internet*, we use *parking* and *wifi* as their corresponding values for filtering.

where $H_t^{\texttt{CLS}} \in \mathbb{R}^{d_{\text{emb}}}$ is the representations of `CLS` token of BERT with dimension d_{emb}. We then feed $H_t^{\texttt{CLS}}$ into a linear projection layer followed by `Sigmoid` function:

$$P = \texttt{Sigmoid}(W(H_t^{\texttt{CLS}})) \in \mathbb{R}^N, \;\; \mathcal{J}_{\text{cls}} = -\frac{1}{N}\sum_{j=1}^{N}(Y_j \cdot \log P_j + (1-Y_j)\cdot \log(1-P_j)) \quad (4.3)$$

where $W \in \mathbb{R}^{N \times d_{\text{emb}}}$ is the trainable weight of the linear projection layer and P_j is probability that slot S_j appears at t-th turn of X_t with Y_j as its label. The classifier is trained with $\mathcal{J}_{\text{cls}}$, i.e. the mean binary cross entropy loss of every slot S_j and achieves a precision of 92.3% and a recall of 93.5% on the development set [4]. During inference, the classifier takes $\hat{X}_t$ as input and predicts whether a slot S_i appears at t-th turn or not with threshold 0.5. We use this filter to eliminate generated candidates for which the classifier predicts at least one slot S_j mentioned in $(U_t^{\text{sys}}, \hat{U}_t^{\text{usr}})$ while $S_j \notin \hat{L}_t$. In Figure 4.2, our classifier filter eliminates the second hypothesis from the output of beam search because $\hat{L}_t$ does not contain the slot *restaurant-book time* while it is mentioned in the generated utterance.

4.5 Experiments

4.5.1 Experimental Setup

We consider three strong multi-domain DST models to evaluate the effect of CoCo-generated counterfactual conversations in several scenarios. TRADE [87] builds upon pointer generator network and contains a slot classification gate and a state generator module to generate states. TRIPPY [88] introduces a classification gate and a triple copy module, allowing the model to copy values either from the conversation context or previous turns' predictions or system informs. SIMPLETOD [89] models DST as a conditional generation problem with conversation history as its condition and belief state as its target and finetunes on GPT2.

[4] We defer further details of the classifier to Appendix B.3.2.

Evaluation. We train each of these three models following their publicly released implementations on MultiWOZ 2.1 [60]. We use the joint goal accuracy to evaluate the performance of DST models. It is 1.0 if and only if the set of (*domain-slot*, value) pairs in the model output exactly matches the oracle one, otherwise 0.

Slot-Value Dictionary. We carefully design two sets of slot-value dictionaries to capture the effect of unseen slot values from two perspectives, namely *in-domain* (*I*) and *out-of-domain* (*O*). *I* is a dictionary that maps each slot to a set of values that appear in MultiWOZ test set, but not in the training set.[5] On the other hand, we construct *O* using external values (e.g., hotel names from Wikipedia) that fall completely outside of the MultiWOZ data for the slots (e.g., *hotel-name*, *restaurant-name*, etc.). Otherwise, we follow a similar fall-back strategy for slots (e.g., *hotel-internet*) with no possible external values beyond the ones (e.g., *yes* and *no*) in the original data.

Slot-Combination Dictionary. As illustrated in Table 4.2, held-out evaluation set follows almost the same slot co-occurrence distribution with training data. This makes it difficult to estimate how well DST models would generalize on the valid conversation scenarios that just do not obey the same distribution. CoCo's flexibility at generating a conversation for an arbitrary turn-level belief state naturally allows us to seek an answer to this question. To this end, we design three slot combination dictionaries, namely *freq*, *neu* and *rare*. A slot combination dictionary directly controls how different slots can be combined while generating counterfactual goals. As suggested by their names, *freq* contains frequently co-occurring slot combinations (e.g., *book people* is combined only with *book day* and *book time* slots), while *rare* is the opposite of *freq* grouping rarely co-occurring slots together, and *neu* is more neutral allowing

[5]When this set is empty for a slot (e.g., *hotel-area*), we use the set of all possible values (e.g., *center*, *east*, *west*, *south*, *north*) for this slot from training data. Please see Appendix B.7 for further details.

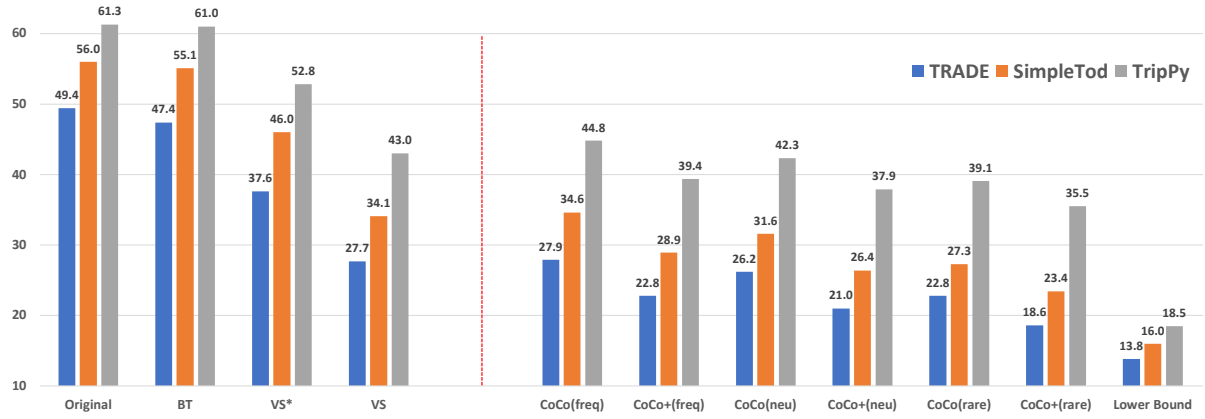

Figure 4.3: Joint goal accuracy (%) across different methods. "Original" refers to the results on the original held-out test set. * denotes results obtained from in-domain unseen slot-value dictionary (*I*). VS, CoCo and CoCo+ results use out-of-domain slot-value dictionary (*O*). For brevity, we omit CoCo and CoCo+ results using in-domain slot-value dictionary. See Appendix B.2 for the full results. *freq*, *neu*, and *rare* indicate which slot-combination dictionary is used. Lower bound refers to the percentage of correct predictions on turns with empty turn-level belief state over original held-out test set.

any meaningful combination within the same domain.[6]

4.5.2 Generalization Evaluation

Before reporting our results, it is important to note that several different post-processing strategies are used by different DST models. To make a fair comparison across different models, we follow the same post-processing strategy employed by SimpleTod evaluation script for TRADE and TripPy as well. We summarize our generalization evaluation results of different DST models on original training set in Figure 4.3. While all three DST models are quite robust to back-translation (BT) [7], their performance significantly drop on counterfactual conversations generated by each of VS, CoCo and CoCo+ compared to MultiWOZ held-out set accuracy (original).

[6]Please see Appendix B.6 for further details.

[7]Similar to CoCo, we back-translate only the turns with non-empty turn-level belief states and apply slot-value match filter. We fall back to original user utterance if none of the paraphrases passes the filter.

Unseen Slot-Value Generalization. We analyze the effect of unseen slot values for the two dictionaries (*I* and *O*) introduced in the previous section compared to the original set of slot values that have large overlap with the training data. Results presented on the left part of Figure 4.3 show that the performance of DST models significantly drops up to 11.8% compared to original accuracy even on the simple counterfactuals generated by VS strategy using in-domain unseen slot-value dictionary (I). Furthermore, using out-of-domain slot-value dictionary (O) results in about 10% additional drop in accuracy consistently across the three models. Consistent and similar drop in accuracy suggests that TRADE, SIMPLETOD, and TRIPPY are almost equally susceptible to unseen slot values.

Generalization to Novel Scenarios. The right section of Figure 4.3 presents the novel scenario generalization results. Based on these results, we see that state-of-the-art DST models are having a serious difficulty generalizing to novel scenarios generated by both COCO and COCO+ using three different slot combination strategies. The generalization difficulty becomes even more serious on counterfactuals generated by COCO+. As expected, the performance drop consistently increases as we start combining less and less frequently co-occurring slots (ranging from *freq* to *rare*) while generating our counterfactual goals. In particular, COCO+(rare) counterfactuals drops the accuracy of TRADE from 49.4% to 18.6%, pushing its performance very close to its lower bound of 13.8%. Even the performance of the most robust model (TRIPPY) among the three drops by up to 25.8%, concluding that held-out accuracy for state-of-the-art DST models may not sufficiently reflect their generalization capabilities.

Transferability Across Models. A significant difference and advantage of our proposed approach lies in its model-agnostic nature, making it immediately applicable for evaluation of any DST model. As can be inferred from Figure 4.3, the effect of COCO-generated counterfactuals on the joint goal accuracy is quite consistent across all three DST models. This result empirically

proves the transferability of CoCo, strengthening its reliability and applicability to be generally employed as a robustness evaluation of DST models by the future research.

	Human likeliness	Correctness
Human	87%	85%
CoCo(ori)	90%	91%
CoCo(freq)	90%	99%
CoCo(neu)	79%	98%
CoCo(rare)	82%	96%

Table 4.3: Human evaluation.

4.5.3 Human Evaluation

We next examine the quality of our generated data from two perspectives: "human likeliness" and "turn-level belief state correctness". The human likeliness evaluates whether a user utterance is fluent and consistent with its dialog context. The turn-level belief state correctness evaluates whether (U_t^{sys}, $\hat{U}_t^{\text{usr}}$) exactly expresses goals in $\hat{L}_t$. Both metrics are based on binary evaluation. We randomly sample 100 turns in the original test data and their corresponding CoCo-generated ones. For the CoCo-generated data, we have two different settings to examine its quality. The first is to use the original turn-level belief state to generate user utterance, denoted by CoCo(ori). The second setting is to verify the quality of the conversations generated by CoCo(freq)-, CoCo(neu)- and CoCo(rare) as they hurt the DST models' accuracy significantly as shown in Figure 4.3. For each result row reported in Table 4.3, we ask three individuals with proficient English and advanced NLP background to conduct the evaluation, and use majority voting to determine the final scores.

We can see that CoCo(ori) generated conversations are almost as human-like as original conversations and have slightly higher correctness score than the original utterances. In addition,

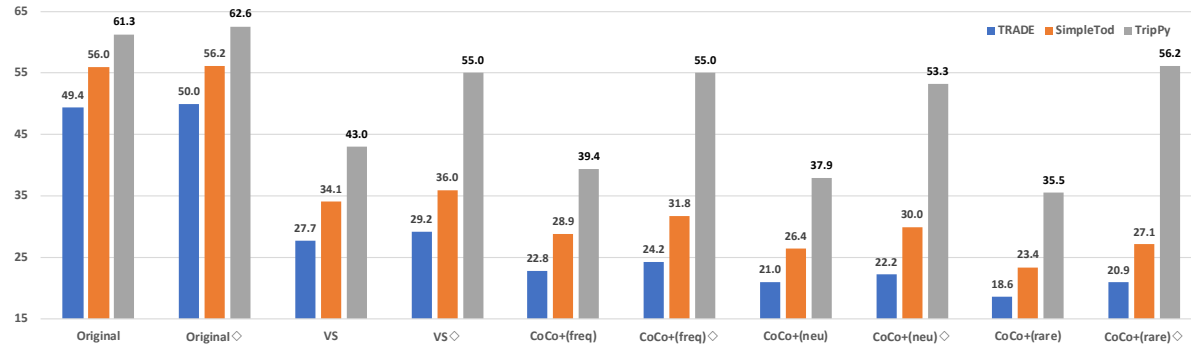

Figure 4.4: Comparison of retrained DST models (indicated by ◇) on CoCo+(rare)-augmented training data with their counterparts trained on original MultiWOZ train split.

all three variants of the CoCo-generated conversations consistently outperform human response in terms of the turn-level belief state correctness. Although CoCo(neu) and CoCo(rare) are slightly less human-like than the original human response, CoCo(freq)-generated utterances have similar human-likeness as original ones. These results demonstrate the effectiveness of our proposed approach in generating not only high-fidelity but also human-like user utterances.

4.5.4 Analysis of CoCo+ as Data Augmentation Method

So far, we have focused on the generalization capability of DST models on CoCo-generated conversations using different slot-value and slot-combination dictionaries. We have observed that all three DST models are consistently most susceptible to conversations generated by CoCo+(rare) strategy. Instead, we now seek to answer the following question: *Would using conversations generated by* CoCo+*(rare) to augment the training data help these DST models in better generalization?* Towards exploring this direction in a principled way, we design a new slot value dictionary (*train-O*) similar to out-of-domain unseen slot-value dictionary (*O*). For a fair comparison, we make sure that the slot values in *train-O* (please refer to Appendix B.7 for the complete dictionary) do not overlap with the one (*O*) used for generating test conversations.

We first retrain each DST model on the MultiWOZ training split augmented with CoCo+(rare)-

generated conversations using *train-O* slot-value dictionary. Retrained DST models are then evaluated on original test set as well as on the counterfactual test sets generated by VS and various versions of CoCo+. Results presented in Figure 4.4 show that retraining on the CoCo+(rare)-augmented training data improves the robustness of all three DST models across the board. Most notably, it rebounds the performance of TripPy on CoCo+(rare)-generated test set from 35.5% to 56.2%, significantly closing the gap with its performance (61.3%) on the original test set. To better understand such performance gain on TripPy, we compare the slot-level accuracy between original model and its counterpart after data augmentation. Results are shown in Figure 4.5. Comparison of *blue* and *orange* lines reveals that counterfactuals generated by CoCo+(rare) consistently drops the performance of TripPy model (trained on the original MultiWOZ train split) across all the slots, significantly hurting the accuracy of most slots in *train* domain along with *book day* slot for *hotel* domain. On the other hand, comparing *green* and *orange* lines clearly demonstrates the effectiveness of CoCo+(rare) as a data augmentation method, assisting TripPy in recovering from most of the errors it made on CoCo+(rare) evaluation set. In fact, it rebounds the joint goal accuracy of TripPy from 35.5% to 56.2% as presented more quantitatively in Figure 4.4.

In addition, we observe that all three retrained DST models obtain an improved joint goal accuracy on the original MultiWOZ test set compared to their counterparts trained only on the original MultiWOZ train split, further validating the quality of CoCo-generated conversations and its generalizability and effectiveness as a data augmentation method for DST. Finally, we would like to highlight that retrained TripPy achieves 62.6% joint goal accuracy, improving the previous state-of-the-art by 1.3%.

Figure 4.5: Slot-level accuracy analysis of TRIPPY. "Ori-TripPy-Clean" (blue) and "Ori-TripPy-CoCo+(rare)" (orange) denote TRIPPY (trained on original MultiWOZ training data) when evaluated against original test set and COCO+(rare) generated test set, respectively. "Aug-TripPy-CoCo+(rare)" (green) indicates slot-level accuracy of TRIPPY after data augmentation (see Section 4.5.4 for further details) when evaluated against test set generated by COCO+(rare).

4.5.5 CoCo+ Multi-round Data Augmentation on TripPy

Section 4.5.4 shows that CoCo+ as data augmentation (COCOAUG) improves TRIPPY's joint goal accuracy by 1.3% when evaluated on the original test set following the post-processing strategy employed by SIMPLETOD. In this section, we further extend previous single-round data augmentation into multiple rounds. Specifically, for each tuple $< X_t, L_t, B_t >$ in the

Model	JOINT GOAL ACCURACY
DSTreader [92]	36.40%†
TRADE [87]	45.60% †
MA-DST [114]	51.04% †
NA-DST [115]	49.04% †
DST-picklist [93]	53.30% †
SST [116]	55.23% †
MinTL(T5-small) [117]	50.95% §
SimpleTOD [89]	55.76% §
ConvBERT-DG+Multi [118]	58.70% §¶
TRIPPY [88]	55.04%*
+ COCOAUG (1×)	56.00%
+ COCOAUG (2×)	56.94%
+ COCOAUG (4×)	59.73%
+ COCOAUG (8×)	**60.53**%

Table 4.4: Joint goal accuracy results on MultiWOZ 2.1 [60] of different methods. The upper part are results of various baselines and lower part are results of TRIPPY without or with $\{1, 2, 4, 8\}$ times data augmentation size over original training data. †: results reported from [93]. §: results reported in their original papers. $*$: results of our run based on their officially released code. ¶: results need open-domain dialogues and DialoGLUE data.

original training set, we can generate multiple $< \hat{X}_t, \hat{L}_t, \hat{B}_t >$ by sampling $\hat{L}_t$ multiple times and utilizing CoCo+ to generate corresponding $\hat{X}_t$ and $\hat{B}_t$. With this approach, generated multiple $< \hat{X}_t, \hat{L}_t, \hat{B}_t >$ combined with original $< X_t, L_t, B_t >$ can be used to train DST models.

We experiment with $\{1, 2, 4, 8\}$ times data augmentation size over original training data on TRIPPY following its own default cleaning so that results with previous methods are comparable. Comparison results with different baselines and data augmentation sizes are summarized in Table 4.4. When using more and more CoCo+ generated training data, TRIPPY gains benefits from more training data and consistently improves over strong baselines. When using 8x CoCo+ generated training data, TRIPPY provides 5.49% accuracy improvement over its counterpart without data augmentation. Furthermore, it achieves the new state-of-the-art join goal accuracy[8],

[8]Code is available at `https://github.com/salesforce/coco-dst/tree/multi_fold_coco_aug`

outperforming CONVBERT-DG+MULTI, which uses open-domain dialogues and DialoGLUE [118] as additional training data.

4.6 Conclusion

We propose a principled, model-agnostic approach (COCO) to evaluate and augment dialogue state trackers. We show that state-of-the-art DST models' performance significantly drop when evaluated on the COCO-generated conversations. Human evaluations validate that they have high-fidelity and are human-like. Hence, we conclude that these strong DST models have difficulty in generalizing to novel scenarios with unseen slot values and rare slot combinations, confirming the limitations of relying only on the held-out accuracy. When utilizing COCO as a data augmentation method, it consistently improves state-of-the-art DST models not only on the COCO-generated evaluation set but also on the original test set. CoCo multi-round data augmentation further improves 5.49% accuracy over its counterpart without data augmention, making it achieve new state-of-the-art accuracy. These results prove the benefit and potential of our approach to be adopted for both evaluation and augmentation of DST models.

Chapter 5

Improving SLM Reasoning via Explanation Synthesis from LLM

5.1 Introduction

Large language models (LLMs) have achieved impressive results with in-context learning; by adding a few demonstrations as the prompts, they can solve unseen tasks without any parameter update [22, 23, 24, 25, 26, 27]. Recently, it is shown that adding explanation-augmented prompts can elicit strong performance in various reasoning tasks [28, 119], such as math word problem [120], symbolic reasoning [28], numerical reasoning [121] and commonsense reasoning tasks [8]. In addition, they also enable LLM to generate reasonable explanations to justify the reasoning outcomes [32]. However, these strong few-shot reasoning abilities only emerge when models scale to dozens or hundreds of billions of parameters [29], making it costly expensive to deploy them at scale in real-world applications [28].

Small language models (SLMs)[1] provide an alternative and could be more favorable over

[1]We argue that small and large models are relative concepts. The same model can be small or large, depending on the context.

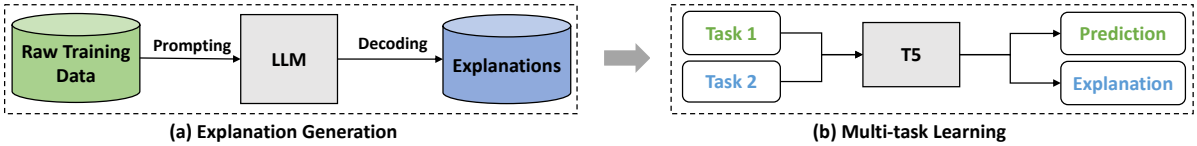

Figure 5.1: Overview of our framework. As shown in (a), we first utilize several examples with human-written explanations as demonstrations for LLM to generate explanations for the whole *training* set. Then, as shown in (b), we adopt a multi-task learning framework to utilize the LLM-generated explanations, where one task is training SLM to generate predictions while the other is training them to generate explanations as additional supervision signals. During inference, models can generate both predictions and explanations with different task prompts.

LLM in many real-world applications due to their low cost in both storage and computation. Nevertheless, one important open question is how to close the gap between LLM and SLM on complicated reasoning tasks, as is observed in [29, 30], especially in few-shot settings [31].

An intuitive way is to utilize explanations written by humans as additional training signals to improve SLM reasoning capability. Surprisingly, [122] shows that using human-annotated explanations does not improve the performance compared to standard finetuning on T5 [40]. One possible reason is that many human-annotated explanations collected via crowdsourcing [123] could be logically inconsistent and grammatically incorrect [124], which restricts the amount of available high-quality explanations. As an example, explanations of *CommonsenseQA* dataset [8] collected by [125] include many meaningless and irrelevant explanations, e.g., "rivers flow trough valleys." and "this word is most relevant", which appear hundreds of times in the training set [2]. On the other hand, using explanation-augmented prompts enables LLM to automatically generate reasonable explanations [32], making it a plausible alternative to generate an arbitrary amount of explanations quickly and cheaply [3]. Therefore, a key question is: *If we utilize high-quality explanations generated by LLM rather than the ones from humans, can they improve the reasoning capability of SLM?*

[2] `https://github.com/salesforce/cos-e/issues/2`

[3] Most powerful *Davinci* model costs 0.02 USD/1K tokens as of Jan. 19, 2023 according to `https://openai.com/api/pricing/`. In addition, decoding an explanation with one API call often takes several seconds and API calls can be executed in parallel.

In this chapter, we propose a framework leveraging explanations generated from LLM to improve reasoning capability of SLM. Our framework is shown in Figure 5.1. Specifically, we first utilize several examples with human-written explanations as demonstrations for LLM and then generate explanations for the whole *training* set. After that, we adopt a multi-task learning setup to utilize the LLM-generated explanations to facilitate SLM to acquire strong reasoning power together with explanation generation capabilities. Under this setup, one task is training SLM to generate predictions the same as standard reasoning models, while the other is training them to generate explanations as additional supervision signals. Such a setup enables the models not only to generate predictions but also to generate explanations to justify their predictions during inference. Experimental results show that our framework can consistently improve the reasoning capability of SLM with multiple explanation generation approaches as well as different multi-task learning setups. In addition, our method can outperform standard finetuning baseline by up to 8.1% in accuracy and even perform better than finetuning/prompting a 60x larger GPT-3 model (175B) [22] by up to 9.5% in accuracy on *CommonsenseQA* dataset. Finally, as a side benefit, human evaluation further shows that our method can generate high-quality explanations to justify its predictions, moving towards the goal of building more explainable AI systems [126].

In a nutshell, we summarize our contribution as follows:

- We propose a framework using explanations from LLM to improve the reasoning capabilities of SLM. Experimental results show that our framework can consistently and significantly improve strong T5 baselines, especially in few-shot settings, which is in stark contrast to the results in [122], where finetuning T5 using crowdsourced explanations only perform comparably.
- We demonstrate that our method can perform better than finetuning/prompting a 60x larger GPT-3 model (175B) by up to 9.5% in accuracy on *CommonsenseQA* dataset.

- We show that SLM trained with explanations from LLM can generate competitive explanations to justify their predictions compared to GPT-3 towards the goal of explainable AI.

5.2 Related Work

Prompting with Explanations. Recently, a new learning paradigm, *in-context learning* where several training examples are used as demonstrations for LLM without any parameter update, has shown promising results in various NLP tasks [22]. Although promising, LLM still struggles with tasks requiring strong reasoning capability [28]. To enable better few-shot in-context learning of LLM for reasoning tasks, [28] proposes chain of thought prompting, which provides intermediate reasoning steps as explanations in prompts before answers and has achieved state-of-the-art in arithmetic [120], symbolic [121] and common sense reasoning tasks [127]. [121] further extends chain of thought prompting with least-to-most prompting, which decomposes a complex problem into a list of subproblems with natural languages and then sequentially solves these subproblems in a recursive fashion. [128] moves one step further and shows that LLMs are zero-shot reasoners by simply adding "*Let's think step by step*" without any demonstration in prompts. Unlike these work, [119] explores explanations after answers prompting for LLM, where answers are fed into LLM before providing their explanations in prompts, and also observes consistent gains.

There is also existing work to utilize explanations generated from LLM rather than focusing on their final predictions. [32] explores utilizing LLM to annotate explanations for existing datasets and proposes a sample-then-filter paradigm with human annotations. [129] proposes to utilize a calibrator to calibrate GPT-3 [22] as they find that GPT-3 tends to generate consistent but less factual explanations for textual reasoning tasks. However, none of these work explores if these noisy explanations generated from LLM without human-involved filtering can be used

to improve SLM reasoning capability. Perhaps the closest work to ours is STaR [30]. STaR begins with prompting a descent large language model GPT-J with 6B parameters [130] possibly including answer hints via chain of thought prompting to generate explanations with incorrect answer rejection. After that, they utilize filtered training datasets with explanations to finetune GPT-J as a teacher model and then utilize the teacher model to generate explanations of training datasets to train a student GPT-J model iteratively with a self-training fashion until performance plateaus. However, STaR often requires dozens of iterations to converge, which is both time-consuming and compute-intensive to train a large 6B model. What's worse, their method may not be applicable to smaller language models, e.g., GPT-2 small/medium [3], as few-shot chain of thought reasoning abilities only emerge when models scale to dozens or hundreds of billions of parameters, making self-teaching infeasible. In addition, they only focus on chain of thought style prompting and finetuning while our approach can improve SLMs across model sizes, explanation generation, multi-task finetuning methods, and training data sizes.

Learning with Explanations. Learning with explanations has been commonly studied in robotics [131] and computer vision [132]. Recently, it has received increasing attention in NLP as well. [133] proposes multi-task learning with explanations for natural language inference tasks with LSTM and does not observe gains over standard single-task finetuning, i.e., direct predictions. [124] utilizes a similar setup on both T5-base and T5-11B models but mainly focuses on explanation generation. Instead, [125] observes improvements with two-stage finetuning using human-annotated explanations for common sense reasoning task, where the first stage is to train a model for explanation generations with GPT [9] and the second one utilizes explanations as input to train a classification model based on BERT [2]. However, [122] finds that both two-stage finetuning and multi-task learning with explanation setups only obtain comparable results over standard finetuning baselines on T5. We instead show that our approach can improve SLM across model sizes, explanation generation methods from

LLM, multi-task finetuning setups, and training data sizes consistently and significantly without accuracy-explanation trade-off [134].

5.3 Explanation Generation from LLM

Problem setup. Denote $D = \{(x_i, y_i)\}^N$ to be a dataset with N training instances, where x_i is a problem and y_i is its answer. Also, we have a handful of human-written instances $E = \{(x_i^p, e_i^p, y_i^p)\}^M$, where e_i^p is a free-text explanation to explain why a problem x_i^p has y_i^p as its answer and $\{(x_i^p, y_i^p)\}^M \subset D$ with $M \ll N$ (we set $M = 7$ in our experiments). Our goal is to fully leverage LLM with E as demonstrations for in-context learning to generate explanation e_i for all (x_i, y_i), where $1 \leq i \leq N$, so that we can utilize these generated explanations from LLM to improve SLM reasoning capability.

COTE. A chain of thought is a series of intermediate reasoning steps before providing an answer of a problem, mimicking the human deliberate thinking process to perform complicated reasoning tasks [28]. Chain of thought prompting provides intermediate reasoning steps as explanations before answers in prompts. Formally, for $1 \leq i \leq N$, we first concatenate all instances in E and x_i as prompt $\hat{p}_i$ = (x_1^p, e_1^p , y_1^p, ..., x_M^p, e_M^p, y_M^p, x_i). We then feed prompt $\hat{p}_i$ into LLM and greedily decode until a stop token is generated. After that, we parse the decoded sentence as explanation part $\hat{e}_i$ and prediction part $\hat{y}_i$. Intuitively, if $\hat{y}_i \neq y_i$, $\hat{e}_i$ may not have high quality as incorrect explanations tend to generate incorrect predictions [28]. Thus, we utilize ***C**hain **O**f **T**hought prompting with incorrect answer r**E**jection* (COTE) [30] by only adopting $e_i := \hat{e}_i$ if $\hat{y}_i = y_i$; otherwise, we reject $\hat{e}_i$ and set e_i as *none*.

RP. Since COTE uses the answers in original datasets to reject explanations with incorrect predictions, these instances will no longer have explanations. To alleviate this issue, an alterna-

tive is to apply ***R**ationalization **P**rompting* (RP) [32] to generate explanations for every instance in training sets. Unlike COTE, RP provides explanations given golden answers. Specifically, for $1 \leq i \leq N$, we concatenate all instances in E and (x_i, y_i) as prompt $\bar{p}_i$ = (x_1^p, y_1^p, e_1^p , ..., x_M^p, y_M^p, e_M^p, x_i, y_i). We then feed prompt $\bar{p}_i$ into LLM and greedily decode until a stop token is generated. The decoded sentence $\bar{e}_i$ is parsed and cast as explanation $\hat{e}_i$ without filtering.

CROP. COTE will possibly generate relatively high-quality explanations if LLM give correct predictions of problems at hand, as incorrect explanations tend to generate incorrect predictions [28]. However, for problems with incorrect predictions, COTE casts their explanations as *none*. On the other hand, RP can generate explanations for every instance in the dataset, but we cannot easily assess their quality without human annotation. Therefore, we propose ***C**hain of Thought with **R**ationalization Pr**O**mpting backu**P*** (CROP), where when COTE generates *none* as explanations, we will utilize RP as a backup approach. Intuitively, if LLM cannot predict a problem correctly under chain of thought prompting, the problem may be difficult [30] and RP may provide a meaningful explanation as it can access the golden label during the explanation generation process.

5.4 Multi-task Learning with Explanations

In this section, we elaborate how to utilize explanations generated from LLM to improve SLM reasoning capability with a multi-task learning framework. We detail three multi-task learning with explanations methods in the following.

MT-Re. Multi-task Learning with Reasoning (MT-Re) is introduced by [122] (see Figure 5.2 (a)). MT-Re is trained to directly generate predictions for *qta* (question to answer) task the same as standard finetuning without explanations and generate explanations without explicitly

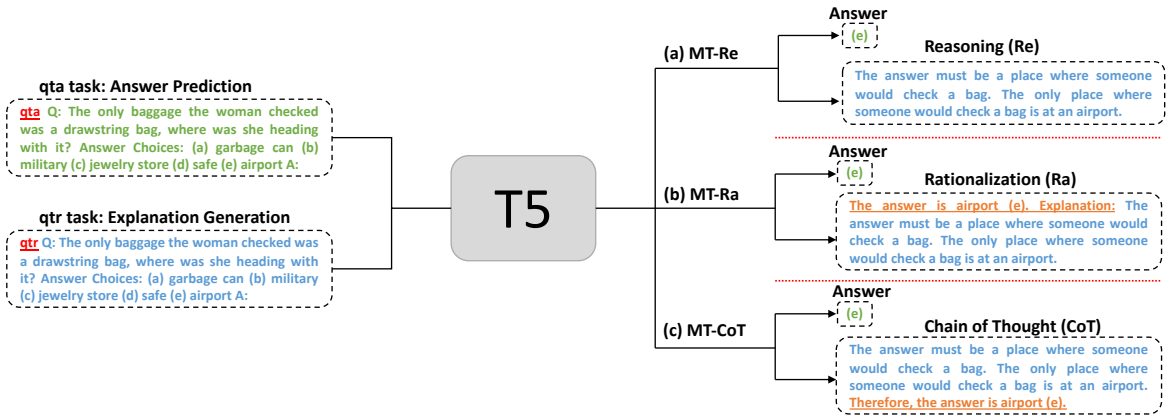

Figure 5.2: The comparison among (a) MT-Re [122], (b) MT-Ra [133] and (c) our proposed MT-CoT for multi-task learning with explanations under text-to-text format using T5. The left parts are inputs of T5, and the right is targets for different multi-task learning setups. Task *qta* (question to answer) is trained to directly generate answers for all three modes while *qtr* (question to reason) task is trained to generate reasoning, rationalization, and chain of thought for (a) MT-Re, (b) MT-Ra and (c) MT-CoT, respectively.

providing answers in *qtr* (question to reason) task. The training objective of MT-Re is to mix loss $\mathcal{L}_{\text{qta}}$ for *qta* task and $\mathcal{L}_{\text{qtr}}$ for *qtr* task:

$$\mathcal{L}_{\text{mt}} = \alpha\mathcal{L}_{\text{qta}} + (1-\alpha)\mathcal{L}_{\text{qtr}}, \tag{5.1}$$

where α weights $\mathcal{L}_{\text{qta}}$ and $\mathcal{L}_{\text{qtr}}$ loss, and is tuned on development set.

MT-Ra. Multi-task Learning with Rationalization (MT-Ra) is first proposed by [133] for natural language inference task using LSTM-based models [135], and we adopt it with a more powerful T5 model for other reasoning tasks. As shown in Figure 5.2 (b), models are trained to generate predictions for *qta* task the same as MT-Re and also trained to generate rationalization for *qtr* task. This is different from MT-Re as MT-Ra allows explanations to be explicitly conditioned on predictions. For MT-Ra, we use the same training objective as Equation 5.1 and tune α on the development set.

MT-CoT. MT-Re does not explicitly model interactions between explanations and answers during training, which may make it hard to capture their relations. While MT-Ra is explicitly trained to generate explanations conditioned on answers, it may still have difficulty understanding their causal effects as answers are never trained to explicitly access their explanations. To bridge this gap, we propose Multi-task Learning with Chain of Thought (MT-CoT), where models are trained to generate answers for *qta* task and generate chain of thought for *qtr* task, as shown in Figure 5.2 (c). For MT-CoT, we use the same training objective as Equation 5.1 and tune α on the development set.

In the MT-CoT training paradigm, models not only know answers from *qta* task but also are explicitly shown how answers are derived with intermediate reasoning steps before knowing them from *qtr* task. As we will show in experiments, this training paradigm is a supplement to MT-Re and MT-Ra, and can consistently improve small language model reasoning capability and also outperform MT-Re and MT-Ra on two datasets.

5.5 Experiments

5.5.1 Experimental setup

We evaluate our methods on three reasoning tasks.

(1) **CommonsenseQA** [8] is a 5-way multi-choice question answering dataset that requires common sense reasoning with 9741/1221/1140 for training/development/test set questions, respectively. Since its test set is not publicly available, we report results on its development set following previous work [30, 31].

(2) **StrategyQA** is a binary yes/no question answering dataset requiring implicit multi-hop reasoning steps and should be inferred using a strategy [127]. It has 2290 questions in the training set and 490 in the test set. Since its test set is not publicly available, we utilize their

split in GitHub [4], where the original training set is randomly split into 90% for the training and 10% for the development set. In our experiments, we report results on their Github development set and utilize their Github training set for training without utilizing explanations from their original annotations.

(3) **OpenbookQA** is a 4-way multi-choice question answering dataset requiring open book facts with broad common knowledge and multi-hop reasoning [136]. It has 4957/500/500 questions for training/development/test set splits, respectively, and we report results on its test set.

Explanation generation from LLM. We utilize GPT-3 *text-davinci-002* engine with official OpenAI API [5] to generate explanations through greedy decoding (by setting temperature as 0) following in-context learning paradigm. In each dataset, we have the same 7-shot examples with human-written explanations for COTE, RP, and CROP detailed in section 5.2. We defer details of prompts into Appendix C.1.

Multi-task learning with explanations. After obtaining explanations by COTE, RP, and CROP, we utilize MT-Re, MT-Ra, and MT-CoT introduced in section 5.4 to train models with explanations based on T5 on NVIDIA RTX A6000. We implement multi-task learning (MT) framework with Huggingface *transformers* library [137]. For baselines, we utilize single-task finetuning (ST) without explanations. For a fair comparison with ST, we keep hyper-parameters of multi-task learning the same as its corresponding ST except weight α, which we tune with grid search $\{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$ on development sets. When training on *none* explanations generated by COTE, we mask their loss for *qtr* task. For both ST and MT, we directly generate predictions from *qta* task for fair comparisons.

[4] https://github.com/eladsegal/strategyqa
[5] https://beta.openai.com/docs/models/gpt-3

	CommonsenseQA			**StrategyQA**			**OpenbookQA**		
	COTE	RP	CROP	COTE	RP	CROP	COTE	RP	CROP
ST		$63.05_{0.50}$			$58.60_{1.36}$			$58.08_{0.65}$	
MT-Re	$63.78_{0.43}$	$63.78_{0.20}$	$64.05_{0.22}$	$60.26_{0.92}$	$60.52_{0.81}$	$60.26_{0.62}$	$59.48_{0.93}$	$60.44_{1.49}$	$59.04_{1.63}$
MT-Ra	$\underline{64.05}_{0.60}$	$\underline{64.14}_{0.22}$	$\underline{\mathbf{64.50}}_{0.22}$	$\underline{60.52}_{0.86}$	$\underline{60.79}_{0.43}$	$60.61_{0.64}$	$58.68_{2.11}$	$59.52_{0.20}$	$\underline{60.40}_{0.59}$
MT-CoT	$63.88_{0.14}$	$63.69_{0.30}$	$63.75_{0.51}$	$60.26_{1.46}$	$\underline{60.79}_{1.31}$	$\underline{\mathbf{61.05}}_{0.85}$	$\underline{\mathbf{60.68}}_{0.37}$	$\underline{60.64}_{0.66}$	$59.64_{0.90}$

Table 5.1: Accuracy comparison (%) of single-task finetuning baselines (ST) with MT-Re, MT-Ra, and MT-CoT utilizing explanations generated by COTE, RP, and CROP. Results are averaged over five runs with their standard deviation in the subscript. The best results for each column with the same explanations are underlined and best results for each **dataset** are bold.

5.5.2 Main results

In this section, we compare results between multi-task learning with explanations and its single-task finetuning counterpart using full training data on three datasets introduced in section 5.5.1. Specifically, we generate explanations for each dataset with COTE, RP, and CROP. For each explanation generation method, we train T5-base model under MT-Re, MT-Ra and MT-CoT setups with 5 different runs in each setting. For single-task finetuning baselines, we only keep *qta* task by removing *qtr* task in the multi-task learning setup. Results are summarized in Table 5.1.

Three multi-task learning with three different explanation generation methods consistently and significantly outperform single-task finetuning baselines, showing the effectiveness of utilizing explanations from LLM. However, MT-CoT and MT-Ra have 4 and 6 underlined results, respectively, while MT-Re does not have any. We hypothesize it is because MT-CoT and MT-Ra *explicitly* mention answers by *the answer is* in *qtr* task, making it easier for T5 to model relations between explanations and answers. Considering the best results for each dataset, two of three are obtained via CROP with the remaining one obtained by COTE, showing that chain of thought prompting generates better explanations for SLM finetuning when their predictions are correct and RP backup can possibly further improve SLM reasoning capability. In addition,

two of these three best results are obtained by MT-CoT, demonstrating that our method MT-CoT can serve as a good candidate to improve SLM reasoning with explanations from the toolbox.

5.5.3 Few-shot learning results

	50	100	200	400
		CommonsenseQA		
ST	$21.92_{1.57}$	$27.06_{2.83}$	$28.04_{2.78}$	$44.49_{2.16}$
MT	$\mathbf{29.25}_{3.03}$	$\mathbf{33.28}_{3.53}$	$\mathbf{36.13}_{5.29}$	$\mathbf{46.55}_{1.53}$
α*	0.1	0.2	0.3	0.6
		OpenbookQA		
ST	$27.08_{2.96}$	$28.32_{2.88}$	$30.68_{2.10}$	$37.80_{4.64}$
MT	$\mathbf{29.76}_{3.74}$	$\mathbf{32.92}_{0.95}$	$\mathbf{34.84}_{1.27}$	$\mathbf{43.68}_{0.94}$
α*	0.1	0.1	0.2	0.2

Table 5.2: Accuracy comparison (%) between single-task finetuning (ST) and multi-task learning with explanations (MT) along with optimal α* in development sets under different training sample sizes. Results are averaged over five different training data splits with their standard deviation listed in the subscript.

We have shown the effectiveness of our method on full-training settings in section 5.5.2 and further explore if explanations can improve SLM reasoning capability under few-shot settings. We conduct few-shot learning experiments for both *CommonsenseQA* and *OpenbookQA* datasets with the best settings in section 5.5.2. Specifically, we choose MT-Ra finetuning with explanations generated by CROP for *CommonsenseQA* dataset and MT-CoT finetuning with explanations generated by COTE for *OpenbookQA* dataset. We conduct experiments with $\{50, 100, 200, 400\}$ training sample sizes for both datasets on T5-base model, and for each sample size, we randomly sample five data splits from its whole training set, and each data split has a single run. Similar to previous experiments, we have single-task finetuning as our baselines and tune α using grid search on development sets for multi-task learning experiments. Besides accuracy, we also report optimal α on development sets, denoted as α*. Intuitively,

if $\alpha*$ is small, $\mathcal{L}_{\text{qtr}}$ loss has more weight in the multi-task learning training objective listed in Equation 5.1 and hence, explanations are more important for correct prediction. We summarize our results in Table 5.2.

Multi-task learning with explanations (MT) consistently and significantly outperforms single-task finetuning baselines (ST). For *CommonsenseQA*, when training sample sizes are in $\{50, 100, 200\}$, MT significantly improves over ST with about 6%-8% absolute accuracy. For *OpenbookQA* dataset, when training sample sizes are in $\{100, 200, 400\}$, MT improves over ST with about 4%-6% absolute accuracy. More interestingly, $\alpha*$ tends to be smaller when less training data is used on both datasets. Intuitively, when training data sizes are small, models may have difficulty in learning just from the limited problem and answer pairs and hence, requires a small $\alpha*$ in the multi-task training objective 5.1, i.e., larger weight on $\mathcal{L}_{\text{qtr}}$ loss during the multi-task learning process. These consistent and significant gains show that our method not only can improve results in full-training settings but also is very useful when training data is limited.

5.5.4 Results across model sizes

	T5-small	T5-base	T5-large	T5-3B
CommonsenseQA				
ST	48.26	63.05	72.56	81.82
MT	**49.17**	**64.50**	**74.37**	**82.47**
OpenbookQA				
ST	50.36	58.08	61.60	76.60
MT	**51.72**	**60.68**	**64.60**	**78.60**

Table 5.3: Accuracy comparison (%) between ST and MT across different model sizes.

Previous experiments utilize T5-base model, and we further explore if explanations can improve language model reasoning capability across model sizes. We conduct full-training

set experiments for both *CommonsenseQA* and *OpenbookQA* datasets with best settings for each dataset in section 5.5.2 across $\{\text{T5-small}, \text{T5-base}, \text{T5-large}, \text{T5-3B}\}$. For T5-small and T5-base, we have five different runs for each setting, and their average results are reported. For T5-large and T5-3B, we only report a single run due to their intensive computational cost. Results are summarized in Table 5.3.

MT consistently improves its ST counterpart on both *CommonsenQA* and *OpenbookQA* across model sizes from T5-small (60 million parameters) to T5-3B. For *CommonsenQA*, MT improves ST by about 0.7%-1.8% absolute accuracy, and for *OpenbookQA*, MT improves ST by about 1.4%-3.0% absolute accuracy. Even for T5-3B, MT can improve strong ST with 2% absolute accuracy. These consistent results show that our approach can work on both small and relatively large models.

5.5.5 Results comparison with LLM

We further compare our method on T5-3B with state-of-the-art LLM. Specifically, we adopt GPT-J direct finetuning, its self-bootstrapping version (STaR) [30] and GPT-3 direct finetuning [6] as baselines with parameter update on downstream tasks. We also adopt GPT-3 direct prompting [22], GPT-3 chain of thought prompting [28] and GPT-3 explanations after answers prompting [119] as prompting baselines. These three prompting methods utilize the same set of demonstrations for explanation generation in section 5.2, and we defer their prompts into Appendix C.1. Results are summarized in Table 5.4.

Our approach can outperform strong 60x larger GPT-3 finetuning and various GPT-3 prompting methods on *CommonsenseQA* up to about 9.5% accuracy. Also, although STaR can outperform its GPT-J baseline with chain-of-thought style iterative finetuning, their result still has about 10% accuracy gap with our method on *CommonsenseQA* even with doubled parameter size and more compute during the iterative finetuning process. For *OpenbookQA*, our model

	CSQA	OBQA
GPT-J Direct Finetuning (6B) ◇	60.0	-
STaR (6B) ◇	72.5	-
GPT-3 Direct Finetuning (175B)*	73.0	-
GPT-3 Direct Prompting (175B)	80.59	83.00
GPT-3 Chain of Thought Prompting (175B)	73.71	72.60
GPT-3 Explanation after Answer Prompting (175B)	80.84	**83.40**
T5 MT (3B)	**82.47**	78.60

Table 5.4: Accuracy comparison (%) between T5 multi-task learning with explanations with various state-of-the-art LLM on CommonsenseQA (CSQA) and OpenbookQA (OBQA), and model sizes are listed in the parenthesis. Results with ◇ and * are from [30] and [6], respectively.

underperforms GPT-3 direct prompting and explanation after answer prompting but can still outperform GPT-3 chain of thought prompting with 6% absolute accuracy. In short, our method can achieve strong performance even compared with 60x larger GPT-3.

5.5.6 Explanation comparison with LLM

A side benefit of our model is to generate explanations towards more explainable AI to alleviate the notorious black box issue of deep neural networks [138]. Our model is trained with explanations generated by GPT-3, and we would like to know its generated explanation quality compared to that of GPT-3, which has been shown to be competitive even compared to human-written ones in [32]. Therefore, we conduct human evaluation since automatic metrics are not highly correlated with human assessment [139, 140].

Specifically, we perform a head-to-head explanation comparison on *CommonsenseQA* between T5-3B and GPT-3 175B few-shot explanations after answers prompting since these models achieve close performance on this dataset, as shown in Table 5.4. We randomly sample 100 examples that are predicted correctly by both GPT-3 and T5 from *qta* task. For each example, we present a question, its ground truth answer and two randomly shuffled and parsed explanations as (a) and (b) generated by GPT-3 and T5 from *qtr* task with greedy decoding to

Preference			Agreement Level		
T5	Tie	GPT-3	Level 0	Level 1	Level 2
14%	44%	42%	7%	56%	37%

Table 5.5: Head-to-head human explanation preference comparison between T5 (3B) and GPT-3 (175B) on *CommonsenseQA* along with their agreement percentage on three levels.

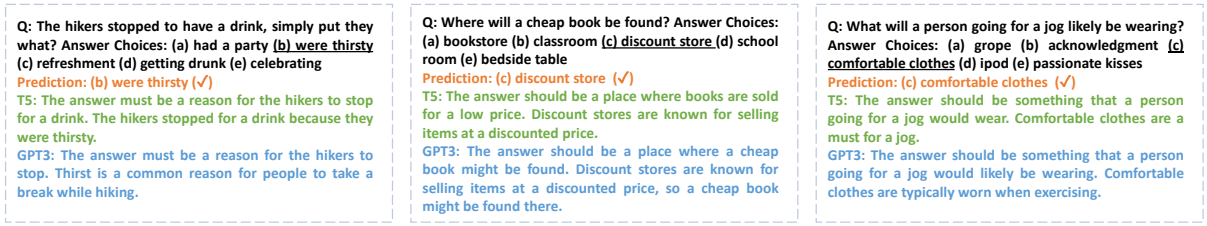

Figure 5.3: T5 and GPT-3 generated explanations used in human evaluations. In example (a), (b) and (c), T5 wins over, is tied with, and loses to GPT-3, respectively.

three different human annotators with advanced NLP backgrounds and then ask them which explanation they prefer: (a), (b) or tie, similar to [32]. Finally, we adopt majority voting to decide preference on each example if at least two annotators have the same preference; otherwise, we cast that example's two explanations are tied. In addition, we report agreement percentages across three levels. Level 0 means all three annotators have different preferences; level 1 means only two annotators have the same preference, and level 2 means all three annotators have the same preference. Results are summarized in Table 5.5.

As expected, explanations generated by T5 are less preferred over those from GPT-3, but there are still 58% (14%+44%) explanations having better or competitive quality over GPT-3. In addition, only 37% explanations are in level 2 agreement, and more than 60% explanations have disagreement (7% in level 0 + 56% in level 1). Given [32] finds that GPT-3 can generate competitive explanations even compared to human-written ones, we argue that this high disagreement is because explanations generated by both T5 and GPT-3 are high-quality, making humans hard to choose. Therefore, we choose three T5 and GPT-3 generated explanation examples used in our human evaluation experiments, as shown in Figure 5.3. Both T5 and

GPT-3 can generate plausible explanations to justify their predictions in all three examples. Even though T5 loses to GPT-3 in example (c), its explanation is still reasonably good. We also provide examples with incorrect predictions of both T5 and GPT-3 in Appendix C.2, some of which we find still have plausible predictions and explanations although they are different from golden labels. These results demonstrate that explanations generated by our model are competitive even compared to strong GPT-3 with a 60x larger size.

5.6 Conclusion

In this chapter, we leverage explanations from LLM to improve small reasoners in a multi-task learning framework. Extensive experiments on multiple reasoning tasks show our method can consistently and significantly outperform single-task finetuning baselines across explanation generation methods, multi-task learning setups, training samples, and small reasoner sizes, and can outperform strong finetuning/prompting a 60x larger GPT-3 175B on *CommonsenseQA* dataset by up to 9.5% in accuracy. In addition, human evaluation shows that our model can generate competitive explanations even compared to strong GPT-3 175B towards more explainable AI.

Chapter 6

Conclusion

NLP has many important applications in our life but state-of-the-art NLP models often require a large amount of labeled data to learn, hindering their applications in real-world problems. In this dissertation, we studied different techniques towards the goal of more label-efficient learning in NLP. We leverage unlabeled data to improve model label efficiency for NLU tasks and showed that TAPT and ST, two major semi-supervised approaches for NLU, are complementary and their performance gains can be strongly additive. Also, we propose an approach utilizing a structured KB to generate large-scale logical reasoning question-answer pairs for improving the commonsense reasoning capability of PLMs. We further evaluate and augment state-of-the-art DST models with synthesized dialogue data generated from PLMs. Finally, we enhance the reasoning capability of SLMs by utilizing explanations generated from LLMs.

While lots of efforts have been devoted towards the goal of label-efficient learning in NLP, the research problem is far from being solved. Here we emphasize two future research directions worth exploring:

Transfer Learning from Relevant NLP Tasks In this dissertation, we focus on utilizing unlabeled and synthesized data to improve label efficiency. Another promising direction is to

leverage transfer learning from other relevant NLP tasks. Intuitively, different NLP tasks may require similar skills and learning one task may facilitate learning another task as well. As an example, [141] leverages dialogue summarization dataset to improve the task of dialogue state tracking. However, [142] shows both positive and negative transfer will happen. Therefore, how to efficiently select most useful relevant tasks of a specific given one for positive transfer learning is still worth exploring in future work.

Label-efficient Fine-tuning for NLP We have discussed different techniques to leverage additional data to improve label efficiency. Another direction is to develop label-efficient fine-tuning algorithms that can better leverage knowledge in PLMs for specific downstream tasks since different fine-tuning algorithms perform differently even using the same set of data. For example, [143] shows that prompt-based fine-tuning, which adds task templates and several demonstrations for a given input, can consistently outperform standard fine-tuning [2] in few-shot settings. However, different task templates and demonstration selection strategies can largely impact model performance [143]. Hence, finding more robust and effective ways to fine-tune PLMs that can consistently improve label efficiency under different scenarios is still worth exploring in future study.

Appendix A

Supplementary Materials for Teaching PLMs with Commonsense Reasoning via Data Synthesis from KB

A.1 Experimental Details

In this section, we describe more details of the experimental settings and the choice of hyper-parameters.

A.1.1 Experiment details of the refinement process

Handling invalid logical forms. We find that some subgraphs (3.1) sampled from KB could not generate all the 14 logical forms in Appendix A.2. For example, if $\mathcal{S}_1$ is an empty set for a specific subgraph, logical form #0 is invalid. In our implementation, we create a specific 14-dimension $0/1$-mask vector for each subgraph to indicate which logical forms are valid for sampling.

Efficiency considerations. In our implementation, we use the `torch.utils.data.Dataset` class in PyTorch [144] to generate the training data for the refinement process on-the-fly. We observe that calculating $\mathcal{S}_4$ is relatively time-consuming because we have to remove all the elements in $\mathcal{S}_1, \mathcal{S}_2$, and $\mathcal{S}_3$ from the total set for each sampled subgraph. This can be a bottleneck for the dataloader and will finally reduce the overall GPU utilization. To address this issue, we approximate $\mathcal{S}_4$ by the total set in our experiments, which is an efficient and relatively accurate approximation. Note that since the number of elements in $\mathcal{S}_1, \mathcal{S}_2$, and $\mathcal{S}_3$ is much smaller than that in the total set, the chance of sampling an element from $\mathcal{S}_1, \mathcal{S}_2$, and $\mathcal{S}_3$ is extremely small. Therefore, this could be an efficient and good approximation to sampling from $\mathcal{S}_4$.

Hyper-parameters for the refinement process. When we refine BERT, GPT, and XLNet, we only train them for one epoch. This is because we find that training too many iterations on our generated multiple-choice question answering dataset may make the model forget the pretrained language modeling capability and eventually hurt performance. We set the maximum sequence length to be 40 during refinement as it covers most of the input texts for all three pretrained models, and we set the optimizers and the learning rates to be the same as their default values. The learning rates are set to be 2×10^{-5}, 6.25×10^{-5}, and 2×10^{-5} for BERT, GPT, and XLNet, respectively. We do not tune their hyper-parameters (e.g., learning rate) due to limited resources. Note that for GPT, language model coefficient is set to be 0 during refining since we argue that the texts in our template datasets may not be as natural as the ones used for pretraining.

A.1.2 Description of the downstream tasks

CommonsenseQA dataset consists of 12,247 multi-choice question answer pairs with one correct answer and four incorrect answers requiring commonsense reasoning capability [8]. This dataset has two kinds of splits, namely *question concept split* and *random split* [8], and our experiments are conducted on the official random split. For few-shot learning experiments, we

allow models to train more epochs to make sure that they converge. Specifically, for training data size in $\{100, 200, 400, 800, 1600, 3200\}$, we train models with *epoch* $\in \{100, 50, 25, 12, 8, 8\}$, respectively and keep other settings fixed. For training on the whole dataset, we follow similar settings of officially released code [1]. For a fair comparison between baselines and our refining methods, we keep their epochs, batch sizes, and other settings the same. The only differences are parameters where baselines utilize officially pretrained models, and ours use checkpoints during the proposed refining processes.

CosmosQA dataset consists of 35.6K multiple-choice reading comprehension problems requiring commonsense reasoning capability from given contexts [74]. Therefore, it is an appropriate dataset for testing commonsense reasoning of models. We finetune baselines and our proposed methods for four epochs with learning rate 2e-5 and batch size of 36. We evaluate models on the development set in every epoch and report the best performance for each experiment.

A.2 All Logical Forms with Example Multiple-Choice Questions

In this appendix, we show all the 14 logical relations that could be sampled from a particular triple pair, and the examples for the corresponding generated multiple-choice questions. Specifically, we consider the following example of triple pair:

$$(\texttt{arise} \overset{\texttt{Antonym}}{\longrightarrow} \texttt{sit}, \texttt{sit} \overset{\texttt{RelatedTo}}{\longrightarrow} \texttt{sit up})$$

Then, all the 14 logical forms and the corresponding example questions are given below, where the correct answer is highlighted in red and bolded:

[1] https://github.com/jonathanherzig/commonsenseqa/tree/master/bert.

- logical form #0: $\mathcal{S}_1$

$$(A \xrightarrow{R_1} ?\,) \wedge \neg(?\, \xrightarrow{R_2} C) \tag{A.1}$$

Q: *which of the following is an antonym of arise and meanwhile is not related to sit up ?*

A: ***set***

B: *fancifying*

C: *storing space shuttle*

- logical form #1: $\mathcal{S}_2$

$$(A \xrightarrow{R_1} ?\,) \wedge (?\, \xrightarrow{R_2} C) \tag{A.2}$$

Q: *which of the following is an antonym of arise and meanwhile is related to sit up ?*

A: ***sit***

B: *sitting up*

C: *stand up*

- logical form #2: $\mathcal{S}_1 \cup \mathcal{S}_2$

$$(A \xrightarrow{R_1} ?\,) \tag{A.3}$$

Q: *which of the following is an antonym of arise ?*

A: *promegapoietin*

B: *sleigher*

C: ***set***

- logical form #3: $\mathcal{S}_3$

$$\neg(A \xrightarrow{R_1} ?\,) \wedge (?\, \xrightarrow{R_2} C) \tag{A.4}$$

Q: *which of the following is not an antonym of arise and meanwhile is related to sit up ?*

A: *craftist*

B: *queer anarchism*

C: ***stand up***

- logical form #4: $\mathcal{S}_1 \cup \mathcal{S}_3$

$$((A \xrightarrow{R_1} ?\,) \vee (?\, \xrightarrow{R_2} C)) \wedge \neg((A \xrightarrow{R_1} ?\,) \wedge (?\, \xrightarrow{R_2} C)) \tag{A.5}$$

Q: *which of the following is an antonym of arise or is related to sit up, but not both of them ?*

A: *sit down*

B: *make refreshing dessert*

C: ***lower***

- logical form #5: $\mathcal{S}_2 \cup \mathcal{S}_3$

$$(? \xrightarrow{R_2} C) \tag{A.6}$$

Q: *which of the following is related to sit up ?*

A: *lay*

B: ***sitting up***

C: *descend*

- logical form #6: $\mathcal{S}_1 \cup \mathcal{S}_2 \cup \mathcal{S}_3$

$$(A \xrightarrow{R_1} ?) \vee (? \xrightarrow{R_2} C) \tag{A.7}$$

Q: *which of the following is an antonym of arise or is related to sit up ?*

A: *marksberrys*

B: ***sit down***

C: *previsive*

- logical form #7: $\mathcal{S}_4$

$$\neg(A \xrightarrow{R_1} ?) \wedge \neg(? \xrightarrow{R_2} C) \tag{A.8}$$

Q: *which of the following is not an antonym of arise and is not related to sit up ?*

A: *crunch*

B: ***millikin***

C: *sit*

- logical form #8: $\mathcal{S}_1 \cup \mathcal{S}_4$

$$\neg(? \xrightarrow{R_2} C) \tag{A.9}$$

Q: *which of the following is not related to sit up ?*

A: *sit down*

B: ***simpliciter***

C: *crunch*

- logical form #9: $\mathcal{S}_2 \cup \mathcal{S}_4$

$$((A \xrightarrow{R_1} ?) \wedge (? \xrightarrow{R_2} C)) \vee (\neg(A \xrightarrow{R_1} ?) \wedge \neg(? \xrightarrow{R_2} C)) \tag{A.10}$$

Q: *which of the following is an antonym of arise and is related to sit up, or neither of them ?*

A: *fall down*

B: ***cremators***

C: *lower*

- logical form #10: $\mathcal{S}_1 \cup \mathcal{S}_2 \cup \mathcal{S}_4$

$$(A \xrightarrow{R_1} ?\,) \vee \neg(?\, \xrightarrow{R_2} C) \tag{A.11}$$

Q: *which of the following is an antonym of arise or is not related to sit up ?*

A: *sitting up*

B: ***sit down***

C: *stand up*

- logical form #11: $\mathcal{S}_3 \cup \mathcal{S}_4$

$$\neg(A \xrightarrow{R_1} ?\,) \tag{A.12}$$

Q: *which of the following is not an antonym of arise ?*

A: *lay down*

B: ***free criminals***

C: *abed*

- logical form #12: $\mathcal{S}_1 \cup \mathcal{S}_3 \cup \mathcal{S}_4$

$$\neg(A \xrightarrow{R_1} ?\,) \vee \neg(?\, \xrightarrow{R_2} C) \tag{A.13}$$

Q: *which of the following is not an antonym of arise or is not related to sit up ?*

A: *sit down*

B: *sit down*

C: ***snub line***

- logical form #13: $\mathcal{S}_2 \cup \mathcal{S}_3 \cup \mathcal{S}_4$

$$\neg(A \xrightarrow{R_1} ?\,) \vee (?\, \xrightarrow{R_2} C) \tag{A.14}$$

Q: *which of the following is not an antonym of arise or is related to sit up ?*

A: *lower*

B: *fall*

C: ***sit down***

Appendix B

Supplementary Materials for Evaluating and Augmenting Dialogue State Tracking via Data Synthesis from PLM

B.1 Ablation Study on Operations

In Table B.1, we present ablation results on three meta operations (i.e., *drop*, *change*, *add*) that are used to generate counterfactual goals. The result in the first row corresponds to the performance of three DST models on evaluation set generated by CoCo including all three meta operations along with the classifier filter. Each row analyzes the effects of the corresponding meta operation or classifier by removing it from full models. From Table B.1, we observe that removing *drop* operation from full models hurts the performance of the three models further. This may indicate that the investigated DST models are more vulnerable against user utterances including more slot combinations.

CoCo	TRADE	SIMPLETOD	TRIPPY
Full	26.2	31.6	42.3
-Drop	25.7	31.1	42.1
-Add	30.4	36.0	50.4
-Change	34.1	40.9	48.3
-Classifier	25.3	30.5	41.3

Table B.1: Ablation study on the meta operations and classifier based filtering.

B.2 Full figure for Generalization evaluation

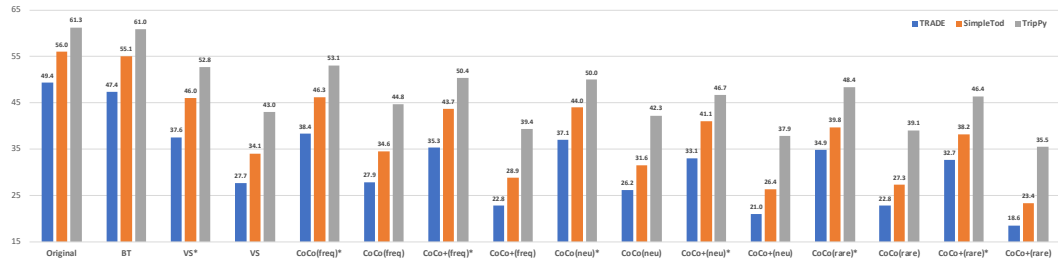

Figure B.1: Joint goal accuracy (%) across different methods. "Original" refers to the results on the original held-out test set. * denotes results obtained from in-domain unseen slot-value dictionary (*I*) while other results use out-of-domain slot-value dictionary (*O*). *freq*, *neu*, and *rare* indicate which slot-combination dictionary is used.

B.3 Model details

B.3.1 The details of controllable generation model

We instantiate $p_\theta(U_t^{\text{usr}}|U_t^{\text{sys}}, L_t)$ with `T5-small` [40] and utilize MultiWOZ 2.2 as its training data since it's cleaner than previous versions [145]. During training, we use Adam optimizer [146] with an initial learning rate $5e-5$ and set linear warmup to be 200 steps. The batch size is set to 36 and training epoch is set to be 10. The maximum sequence length of both encoder and decoder is set to be 100. We select the best checkpoint according to lowest perplexity on development set.

B.3.2 The details of classifier filter

We employ `BERT-base-uncased` as the backbone of our classifier filter and train classifier filter with Adam optimizer [146] on MultiWOZ 2.2 since it's cleaner than previous versions [145]. We select the best checkpoint based on the highest recall on development set during training process. The best checkpoint achieves a precision of 92.3% and a recall of 93.5% on the development set of MultiWOZ 2.2 and, a precision of 93.1% and a recall of 91.6% on its original test set.

B.4 Diversity Evaluation

slot name	data	area	book day	book time	food	name	price range	entropy
book people	Ori-test	2.7	36.9	37.7	1.6	18.7	2.4	0.57
	CoCo-test	3.6	38.5	25.2	15.6	14.8	2.2	0.65

Table B.2: Original test set (*Ori-test*) and CoCo generated test set (*CoCo-test*) co-occurrence distribution(%) comparisons of *book people* slot with other slots in *restaurant* domain within the same user utterance. The distribution entropy of *CoCo-test* is higher than its counterpart of *Ori-test* with an upper bound 0.78 corresponding to uniform distribution, meaning that *CoCo-test* is more diverse compared to *Ori-test* in terms of slot combinations.

Data	Distinct-1 ↑	Distinct-2 ↑	Distinct-3 ↑	Distinct-4 ↑
Ori-test	0.009	0.051	0.105	0.151
CoCo-test	0.009	0.053	0.113	0.166

Table B.3: Language diversity comparisons of data points between *Ori-test* and *CoCo-test*. We use unique n-gram ratio [147] as our diversity metric. ↑ represents a higher number means more diversity. Overall, *CoCo-test* has similar (if not better) diversity scores compared to *Ori-test*.

B.5 Generated examples by CoCo

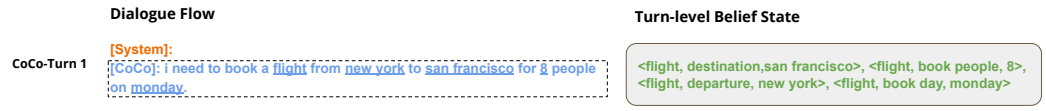

Figure B.2: Zero-shot generation ability of CoCo on *flight* domain, which is never seen during training.

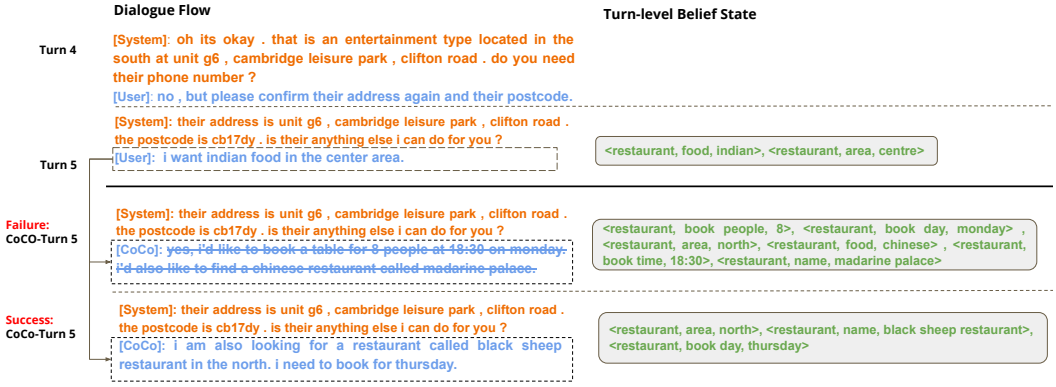

Figure B.3: A success and failure example generated by CoCo with different slot-value combinations.

Figure B.4: An example generated by CoCo with correct predictions by TRADE, SIMPLETOD and TRIPPY without retraining.

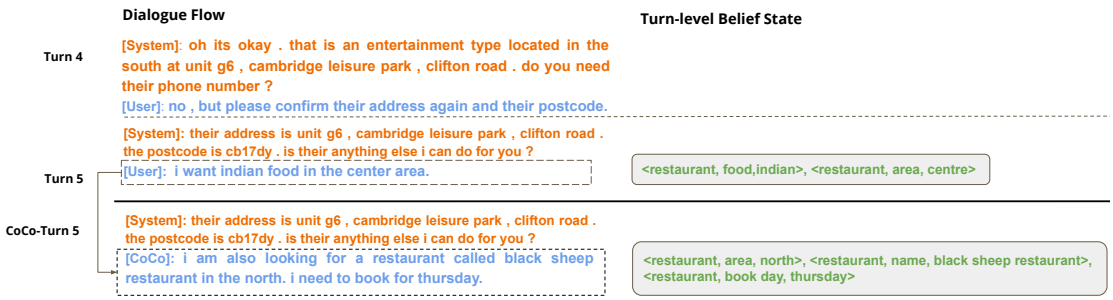

Figure B.5: An example generated by CoCo with incorrect predictions by TRADE, SIMPLETOD and TRIPPY without retraining.

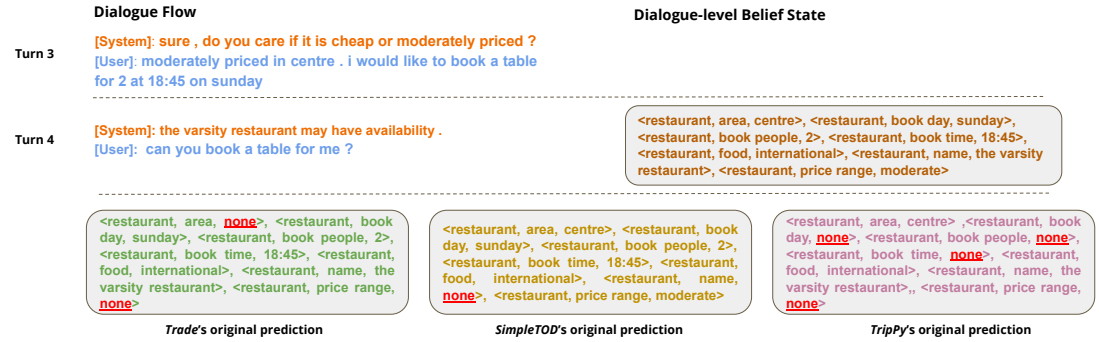

Figure B.6: An example from original MultiWOZ test set, which is predicted incorrectly by original TRADE, SIMPLETOD and TRIPPY, is corrected by their retraining counterparts.

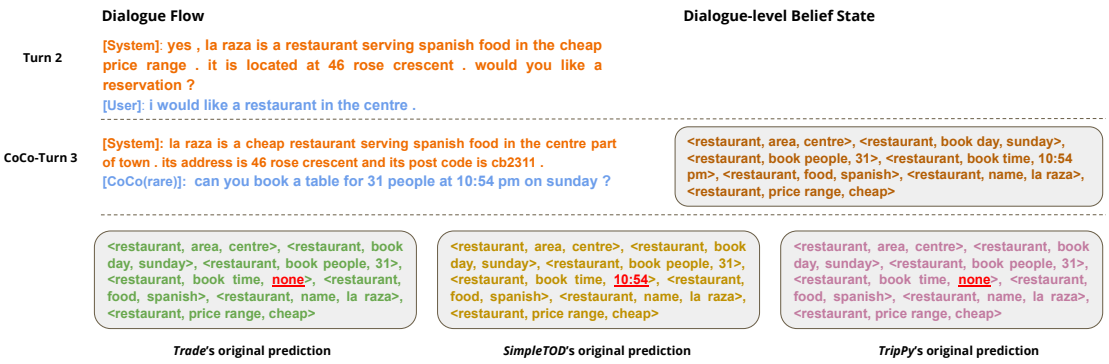

Figure B.7: An example generated by CoCo(rare) evaluation set, which is predicted incorrectly by original TRADE, SIMPLETOD and TRIPPY, is corrected by their retraining counterparts.

B.6 Slot-Combination Dictionary

Please find the different slot-combination dictionaries introduced in chapter 4 below.

domain-slot	*freq*
”hotel-internet”	[”hotel-area”,”hotel-parking”,”hotel-pricerange”,”hotel-stars”,”hotel-type”]
”hotel-type”	[”hotel-area”,”hotel-internet”,”hotel-parking”,”hotel-pricerange”,”hotel-stars”]
”hotel-parking”	[”hotel-area”,”hotel-internet”,”hotel-pricerange”,”hotel-stars”,”hotel-type”]
”hotel-pricerange”	[”hotel-area”,”hotel-internet”,”hotel-parking”,”hotel-stars”,”hotel-type”]
”hotel-book day”	[”hotel-book people”,”hotel-book stay”]
”hotel-book people”	[”hotel-book day”,”hotel-book stay”]
”hotel-book stay”	[”hotel-book day”,”hotel-book people”]
”hotel-stars”	[”hotel-area”,”hotel-internet”,”hotel-parking”,”hotel-pricerange”,”hotel-type”]
”hotel-area”	[”hotel-internet”,”hotel-parking”,”hotel-pricerange”,”hotel-stars”,”hotel-type”]
”hotel-name”	[”hotel-book day”,”hotel-book people”,”hotel-book stay”]
”restaurant-area”	[”restaurant-food”,”restaurant-pricerange”]
”restaurant-food”	[”restaurant-area”,”restaurant-pricerange”]
”restaurant-pricerange”	[”restaurant-area”,”restaurant-food”]
”restaurant-name”	[”restaurant-book day”,”restaurant-book people”,”restaurant-book time”]
”restaurant-book day”	[”restaurant-book people”,”restaurant-book time”]
”restaurant-book people”	[”restaurant-book day”,”restaurant-book time”]
”restaurant-book time”	[”restaurant-book day”,”restaurant-book people”]
”taxi-arriveby”	[”taxi-leaveat”,”train-book people”]
”taxi-leaveat”	[”taxi-arriveby”,”train-book people”]
”taxi-departure”	[”taxi-destination”,”taxi-leaveat”,”taxi-arriveby”]
”taxi-destination”	[”taxi-departure”,”taxi-arriveby”,”taxi-leaveat”]
”train-arriveby”	[”train-day”,”train-leaveat”,”train-book people”]
”train-departure”	[”train-arriveby”,”train-leaveat”,”train-destination”,”train-day”,”train-book people”]
”train-destination”	[”train-arriveby”,”train-leaveat”,”train-departure”,”train-day”,”train-book people”]
”train-day”	[”train-arriveby”,”train-leaveat”,”train-book people”]
”train-leaveat”	[”train-day”]
”train-book people”	[]
”attraction-name”	[]
”attraction-area”	[”attraction-type”]
”attraction-type”	[”attraction-area”]

Table B.4: Slot-combination dictionary for *freq* case.

slot-name	*neu*
”hotel-internet”	[”hotel-book day”,”hotel-name”,”hotel-book stay”,”hotel-pricerange”, ”hotel-stars”,”hotel-area”,”hotel-book people”,”hotel-type”,”hotel-parking”]
”hotel-area”	[”hotel-book day”,”hotel-name”,”hotel-book stay”,”hotel-pricerange”, ”hotel-stars”,”hotel-book people”,”hotel-internet”,”hotel-type”,”hotel-parking”]
”hotel-parking”	[”hotel-book day”,”hotel-name”,”hotel-book stay”,”hotel-pricerange”,”hotel-stars”, ”hotel-area”,”hotel-book people”,”hotel-internet”,”hotel-type”]
”hotel-pricerange”	[”hotel-book day”,”hotel-name”,”hotel-book stay”,”hotel-stars”,”hotel-area”, ”hotel-book people”,”hotel-internet”,”hotel-type”,”hotel-parking”]
”hotel-stars”	[”hotel-book day”,”hotel-name”,”hotel-book stay”,”hotel-pricerange”,”hotel-area”, ”hotel-book people”,”hotel-internet”,”hotel-type”,”hotel-parking”]
”hotel-type”	[”hotel-book day”,”hotel-book stay”,”hotel-pricerange”,”hotel-stars”,”hotel-area”, ”hotel-book people”,”hotel-internet”,”hotel-parking”]
”hotel-name”	[”hotel-book day”,”hotel-book stay”,”hotel-pricerange”,”hotel-stars”,”hotel-area”, ”hotel-book people”,”hotel-internet”,”hotel-parking”]
”hotel-book day”	[”hotel-name”,”hotel-book stay”,”hotel-pricerange”,”hotel-stars”,”hotel-area”, ”hotel-book people”,”hotel-internet”,”hotel-type”,”hotel-parking”]
”hotel-book people”	[”hotel-book day”,”hotel-name”,”hotel-book stay”,”hotel-pricerange”,”hotel-stars”, ”hotel-area”,”hotel-internet”,”hotel-type”,”hotel-parking”]
”hotel-book stay”	[”hotel-book day”,”hotel-name”,”hotel-pricerange”,”hotel-stars”,”hotel-area”, ”hotel-book people”,”hotel-internet”,”hotel-type”,”hotel-parking”]
”restaurant-area”	[”restaurant-book day”,”restaurant-name”,”restaurant-food”,”restaurant-book people”, ”restaurant-book time”,”restaurant-pricerange”]
”restaurant-food”	[”restaurant-book day”,”restaurant-book people”,”restaurant-book time”, ”restaurant-area”,”restaurant-pricerange”]
”restaurant-pricerange”	[”restaurant-book day”,”restaurant-name”,”restaurant-food”,”restaurant-book people”, ”restaurant-book time”,”restaurant-area”]
”restaurant-name”	[”restaurant-book day”,”restaurant-book people”,”restaurant-book time”, ”restaurant-area”,”restaurant-pricerange”]
”restaurant-book day”	[”restaurant-name”,”restaurant-food”,”restaurant-book people”,”restaurant-book time”, ”restaurant-area”,”restaurant-pricerange”]
”restaurant-book people”	[”restaurant-book day”,”restaurant-name”,”restaurant-food”,”restaurant-book time”, ”restaurant-area”,”restaurant-pricerange”]
”restaurant-book time”	[”restaurant-book day”,”restaurant-name”,”restaurant-food”,”restaurant-book people”, ”restaurant-area”,”restaurant-pricerange”]
”taxi-departure”	[”taxi-destination”, ”taxi-leaveat”, ”taxi-arriveby”]
”taxi-destination”	[”taxi-departure”, ”taxi-leaveat”, ”taxi-arriveby”]
”taxi-leaveat”	[”taxi-departure”, ”taxi-destination”, ”taxi-arriveby”]
”taxi-arriveby”	[”taxi-departure”, ”taxi-destination”, ”taxi-leaveat”]
”train-arriveby”	[”train-book people”,”train-day”,”train-leaveat”,”train-departure”,”train-destination”]
”train-leaveat”	[”train-book people”,”train-arriveby”,”train-day”,”train-departure”,”train-destination”]
”train-departure”	[”train-book people”,”train-arriveby”,”train-day”,”train-leaveat”,”train-destination”]
”train-destination”	[”train-book people”,”train-arriveby”,”train-day”,”train-leaveat”,”train-departure”]
”train-day”	[”train-book people”,”train-arriveby”,”train-leaveat”,”train-departure”,”train-destination”]
”train-book people”	[”train-arriveby”,”train-day”,”train-leaveat”,”train-departure”,”train-destination”]
”attraction-name”	[”attraction-area”]
”attraction-area”	[”attraction-type”]
”attraction-type”	[”attraction-area”]

Table B.5: Slot-combination dictionary for *neu* case.

slot-name	*rare*
”hotel-internet”	[”hotel-book people”,”hotel-book day”,”hotel-name”,”hotel-book stay”]
”hotel-area”	[”hotel-book people”,”hotel-book day”,”hotel-name”,”hotel-book stay”]
”hotel-parking”	[”hotel-book people”,”hotel-book day”,”hotel-name”,”hotel-book stay”]
”hotel-pricerange”	[”hotel-book people”,”hotel-book day”,”hotel-name”,”hotel-book stay”]
”hotel-stars”	[”hotel-book people”,”hotel-book day”,”hotel-name”,”hotel-book stay”]
”hotel-type”	[”hotel-book people”,”hotel-book day”,”hotel-book stay”]
”hotel-name”	[”hotel-pricerange”,”hotel-stars”,”hotel-area”,”hotel-internet”,”hotel-parking”]
”hotel-book day”	[”hotel-name”,”hotel-pricerange”,”hotel-stars”,”hotel-area”,”hotel-internet”, ”hotel-type”,”hotel-parking”]
”hotel-book people”	[”hotel-name”,”hotel-pricerange”,”hotel-stars”,”hotel-area”,”hotel-internet”, ”hotel-type”,”hotel-parking”]
”hotel-book stay”	[”hotel-name”,”hotel-pricerange”,”hotel-stars”,”hotel-area”,”hotel-internet”, ”hotel-type”,”hotel-parking”]
”restaurant-area”	[”restaurant-book day”,”restaurant-name”,”restaurant-book time”, ”restaurant-book people”]
”restaurant-food”	[”restaurant-book day”,”restaurant-book time”,”restaurant-book people”]
”restaurant-pricerange”	[”restaurant-book day”,”restaurant-name”,”restaurant-book time”, ”restaurant-book people”]
”restaurant-name”	[”restaurant-area”,”restaurant-pricerange”]
”restaurant-book day”	[”restaurant-name”,”restaurant-area”,”restaurant-food”,”restaurant-pricerange”]
”restaurant-book people”	[”restaurant-name”,”restaurant-area”,”restaurant-food”,”restaurant-pricerange”]
”restaurant-book time”	[”restaurant-name”,”restaurant-area”,”restaurant-food”,”restaurant-pricerange”]
”taxi-departure”	[]
”taxi-destination”	[]
”taxi-leaveat”	[”taxi-departure”, ”taxi-destination”]
”taxi-arriveby”	[”taxi-departure”, ”taxi-destination”]
”train-arriveby”	[”train-destination”, ”train-departure”]
”train-leaveat”	[”train-destination”,”train-book people”,”train-arriveby”,”train-departure”]
”train-departure”	[]
”train-destination”	[]
”train-day”	[”train-destination”, ”train-departure”]
”train-book people”	[”train-arriveby”,”train-departure”,”train-destination”,”train-day”,”train-leaveat”]
”attraction-name”	[”attraction-area”]
”attraction-area”	[”attraction-name”]
”attraction-type”	[]

Table B.6: Slot-combination dictionary for *rare* case.

B.7 Slot-Value Dictionary

Please find the different slot-value dictionaries introduced in chapter 4 below.

slot-name	train-O
”hotel-internet”	[”yes”]
”hotel-type”	[”hotel”, ”guesthouse”]
”hotel-parking”	[”yes”]
”hotel-pricerange”	[”moderate”, ”cheap”, ”expensive”]
”hotel-book day”	[”march 11th”, ”march 12th”, ”march 13th”, ”march 14th”, ”march 15th”, ”march 16th”, ”march 17th”,”march 18th”, ”march 19th”, ”march 20th”]
”hotel-book people”	[”20”,”21”,”22”,”23”,”24”,”25”,”26”,”27”,”28”,”29”]
”hotel-book stay”	[”20”,”21”,”22”,”23”,”24”,”25”,”26”,”27”,”28”,”29”]
”hotel-area”	[”south”, ”north”, ”west”, ”east”, ”centre”]
”hotel-stars”	[”0”, ”1”, ”2”, ”3”, ”4”, ”5”]
”hotel-name”	[”moody moon”, ”four seasons hotel”, ”knights inn”, ”travelodge”, ”jack summer inn”, ”paradise point resort”]
”restaurant-area”	[”south”, ”north”, ”west”, ”east”, ”centre”]
”restaurant-food”	[”asian fusion”, ”burger”, ”pasta”, ”ramen”, ”taiwanese”]
”restaurant-pricerange”:	[”moderate”, ”cheap”, ”expensive”]
”restaurant-name”	[”buddha bowls”,”pizza my heart”,”pho bistro”, ”sushiya express”,”rockfire grill”,”itsuki restaurant”]
”restaurant-book day”	[”monday”,”tuesday”,”wednesday”,”thursday”,”friday”, ”saturday”,”sunday”]
”restaurant-book people”	[”20”,”21”,”22”,”23”,”24”,”25”,”26”,”27”,”28”,”29”]
”restaurant-book time”	[”19:01”,”18:06”,”17:11”,”19:16”,”18:21”,”17:26”,”19:31”, ”18:36”,”17:41”,”19:46”,”18:51”, ”17:56”, ”7:00 pm”, ”6:07 pm”,”5:12 pm”,”7:17 pm”,”6:17 pm”,”5:27 pm”, ”7:32 pm”,”6:37 pm”, ”5:42 pm”, ”7:47 pm”, ”6:52 pm”, ”5:57 pm”, ”11:00 am”,”11:05 am”, ”11:10 am”,”11:15 am”, ”11:20 am”,”11:25 am”, ”11:30 am”,”11:35 am”,”11:40 am”, ”11:45 am”,”11:50 am”, ”11:55 am”]
”taxi-arriveby”	[”17:26”,”19:31”,”18:36”,”17:41”,”19:46”,”18:51”,”17:56”, ”7:00 pm”,”6:07 pm”,”5:12 pm”,”7:17 pm”, ”6:17 pm”, ”5:27 pm”,”11:30 am”,”11:35 am”,”11:40 am”,”11:45 am”, ”11:50 am”,”11:55 am”]
”taxi-leaveat”	[”19:01”,”18:06”,”17:11”,”19:16”,”18:21”,”7:32 pm”, ”6:37 pm”,”5:42 pm”,”7:47 pm”,”6:52 pm”, ”5:57 pm”,”11:00 am”,”11:05 am”,”11:10 am”, ”11:15 am”,”11:20 am”,”11:25 am”]
”taxi-departure”	[”moody moon”, ”four seasons hotel”, ”knights inn”, ”travelodge”, ”jack summer inn”, ”paradise point resort”]
”taxi-destination”	[”buddha bowls”,”pizza my heart”,”pho bistro”, ”sushiya express”,”rockfire grill”,”itsuki restaurant”]
”train-arriveby”	[”17:26”,”19:31”,”18:36”,”17:41”,”19:46”,”18:51”, ”17:56”,”7:00 pm”,”6:07 pm”,”5:12 pm”, ”7:17 pm”, ”6:17 pm”,”5:27 pm”,”11:30 am”,”11:35 am”,”11:40 am”, ”11:45 am”,”11:50 am”, ”11:55 am”]
”train-leaveat”	[”19:01”,”18:06”,”17:11”,”19:16”,”18:21”, ”7:32 pm”, ”6:37 pm”,”5:42 pm”,”7:47 pm”,”6:52 pm”,”5:57 pm”, ”11:00 am”,”11:05 am”,”11:10 am”,”11:15 am”,”11:20 am”,”11:25 am”]
”train-departure”	[”gilroy”,”san martin”,”morgan hill”,”blossom hill”, ”college park”,”santa clara”,”lawrence”,”sunnyvale”]
”train-destination”	[”mountain view”,”san antonio”,”palo alto”,”menlo park”, ”hayward park”,”san mateo”,”broadway”,”san bruno”]
”train-day”	[”march 11th”, ”march 12th”, ”march 13th”, ”march 14th”, ”march 15th”, ”march 16th”, ”march 17th”,”march 18th”, ”march 19th”, ”march 20th”]
”train-book people”	[”20”,”21”,”22”,”23”,”24”,”25”,”26”,”27”,”28”,”29”]
”attraction-area”	[”south”, ”north”, ”west”, ”east”, ”centre”]
”attraction-name”	[”grand canyon”,”golden gate bridge”,”niagara falls”, ”kennedy space center”,”pike place market”,”las vegas strip”]
”attraction-type”	[”historical landmark”, ”aquaria”, ”beach”, ”castle”,”art gallery”]

Table B.7: Slot value dictionary of train-O.

slot-name	I
”hotel-internet”	[”yes”]
”hotel-type”	[”hotel”, ”guesthouse”]
”hotel-parking”	[”yes”]
”hotel-pricerange”	[”moderate”, ”cheap”, ”expensive”]
”hotel-book day”	[”friday”, ”tuesday”, ”thursday”, ”saturday”, ”monday”, ”sunday”, ”wednesday”]
”hotel-book people”	[”1”, ”2”, ”3”, ”4”,”5”, ”6”, ”7”,”8”]
”hotel-book stay”	[”1”, ”2”, ”3”, ”4”,”5”, ”6”, ”7”,”8”]
”hotel-name”	[”alpha milton”, ”flinches bed and breakfast”, ”express holiday inn by cambridge”, ”wankworth house”, ”alexander b and b”, ”the gonville hotel”]
”hotel-stars”	[”0”, ”1”, ”3”, ”2”, ”4”, ”5”]
”hotel-area”	[”south”, ”east”, ”west”, ”north”, ”centre”]
”restaurant-area”	[”south”, ”east”, ”west”, ”north”, ”centre”]
”restaurant-food”	[”europeon”, ”brazliian”, ”weish”]
”restaurant-pricerange”	[”moderate”, ”cheap”, ”expensive”]
”restaurant-name”	[”pizza hut in cherry”, ”the nirala”, ”barbakan”, ”the golden house”, ”michaelhouse”, ”bridge”, ”varsity restaurant”,”loch”, ”the peking”, ”charlie”, ”cambridge lodge”, ”maharajah tandoori”]
”restaurant-book day”	[”friday”, ”tuesday”, ”thursday”, ”saturday”, ”monday”, ”sunday”, ”wednesday”]
”restaurant-book people”	[”8”, ”6”, ”7”, ”1”, ”3”, ”2”, ”4”, ”5”]
”restaurant-book time”	[”14:40”, ”19:00”, ”15:15”, ”9:30”, ”7 pm”, ”11 am”, ”8:45”]
”taxi-arriveby”	[”08:30”, ”9:45”]
”taxi-leaveat”	[”7 pm”, ”3:00”]
”taxi-departure”	[”aylesbray lodge”, ”fitzbillies”, ”uno”, ”zizzi cambridge”, ”express by holiday inn”, ”great saint marys church”, ”county folk museum”,”riverboat”, ”bishops stortford”, ”caffee uno”, ”hong house”, ”gandhi”, ”cambridge arts”, ”the hotpot”, ”regency gallery”, ”saint johns chop shop house”]
”taxi-destination”	[”ashley”, ”all saints”, ”de luca cucina and bar’s”, ”the lensfield hotel”, ”oak bistro”, ”broxbourne”, ”sleeperz hotel”, ”saint catherine’s college”]
”train-arriveby”	[”4:45 pm”, ”18:35”, ”21:08”, ”19:54”, ”10:08”, ”13:06”, ”15:24”, ”07:08”, ”16:23”, ”8:56”, ”09:01”, ”10:23”, ”10:00 am”, ”16:44”, ”6:15”, ”06:01”, ”8:54”,”21:51”, ”16:07”, ”12:43”, ”20:08”, ”08:23”, ”12:56”, ”17:23”, ”11:32”, ”20:54”, ”20:06”, ”14:24”, ”18:10”, ”20:38”, ”16:06”, ”3:00”, ”22:06”, ”20:20”, ”17:51”,”19:52”, ”7:52”, ”07:44”, ”16:08”]
”train-leaveat”	[”13:36”, ”15:17”, ”14:21”, ”3:15 pm”, ”6:10 am”, ”14:40”, ”5:40”, ”13:40”, ”17:11”, ”13:50”, ”5:11”, ”11:17”, ”5:01”, ”13:24”, ”5:35”, ”07:00”, ”8:08”, ”7:40”, ”11:54”, ”12:06”, ”07:01”, ”18:09”, ”13:17”, ”21:45”, ”06:40”, ”01:44”, ”9:17”, ”20:21”, ”20:40”, ”08:11”, ”07:35”, ”14:19”, ”1 pm”, ”19:17”, ”19:48”, ”19:50”, ”10:36”, ”09:19”, ”19:35”, ”8:06”, ”05:29”, ”17:50”, ”15:16”, ”09:17”, ”7:35”, ”5:29”, ”17:16”, ”14:01”, ”10:21”, ”05:01”, ”15:39”, ”15:01”, ”10:11”, ”08:01”]
”train-departure”	[”london liverpool street”, ”kings lynn”, ”norwich”, ”birmingham new street”, ”london kings cross”,”broxbourne”]
”train-destination”	[”bishops stortford”, ”cambridge”, ”ely”, ”stansted airport”, ”peterborough”, ”leicester”, ”stevenage”]
”train-day”	[”friday”, ”tuesday”, ”thursday”, ”monday”, ”saturday”, ”sunday”, ”wednesday”]
”train-book people”	[”9”]
”attraction-name”	[”the cambridge arts theatre”, ”the churchill college”, ”the castle galleries”, ”cambridge”, ”saint catherine’s college”, ”street”, ”corn cambridge exchange”, ”fitzwilliam”, ”cafe jello museum”]
”attraction-area”	[”south”, ”east”, ”west”, ”north”, ”centre”]
”attraction-type”	[”concerthall”, ”museum”, ”entertainment”, ”college”, ”multiple sports”, ”hiking”, ”architecture”, ”theatre”, ”cinema”, ”swimmingpool”, ”boat”, ”nightclub”, ”park”]

Table B.8: Slot-value dictionary for I case.

slot-name	*O*
”hotel-internet”	[”yes”]
”hotel-type”	[”hotel”, ”guesthouse”]
”hotel-parking”	[”yes”]
”hotel-pricerange”	[”moderate”, ”cheap”, ”expensive”]
”hotel-book day”	[”april 11th”, ”april 12th”, ”april 13th”, ”april 14th”, ”april 15th”, ”april 16th”, ”april 17th”,”april 18th”, ”april 19th”, ”april 20th”]
”hotel-book people”	[”30”,”31”,”32”,”33”,”34”,”35”,”36”,”37”,”38”,”39”]
”hotel-book stay”	[”30”,”31”,”32”,”33”,”34”,”35”,”36”,”37”,”38”,”39”]
”hotel-area”	[”south”, ”east”, ”west”, ”north”, ”centre”]
”hotel-stars”	[”0”, ”1”, ”2”, ”3”, ”4”, ”5”]
”hotel-name”	[”white rock hotel”, ”jade bay resort”, ”grand hyatt”, ”hilton garden inn”, ”cottage motel”,”mandarin oriental”],
”restaurant-area”	[”south”, ”east”, ”west”, ”north”, ”centre”]
”restaurant-food”	[”sichuan”, ”fish”, ”noodle”, ”lobster”, ”burrito”, ”dumpling”, ”curry”,”taco”]
”restaurant-pricerange”	[”moderate”, ”cheap”, ”expensive”]
”restaurant-name”	[”lure fish house”,”black sheep restaurant”,”palapa restaurant”, ”nikka ramen”, ”sun sushi”,”super cucas”]
”restaurant-book day”	[”monday”,”tuesday”,”wednesday”,”thursday”,”friday”,”saturday”,”sunday”]
”restaurant-book people”	[”30”,”31”,”32”,”33”,”34”,”35”,”36”,”37”,”38”,”39”]
”restaurant-book time”	[”20:02”,”21:07”,”22:12”,”20:17”,”21:22”,”22:27”,”20:32”,”21:37”,”22:42”, ”20:47”,”21:52”,”22:57”,”8:00 pm”,”9:04 pm”,”10:09 pm”,”8:14 pm”, ”9:19 pm”,”10:24 pm”,”8:29 pm”,”9:34 pm”,”10:39 pm”,”8:44 pm”,”9:49 pm”, ”10:54 pm”,”10:00 am”,”10:06 am”,”10:11 am”,”10:16 am”,”10:21 am”,”10:26 am”, ”10:31 am”,”10:36 am”,”10:41 am”,”10:46 am”,”10:51 am”,”10:56 am”]
”taxi-arriveby”	[”20:02”,”21:07”,”22:12”,”20:17”,”21:22”,”22:27”,”9:34 pm”,”10:39 pm”, ”8:44 pm”,”9:49 pm”,”10:54 pm”, ”10:00 am”,”10:06 am”,”10:11 am”, ”10:16 am”,”10:21 am”,”10:26 am”]
”taxi-leaveat”	[”21:37”,”22:42”,”20:47”,”21:52”,”22:57”,”8:00 pm”,”9:04 pm”,”10:09 pm”, ”8:14 pm”,”9:19 pm”,”10:24 pm”,”8:29 pm”,”10:31 am”,”10:36 am”, ”10:41 am”, ”10:46 am”,”10:51 am”,”10:56 am”]
”taxi-departure”	[”lure fish house”,”black sheep restaurant”,”palapa restaurant”, ”nikka ramen”, ”sun sushi”,”super cucas”]
”taxi-destination”	[”white rock hotel”, ”jade bay resort”, ”grand hyatt”, ”hilton garden inn”, ”cottage motel”,”mandarin oriental”]
”train-departure”	[”northridge”,”camarillo”,”oxnard”,”morepark”,”simi valley”,”chatsworth”, ”van nuys”,”glendale”]
”train-destination”	[”norwalk”,”buena park”,”fullerton”,”santa ana”,”tustin”,”irvine”, ”san clemente”,”oceanside”]
”train-arriveby”	[”20:02”,”21:07”,”22:12”,”20:17”,”21:22”,”22:27”, ”9:34 pm”,”10:39 pm”, ”8:44 pm”,”9:49 pm”,”10:54 pm”,”10:00 am”,”10:06 am”,”10:11 am”, ”10:16 am”,”10:21 am”,”10:26 am”]
”train-day”:	[”april 11th”, ”april 12th”, ”april 13th”, ”april 14th”, ”april 15th”, ”april 16th”, ”april 17th”,”april 18th”, ”april 19th”, ”april 20th”]
”train-leaveat”:	[”21:37”,”22:42”,”20:47”,”21:52”,”22:57”,”8:00 pm”,”9:04 pm”,”10:09 pm”, ”8:14 pm”,”9:19 pm”,”10:24 pm”,”8:29 pm”,”10:31 am”,”10:36 am”, ”10:41 am”,”10:46 am”,”10:51 am”,”10:56 am”]
”train-book people”:	[”30”,”31”,”32”,”33”,”34”,”35”,”36”,”37”,”38”,”39”]
”attraction-area”	[”south”, ”east”, ”west”, ”north”, ”centre”]
”attraction-name”	[”statue of liberty”,”empire state building”,”mount rushmore”, ”brooklyn bridge”,”lincoln memorial”,”times square”]
”attraction-type”	[”temple”, ”zoo”, ”library”, ”skyscraper”,”monument”]

Table B.9: Slot-value dictionary for *O* case.

Appendix C

Supplementary Materials for Improving SLM Reasoning via Explanation Synthesis from LLM

C.1 Prompt details

Here we provide prompts used in our experiments. Our prompts on *CommonsenseQA* are based on [30] while prompts on *StrategyQA* are based [28] and [127]. Explanations in prompts for *OpenbookQA* are based on science facts in *OpenbookQA* dataset Github repository `https://github.com/allenai/OpenBookQA`.

Q: What do people use to absorb extra ink from a fountain pen? Answer Choices: (a) shirt pocket (b) calligrapher's hand (c) inkwell (d) desk drawer (e) blotter
A: blotter (e)

Q: What home entertainment equipment requires cable? Answer Choices: (a) radio shack (b) substation (c) television (d) cabinet (e) desk
A: television (c)

Q: The fox walked from the city into the forest, what was it looking for? Answer Choices: (a) pretty flowers (b) hen house (c) natural habitat (d) storybook (e) dense forest
A: natural habitat (c)

Q: Sammy wanted to go to where the people were. Where might he go? Answer Choices: (a) populated areas (b) race track (c) desert (d) apartment (e) roadblock
A: populated areas (a)

Q: Where do you put your grapes just before checking out? Answer Choices: (a) mouth (b) grocery cart (c) super market (d) fruit basket (e) fruit market
A: grocery cart (b)

Q: Google Maps and other highway and street GPS services have replaced what? Answer Choices: (a) united states (b) mexico (c) countryside (d) atlas (e) oceans
A: atlas (d)

Q: Before getting a divorce, what did the wife feel who was doing all the work? Answer Choices: (a) harder (b) anguish (c) bitterness (d) tears (e) sadness
A: bitterness (c)

Table C.1: Standard prompt for CommonsenseQA, adapted from [30].

Q: What do people use to absorb extra ink from a fountain pen? Answer Choices: (a) shirt pocket (b) calligrapher's hand (c) inkwell (d) desk drawer (e) blotter
A: The answer must be used to absorb extra ink. Blotters are designed to absorb liquids. Therefore, the answer is blotter (e).

Q: What home entertainment equipment requires cable?
Answer Choices: (a) radio shack(b) substation (c) television (d) cabinet (e) desk
A: The answer must require cable. Cable is used to provide satellite channels to televisions. Therefore, the answer is television (c).

Q: The fox walked from the city into the forest, what was it looking for? Answer Choices: (a) pretty flowers (b) hen house (c) natural habitat (d) storybook (e) dense forest
A: The answer must be a reason for a fox to go into the forest. The forest is a fox's natural habitat. Therefore, the answer is natural habitat (c).

Q: Sammy wanted to go to where the people were. Where might he go? Answer Choices: (a) populated areas (b) race track (c) desert (d) apartment (e) roadblock
A: The answer must be a place with many people. Populated areas, by definition, have a lot of people. Therefore, the answer is populated areas (a).

Q: Where do you put your grapes just before checking out? Answer Choices: (a) mouth (b) grocery cart (c) super market (d) fruit basket (e) fruit market
A: The answer should be the place where grocery items are placed before checking out. Of the above choices, grocery cart makes the most sense for holding grocery items. Therefore, the answer is grocery cart (b).

Q: Google Maps and other highway and street GPS services have replaced what? Answer Choices: (a) united states (b) mexico (c) countryside (d) atlas (e) oceans
A: The answer must be something that used to do what Google Maps and GPS services do, which is give directions. Atlases were also used to give directions. Therefore, the answer is atlas (d).

Q: Before getting a divorce, what did the wife feel who was doing all the work? Answer Choices: (a) harder (b) anguish (c) bitterness (d) tears (e) sadness
A: The answer should be a feeling which would cause someone who was doing all the work to get divorced. If someone feels bitter towards their spouse, they are likely to want a divorce. Therefore, the answer is bitterness (c).

Table C.2: Chain of Thought prompt for CommonsenseQA [30].

Q: What do people use to absorb extra ink from a fountain pen? Answer Choices: (a) shirt pocket (b) calligrapher's hand (c) inkwell (d) desk drawer (e) blotter
A: blotter (e) Explanation: the answer must be used to absorb extra ink. Blotters are designed to absorb liquids.

Q: What home entertainment equipment requires cable? Answer Choices: (a) radio shack (b) substation (c) television (d) cabinet (e) desk
A: television (c) Explanation: the answer must require cable. Cable is used to provide satellite channels to televisions.

Q: The fox walked from the city into the forest, what was it looking for? Answer Choices: (a) pretty flowers (b) hen house (c) natural habitat (d) storybook (e) dense forest
A: natural habitat (c) Explanation: the answer must be a reason for a fox to go into the forest. The forest is a fox's natural habitat.

Q: Sammy wanted to go to where the people were. Where might he go? Answer Choices: (a) populated areas (b) race track (c) desert (d) apartment (e) roadblock
A: populated areas (a) Explanation: the answer must be a place with many people. Populated areas, by definition, have a lot of people.

Q: Where do you put your grapes just before checking out? Answer Choices: (a) mouth (b) grocery cart (c) super market (d) fruit basket (e) fruit market
A: grocery cart (b)

Explanation: the answer should be the place where grocery items are placed before checking out. Of the above choices, grocery cart makes the most sense for holding grocery items.

Q: Google Maps and other highway and street GPS services have replaced what? Answer Choices: (a) united states (b) mexico (c) countryside (d) atlas (e) oceans
A: atlas (d) Explanation: the answer must be something that used to do what Google Maps and GPS services do, which is give directions. Atlases were also used to give directions.

Q: Before getting a divorce, what did the wife feel who was doing all the work? Answer Choices: (a) harder (b) anguish (c) bitterness (d) tears (e) sadness
A: bitterness (c) Explanation: the answer should be a feeling which would cause someone who was doing all the work to get divorced. If someone feels bitter towards their spouse, they are likely to want a divorce.

Table C.3: Rationalization prompt for CommonsenseQA, adapted from [30].

Q: Yes or no: Do hamsters provide food for any animals?
A: Hamsters are prey animals. Prey are food for predators. Thus, hamsters provide food for some animals. So the answer is yes.

Q: Yes or no: Could the members of The Police perform lawful arrests?
A: The members of The Police were musicians, not law enforcement officers. Only law enforcement officers can perform lawful arrests. Thus, the members of The Police could not perform lawful arrests. So the answer is no.

Q: Yes or no: Hydrogen's atomic number squared exceeds number of Spice Girls?
A: Hydrogen has an atomic number of 1. 1 squared is 1. There are 5 Spice Girls. Thus, Hydrogen's atomic number squared is less than 5. So the answer is no.

Q: Yes or no: Is it common to see frost during some college commencements?
A: College commencement ceremonies can happen in December, May, and June. December is in the winter, so there can be frost. Thus, there could be frost at some commencements. So the answer is yes.

Q: Yes or no: Could a llama birth twice during War in Vietnam (1945-46)?
A: The War in Vietnam was 6 months. The gestation period for a llama is 11 months, which is more than 6 months. Thus, a llama could not give birth twice during the War in Vietnam. So the answer is no.

Q: Yes or no: Would a pear sink in water?
A: The density of a pear is about 0.6 g/cm^3, which is less than water. Objects less dense than water float. Thus, a pear would float. So the answer is no.

Q: Yes or no: Is shrimp scampi definitely free of plastic?
A: Shrimp scampi is a dish made with shrimp. Shrimp have been found to contain microplastics. Microplastics are plastic material. Thus, shrimp scampi may contain plastic. So the answer is no.

Table C.4: Chain of Thought prompt for StrategyQA, adapted from [28] and [127]

Q: Yes or no: Do hamsters provide food for any animals?
A: yes Explanation: Hamsters are prey animals. Prey are food for predators. Thus, hamsters provide food for some animals.

Q: Yes or no: Could the members of The Police perform lawful arrests?
A: no Explanation: The members of The Police were musicians, not law enforcement officers. Only law enforcement officers can perform lawful arrests. Thus, the members of The Police could not perform lawful arrests.

Q: Yes or no: Hydrogen's atomic number squared exceeds number of Spice Girls?
A: no Explanation: Hydrogen has an atomic number of 1. 1 squared is 1. There are 5 Spice Girls. Thus, Hydrogen's atomic number squared is less than 5.

Q: Yes or no: Is it common to see frost during some college commencements?
A: yes Explanation: College commencement ceremonies can happen in December, May, and June. December is in the winter, so there can be frost. Thus, there could be frost at some commencements.

Q: Yes or no: Could a llama birth twice during War in Vietnam (1945-46)?
A: no Explanation: The War in Vietnam was 6 months. The gestation period for a llama is 11 months, which is more than 6 months. Thus, a llama could not give birth twice during the War in Vietnam.

Q: Yes or no: Would a pear sink in water?
A: no Explanation: The density of a pear is about 0.6 g/cm^3, which is less than water. Objects less dense than water float. Thus, a pear would float.

Q: Yes or no: Is shrimp scampi definitely free of plastic?
A: no Explanation: Shrimp scampi is a dish made with shrimp. Shrimp have been found to contain microplastics. Microplastics are plastic material. Thus, shrimp scampi may contain plastic.

Table C.5: Rationalization prompt for StrategyQA, adapted from [28] and [127]

Q: What is the most likely to be an effect of acid rain on an aquatic environment? Answer Choices: (a) decrease in plant life (b) increase in fish population (c) increase in plant growth (d) cleaner and clearer water
A: (a) decrease in plant life

Q: The moon's surface Answer Choices: (a) is smooth on the entire surface (b) contains large cavities cause by explosions (c) contains an internal core of cheese (d) is filled with lakes
A: (b) contains large cavities cause by explosions

Q: As a car approaches you in the night Answer Choices: (a) the headlights become more intense (b) the headlights recede into the dark (c) the headlights remain at a constant (d) the headlights turn off
A: (a) the headlights become more intense

Q: When the weather changes as it does from Christmas to Easter, Answer Choices: (a) the air may chill (b) the ground may freeze (c) the plants may die (d) the ground may warm
A: (d) the ground may warm

Q: Heat and moisture in the ocean is a good recipe for Answer Choices: (a) a violent storm (b) violent sea animals (c) condensation (d) inland storms
A: (a) a violent storm

Q: Hummingbirds take what with them Answer Choices: (a) Bees (b) energy (c) Pollen (d) Honey
A: (c) Pollen

Q: What covers over 90% of the Earth's surface and 0% of the moon's surface Answer Choices: (a) a magnesium iron silicate mineral (b) chemical element with the symbol S (c) the element with the symbol Fe (d) that which contains 2 hydrogen and 1 oxygen molecules
A: (d) that which contains 2 hydrogen and 1 oxygen molecules

Table C.6: Standard prompt for OpenbookQA, adapted from [136].

Q: What is the most likely to be an effect of acid rain on an aquatic environment? Answer Choices: (a) decrease in plant life (b) increase in fish population (c) increase in plant growth (d) cleaner and clearer water
A: Acid rain has a negative impact on water quality and decrease in water quality leads to a decrease in aquatic life. So the answer is (a) decrease in plant life.

Q: The moon's surface Answer Choices: (a) is smooth on the entire surface (b) contains large cavities cause by explosions (c) contains an internal core of cheese (d) is filled with lakes
A: The moon's surface contains many craters and craters are large cavities caused by explosions. So the answer is (b) contains large cavities cause by explosions.

Q: As a car approaches you in the night Answer Choices: (a) the headlights become more intense (b) the headlights recede into the dark (c) the headlights remain at a constant (d) the headlights turn off
A: Headlights of a car are source of light. As a source of light becomes closer , that source will appear brighter. So the answer is (a) the headlights become more intense.

Q: When the weather changes as it does from Christmas to Easter, Answer Choices: (a) the air may chill (b) the ground may freeze (c) the plants may die (d) the ground may warm
A: Christmas is in the winter season while Easter is in the beginning of spring. Weather change from winter to spring is to become warm. So the answer is (d) the ground may warm.

Q: Heat and moisture in the ocean is a good recipe for Answer Choices: (a) a violent storm (b) violent sea animals (c) condensation (d) inland storms
A: An ocean is a source of heat and moisture for a hurricane and a hurricane is a violent storm. So the answer is (a) a violent storm.

Q: Hummingbirds take what with them Answer Choices: (a) Bees (b) energy (c) Pollen (d) Honey
A: When hummingbirds dip their long bills into flowers to drink nectar, they also pick up grains of pollen. Pollen that sticks to a hummingbird's feathers and bill gets carried to the next flower it visits. So the answer is (c) Pollen.

Q: What covers over 90% of the Earth's surface and 0% of the moon's surface Answer Choices: (a) a magnesium iron silicate mineral (b) chemical element with the symbol S (c) the element with the symbol Fe (d) that which contains 2 hydrogen and 1 oxygen molecules
A: Water covers over 90% of the Earth's surface and 0% of the moon's surface, and contains 2 hydrogen and 1 oxygen molecules. So the answer is (d) that which contains 2 hydrogen and 1 oxygen molecules.

Table C.7: Chain of Thought prompt for OpenbookQA, adapted from [136].

Q: What is the most likely to be an effect of acid rain on an aquatic environment? Answer Choices: (a) decrease in plant life (b) increase in fish population (c) increase in plant growth (d) cleaner and clearer water
A: (a) decrease in plant life Explanation: Acid rain has a negative impact on water quality and decrease in water quality leads to a decrease in aquatic life.

Q: The moon's surface Answer Choices: (a) is smooth on the entire surface (b) contains large cavities cause by explosions (c) contains an internal core of cheese (d) is filled with lakes
A: (b) contains large cavities cause by explosions Explanation: The moon's surface contains many craters and craters are large cavities caused by explosions.

Q: As a car approaches you in the night Answer Choices: (a) the headlights become more intense (b) the headlights recede into the dark (c) the headlights remain at a constant (d) the headlights turn off
A: (a) the headlights become more intense Explanation: Headlights of a car are source of light. As a source of light becomes closer , that source will appear brighter.

Q: When the weather changes as it does from Christmas to Easter, Answer Choices: (a) the air may chill (b) the ground may freeze (c) the plants may die (d) the ground may warm
A: (d) the ground may warm Explanation: Christmas is in the winter season while Easter is in the beginning of spring. Weather change from winter to spring is to become warm.

Q: Heat and moisture in the ocean is a good recipe for Answer Choices: (a) a violent storm (b) violent sea animals (c) condensation (d) inland storms
A: (a) a violent storm Explanation: An ocean is a source of heat and moisture for a hurricane and a hurricane is a violent storm.

Q: Hummingbirds take what with them Answer Choices: (a) Bees (b) energy (c) Pollen (d) Honey
A: (c) Pollen Explanation: When hummingbirds dip their long bills into flowers to drink nectar, they also pick up grains of pollen. Pollen that sticks to a hummingbird's feathers and bill gets carried to the next flower it visits.

Q: What covers over 90% of the Earth's surface and 0% of the moon's surface Answer Choices: (a) a magnesium iron silicate mineral (b) chemical element with the symbol S (c) the element with the symbol Fe (d) that which contains 2 hydrogen and 1 oxygen molecules
A: (d) that which contains 2 hydrogen and 1 oxygen molecules Explanation: Water covers over 90% of the Earth's surface and 0% of the moon's surface, and contains 2 hydrogen and 1 oxygen molecules.

Table C.8: Rationalization prompt for OpenbookQA, adapted from [136].

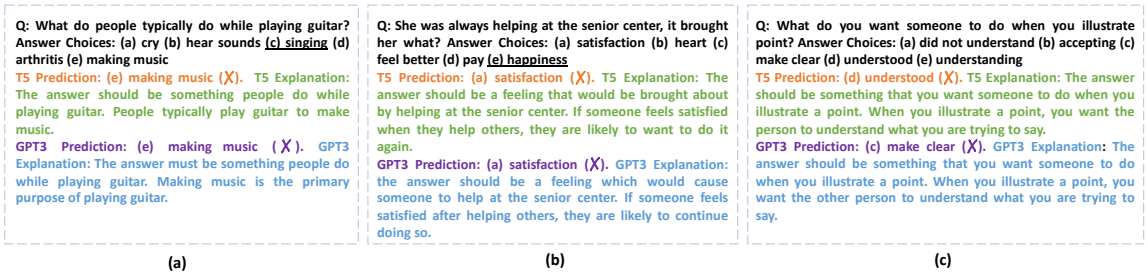

Figure C.1: T5 and GPT-3 generated explanations with incorrect predictions.

C.2 Explanation examples

Here we further provide three examples as shown in Figure C.1, where both T5 and GPT-3 have incorrect predictions. We observe that in both example (b) and (c), T5 and GPT-3 have plausible predictions and explanations, although their predictions are different from golden labels.

Bibliography

[1] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, *Distributed representations of words and phrases and their compositionality*, in *Advances in Neural Information Processing Systems* (C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Weinberger, eds.), vol. 26, Curran Associates, Inc., 2013.

[2] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, *BERT: Pre-training of deep bidirectional transformers for language understanding*, 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)*, (Minneapolis, Minnesota), pp. 4171–4186, Association for Computational Linguistics, June, 2019.

[3] A. Radford, J. Wu, R. Child, L. David, D. Amodei, and I. Sutskever, *Language models are unsupervised multitask learners.*, *OpenAI Blog* (2019).

[4] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer, *Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension*, *ArXiv* **abs/1910.13461** (2020).

[5] P. He, X. Liu, J. Gao, and W. Chen, *Deberta: Decoding-enhanced bert with disentangled attention*, *ArXiv* **abs/2006.03654** (2020).

[6] Y. Xu, C. Zhu, S. Wang, S. Sun, H. Cheng, X. Liu, J. Gao, P. He, M. Zeng, and X. Huang, *Human parity on commonsenseqa: Augmenting self-attention with external attention*, *ArXiv* **abs/2112.03254** (2021).

[7] A. Wang, Y. Pruksachatkun, N. Nangia, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman, *Superglue: A stickier benchmark for general-purpose language understanding systems*, in *NeurIPS*, 2019.

[8] A. Talmor, J. Herzig, N. Lourie, and J. Berant, *CommonsenseQA: A question answering challenge targeting commonsense knowledge*, 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)*, (Minneapolis, Minnesota), pp. 4149–4158, Association for Computational Linguistics, June, 2019.

[9] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever., *Improving language understanding by generative pre-training.*, *https://s3-us-west-2.amazonaws.com/openai-assets/ research-covers/language-unsupervised/ language understanding paper.pdf* (2018).

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

[11] Y. Wang, S. Mukherjee, H. Chu, Y. Tu, M. Wu, J. Gao, and A. H. Awadallah, *Adaptive self-training for neural sequence labeling with few labels*, 2021.

[12] P. Thomas, *Semi-supervised learning by olivier chapelle, bernhard schölkopf, and alexander zien (review)*, *IEEE Transactions on Neural Networks* **20** (01, 2009) 542.

[13] P. Liang, *Semi-supervised learning for natural language*, 2005.

[14] Q. Xie, Z. Dai, E. H. Hovy, M.-T. Luong, and Q. V. Le, *Unsupervised data augmentation for consistency training*, *arXiv: Learning* (2019).

[15] S. Gururangan, A. Marasović, S. Swayamdipta, K. Lo, I. Beltagy, D. Downey, and N. A. Smith, *Don't stop pretraining: Adapt language models to domains and tasks*, in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, (Online), pp. 8342–8360, Association for Computational Linguistics, July, 2020.

[16] H. Scudder, *Probability of error of some adaptive pattern-recognition machines*, *IEEE Transactions on Information Theory* **11** (1965), no. 3 363–371.

[17] E. Davis and G. Marcus, *Commonsense reasoning and commonsense knowledge in artificial intelligence.*, *Commun. ACM* **58** (2015), no. 9 92–103.

[18] B. Y. Lin, X. Chen, J. Chen, and X. Ren, *Kagnet: Knowledge-aware graph networks for commonsense reasoning*, *arXiv preprint arXiv:1909.02151* (2019).

[19] R. Speer, J. Chin, and C. Havasi, *ConceptNet 5.5: An Open Multilingual Graph of General Knowledge*, in *Proceedings of the AAAI*, 2017.

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

[21] Z. Li, W. Chen, S. Li, H. Wang, J. Qian, and X. Yan, *Dialogic: Controllable dialogue simulation with in-context learning*, *arXiv preprint arXiv:2210.04185* (2022).

[22] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, *Language models are few-shot learners*, in *Advances in Neural Information Processing Systems* (H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, eds.), vol. 33, pp. 1877–1901, Curran Associates, Inc., 2020.

[23] R. Thoppilan, D. D. Freitas, J. Hall, N. M. Shazeer, A. Kulshreshtha, H.-T. Cheng, A. Jin, T. Bos, L. Baker, Y. Du, Y. Li, H. Lee, H. Zheng, A. Ghafouri, M. Menegali, Y. Huang, M. Krikun, D. Lepikhin, J. Qin, D. Chen, Y. Xu, Z. Chen, A. Roberts, M. Bosma, Y. Zhou, C.-C. Chang, I. A. Krivokon, W. J. Rusch, M. Pickett, K. S. Meier-Hellstern, M. R. Morris, T. Doshi, R. D. Santos, T. Duke, J. H. Søraker, B. Zevenbergen, V. Prabhakaran, M. Diaz, B. Hutchinson, K. Olson, A. Molina, E. Hoffman-John, J. Lee, L. Aroyo, R. Rajakumar, A. Butryna, M. Lamm, V. O. Kuzmina, J. Fenton, A. Cohen, R. Bernstein, R. Kurzweil, B. Aguera-Arcas, C. Cui, M. Croak, E. Chi, and Q. Le, *Lamda: Language models for dialog applications*, *ArXiv* **abs/2201.08239** (2022).

[24] A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H. W. Chung, C. Sutton, S. Gehrmann, P. Schuh, K. Shi, S. Tsvyashchenko, J. Maynez, A. B. Rao, P. Barnes, Y. Tay, N. M. Shazeer, V. Prabhakaran, E. Reif, N. Du, B. C. Hutchinson, R. Pope, J. Bradbury, J. Austin, M. Isard, G. Gur-Ari, P. Yin, T. Duke, A. Levskaya, S. Ghemawat, S. Dev, H. Michalewski, X. García, V. Misra, K. Robinson, L. Fedus, D. Zhou, D. Ippolito, D. Luan, H. Lim, B. Zoph, A. Spiridonov, R. Sepassi, D. Dohan, S. Agrawal, M. Omernick, A. M. Dai, T. S. Pillai, M. Pellat, A. Lewkowycz, E. O. Moreira, R. Child, O. Polozov, K. Lee, Z. Zhou, X. Wang, B. Saeta, M. Diaz, O. Firat, M. Catasta, J. Wei, K. S. Meier-Hellstern, D. Eck, J. Dean, S. Petrov, and N. Fiedel, *Palm: Scaling language modeling with pathways*, *ArXiv* **abs/2204.02311** (2022).

[25] J. Wei, M. Bosma, V. Zhao, K. Guu, A. W. Yu, B. Lester, N. Du, A. M. Dai, and Q. V. Le, *Finetuned language models are zero-shot learners*, in *International Conference on Learning Representations*, 2022.

[26] V. Sanh, A. Webson, C. Raffel, S. Bach, L. Sutawika, Z. Alyafeai, A. Chaffin, A. Stiegler, A. Raja, M. Dey, M. S. Bari, C. Xu, U. Thakker, S. S. Sharma, E. Szczechla, T. Kim, G. Chhablani, N. Nayak, D. Datta, J. Chang, M. T.-J. Jiang, H. Wang, M. Manica, S. Shen, Z. X. Yong, H. Pandey, R. Bawden, T. Wang, T. Neeraj, J. Rozen, A. Sharma, A. Santilli, T. Fevry, J. A. Fries, R. Teehan, T. L. Scao, S. Biderman, L. Gao, T. Wolf, and A. M. Rush, *Multitask prompted training enables zero-shot task generalization*, in *International Conference on Learning Representations*, 2022.

[27] F. Shi, M. Suzgun, M. Freitag, X. Wang, S. Srivats, S. Vosoughi, H. W. Chung, Y. Tay, S. Ruder, D. Zhou, D. Das, and J. Wei, *Language models are multilingual chain-of-thought reasoners*, *ArXiv* **abs/2210.03057** (2022).

[28] J. Wei, X. Wang, D. Schuurmans, M. Bosma, E. Chi, Q. Le, and D. Zhou, *Chain of thought prompting elicits reasoning in large language models*, *ArXiv* **abs/2201.11903** (2022).

[29] J. Wei, Y. Tay, R. Bommasani, C. Raffel, B. Zoph, S. Borgeaud, D. Yogatama, M. Bosma, D. Zhou, D. Metzler, E. H. Chi, T. Hashimoto, O. Vinyals, P. Liang, J. Dean, and W. Fedus, *Emergent abilities of large language models*, *Transactions on Machine Learning Research* (2022). Survey Certification.

[30] E. Zelikman, Y. Wu, and N. D. Goodman, *Star: Bootstrapping reasoning with reasoning*, *ArXiv* **abs/2203.14465** (2022).

[31] S. Li, J. Chen, and D. Yu, *Teaching pretrained models with commonsense reasoning: A preliminary kb-based approach*, *ArXiv* **abs/1909.09743** (2019).

[32] S. Wiegreffe, J. Hessel, S. Swayamdipta, M. O. Riedl, and Y. Choi, *Reframing human-ai collaboration for generating free-text explanations*, *ArXiv* **abs/2112.08674** (2021).

[33] S. Li, S. Yavuz, W. Chen, and X. Yan, *Task-adaptive pre-training and self-training are complementary for natural language understanding*, in *Findings of the Association for Computational Linguistics: EMNLP 2021*, (Punta Cana, Dominican Republic), pp. 1006–1015, Association for Computational Linguistics, Nov., 2021.

[34] S. Li, S. Yavuz, K. Hashimoto, J. Li, T. Niu, N. Rajani, X. Yan, Y. Zhou, and C. Xiong, *Coco: Controllable counterfactuals for evaluating dialogue state trackers*, in *International Conference on Learning Representations*, 2021.

[35] S. LI, J. Chen, Y. Shen, Z. Chen, X. Zhang, Z. Li, H. Wang, J. Qian, B. Peng, Y. Mao, W. Chen, and X. Yan, *Explanations from large language models make small reasoners better*, *ArXiv* **abs/2210.06726** (2022).

[36] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning*, vol. 1. MIT Press, 2016.

[37] Q. Xie, M.-T. Luong, E. Hovy, and Q. V. Le, *Self-training with noisy student improves imagenet classification*, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June, 2020.

[38] J. He, J. Gu, J. Shen, and M. Ranzato, *Revisiting self-training for neural sequence generation*, in *International Conference on Learning Representations*, 2020.

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

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

[41] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman, *GLUE: A multi-task benchmark and analysis platform for natural language understanding*, in *International Conference on Learning Representations*, 2019.

[42] T. Chen, S. Kornblith, K. Swersky, M. Norouzi, and G. E. Hinton, *Big self-supervised models are strong semi-supervised learners*, in *Advances in Neural Information Processing Systems* (H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, eds.), vol. 33, pp. 22243–22255, Curran Associates, Inc., 2020.

[43] B. Zoph, G. Ghiasi, T.-Y. Lin, Y. Cui, H. Liu, E. D. Cubuk, and Q. Le, *Rethinking pre-training and self-training*, in *NeurIPS*, 2020.

[44] J. Kahn, A. Lee, and A. Hannun, *Self-training for end-to-end speech recognition*, in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 7084–7088, 2020.

[45] J. Du, E. Grave, B. Gunel, V. Chaudhary, O. Celebi, M. Auli, V. Stoyanov, and A. Conneau, *Self-training improves pre-training for natural language understanding*, in *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, (Online), pp. 5408–5418, Association for Computational Linguistics, June, 2021.

[46] S. Mukherjee and A. H. Awadallah, *Uncertainty-aware self-training for text classification with few labels*, *ArXiv* **abs/2006.15315** (2020).

[47] J. Lee, W. Yoon, S. Kim, D. Kim, S. Kim, C. H. So, and J. Kang, *Biobert: a pre-trained biomedical language representation model for biomedical text mining*, *Bioinformatics* **36** (2020) 1234 – 1240.

[48] Q. Sun, X. Li, Y. Liu, S. Zheng, T.-S. Chua, and B. Schiele, *Learning to self-train for semi-supervised few-shot classification*, in *NeurIPS*, 2019.

[49] C. Wei, K. Shen, Y. Chen, and T. Ma, *Theoretical analysis of self-training with deep networks on unlabeled data*, in *International Conference on Learning Representations*, 2021.

[50] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, *A simple framework for contrastive learning of visual representations*, in *Proceedings of the 37th International Conference on Machine Learning* (H. D. III and A. Singh, eds.), vol. 119 of *Proceedings of Machine Learning Research*, pp. 1597–1607, PMLR, 13–18 Jul, 2020.

[51] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. S. Bernstein, A. Berg, and L. Fei-Fei, *Imagenet large scale visual recognition challenge*, *International Journal of Computer Vision* **115** (2015) 211–252.

[52] G. E. Hinton, O. Vinyals, and J. Dean, *Distilling the knowledge in a neural network*, *ArXiv* **abs/1503.02531** (2015).

[53] Q. Xu, A. Baevski, T. Likhomanenko, P. Tomasello, A. Conneau, R. Collobert, G. Synnaeve, and M. Auli, *Self-training and pre-training are complementary for speech recognition*, *ArXiv* **abs/2010.11430** (2020).

[54] X. Zhu and A. B. Goldberg, *Introduction to Semi-Supervised Learning*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2009.

[55] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Ng, and C. Potts, *Recursive deep models for semantic compositionality over a sentiment treebank*, in *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, (Seattle, Washington, USA), pp. 1631–1642, Association for Computational Linguistics, Oct., 2013.

[56] A. Williams, N. Nangia, and S. Bowman, *A broad-coverage challenge corpus for sentence understanding through inference*, in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, (New Orleans, Louisiana), pp. 1112–1122, Association for Computational Linguistics, June, 2018.

[57] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, *SQuAD: 100,000+ questions for machine comprehension of text*, in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, (Austin, Texas), pp. 2383–2392, Association for Computational Linguistics, Nov., 2016.

[58] Z. Chen, H. Zhang, X. Zhang, and L. Zhao, *Quora question pairs*, 2018.

[59] E. F. Tjong Kim Sang and F. De Meulder, *Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition*, in *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pp. 142–147, 2003.

[60] M. Eric, R. Goel, S. Paul, A. Sethi, S. Agarwal, S. Gao, A. Kumar, A. Goyal, P. Ku, and D. Hakkani-Tur, *MultiWOZ 2.1: A consolidated multi-domain dialogue dataset with state corrections and state tracking baselines*, in *Proceedings of the 12th Language Resources and Evaluation Conference*, (Marseille, France), pp. 422–428, European Language Resources Association, May, 2020.

[61] Y. Liu, F. He, H. Zhang, G. Rao, Z. Feng, and Y. Zhou, *How well do machines perform on iq tests: a comparison study on a large-scale dataset*, *Proceedings of the IJCAI* (2019).

[62] S. Ostermann, M. Roth, A. Modi, S. Thater, and M. Pinkal, *SemEval-2018 Task 11: Machine comprehension using commonsense knowledge*, in *Proceedings of the SemEval*, 2018.

[63] T. H. Trinh and Q. V. Le, *A simple method for commonsense reasoning*, *arXiv preprint arXiv:1806.02847* (2018).

[64] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor, *Freebase: a collaboratively created graph database for structuring human knowledge*, in *Proceedings of the ACM SIGMOD*, 2008.

[65] Y. Li and P. Clark, *Answering elementary science questions by constructing coherent scenes using background knowledge*, in *Proceedings of the EMNLP*, 2015.

[66] B. Bi, C. Wu, M. Yan, W. Wang, J. Xia, and C. Li, *Incorporating external knowledge into machine reading for generative question answering*, in *Proceedings of the EMNLP*, 2019.

[67] L. Laugier, A. Wang, C.-S. Foo, T. Guenais, and V. Chandrasekhar, *Encoding knowledge graph with graph cnn for question answering*, *Proceedings of the ICLR* (2019).

[68] T. Kipf and M. Welling, *Semi-supervised classification with graph convolutional networks*, *ArXiv* **abs/1609.02907** (2016).

[69] K. Sun, D. Yu, J. Chen, D. Yu, Y. Choi, and C. Cardie, *DREAM: A challenge data set and models for dialogue-based reading comprehension*, *Transactions of the Association for Computational Linguistics* (2019).

[70] L. Wang, M. Sun, W. Zhao, K. Shen, and J. Liu, *Yuanfudao at SemEval-2018 Task 11: Three-way attention and relational knowledge for commonsense machine comprehension*, in *Proceedings of the SemEval*, 2018.

[71] W. Zhong, D. Tang, N. Duan, M. Zhou, J. Wang, and J. Yin, *Improving question answering by commonsense-based pre-training*, *arXiv preprint arXiv:1809.03568* (2018).

[72] K. Sun, D. Yu, D. Yu, and C. Cardie, *Probing prior knowledge needed in challenging chinese machine reading comprehension*, *arXiv preprint arXiv:1904.09679* (2019).

[73] Z.-X. Ye, Q. Chen, W. Wang, and Z.-H. Ling, *Align, mask and select: A simple method for incorporating commonsense knowledge into language representation models*, *arXiv* (2019).

[74] L. Huang, R. L. Bras, C. Bhagavatula, and Y. Choi, *Cosmos qa: Machine reading comprehension with contextual commonsense reasoning*, in *Proceedings of the EMNLP*, 2019.

[75] Y. Zhang, Z. Ou, and Z. Yu, *Task-oriented dialog systems that consider multiple appropriate responses under the same context*, 2019.

[76] A. Neelakantan, S. Yavuz, S. Narang, V. Prasad, B. Goodrich, D. Duckworth, C. Sankar, and X. Yan, *Neural assistant: Joint action prediction, response generation, and latent knowledge reasoning*, in *NeurIPS 2019 Converstional AI Workshop*, 2019.

[77] B. Peng, C. Li, J. Li, S. Shayandeh, L. Liden, and J. Gao, *Soloist: Few-shot task-oriented dialog with a single pre-trained auto-regressive model*, 2020.

[78] K. Wang, J.-F. Tian, R. Wang, X. Quan, and J. Yu, *Multi-domain dialogue acts and response co-generation*, in *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2020.

[79] J. Wang, Y. Zhang, T.-K. Kim, and Y. Gu, *Modelling hierarchical structure between dialogue policy and natural language generator with option framework for task-oriented dialogue system*, *ArXiv* **abs/2006.06814** (2020).

[80] P. Budzianowski, T.-H. Wen, B.-H. Tseng, I. Casanueva, S. Ultes, O. Ramadan, and M. Gašić, *MultiWOZ - a large-scale multi-domain wizard-of-Oz dataset for task-oriented dialogue modelling*, in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018.

[81] B. Byrne, K. Krishnamoorthi, C. Sankar, A. Neelakantan, B. Goodrich, D. Duckworth, S. Yavuz, A. Dubey, K.-Y. Kim, and A. Cedilnik, *Taskmaster-1: Toward a realistic and diverse dialog dataset*, in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019.

[82] A. Rastogi, X. Zang, S. Sunkara, R. Gupta, and P. Khaitan, *Towards scalable multi-domain conversational agents: The schema-guided dialogue dataset*, *arXiv preprint arXiv:1909.05855* (2019).

[83] B. Liu and I. Lane, *End-to-end learning of task-oriented dialogs*, in *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2018.

[84] J. Pennington, R. Socher, and C. D. Manning, *Glove: Global vectors for word representation*, in *Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532–1543, 2014.

[85] M. T. Ribeiro, T. Wu, C. Guestrin, and S. Singh, *Beyond accuracy: Behavioral testing of NLP models with CheckList*, in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Association for Computational Linguistics, 2020.

[86] K. Patel, J. Fogarty, J. A. Landay, and B. Harrison, *Investigating statistical machine learning as a tool for software development*, in *CHI*, 2008.

[87] C.-S. Wu, A. Madotto, E. Hosseini-Asl, C. Xiong, R. Socher, and P. Fung, *Transferable multi-domain state generator for task-oriented dialogue systems*, in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Association for Computational Linguistics, 2019.

[88] M. Heck, C. van Niekerk, N. Lubis, C. Geishauser, H.-C. Lin, M. Moresi, and M. Gasic, *TripPy: A triple copy strategy for value independent neural dialog state tracking*, in *Proceedings of the 21th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, (1st virtual meeting), pp. 35–44, Association for Computational Linguistics, July, 2020.

[89] E. Hosseini-Asl, B. McCann, C.-S. Wu, S. Yavuz, and R. Socher, *A simple language model for task-oriented dialogue*, 2020.

[90] M. Henderson, B. Thomson, and S. Young, *Word-based dialog state tracking with recurrent neural networks*, in *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, (Philadelphia, PA, U.S.A.), pp. 292–299, Association for Computational Linguistics, June, 2014.

[91] T.-H. Wen, D. Vandyke, N. Mrkšić, M. Gašić, L. M. Rojas-Barahona, P.-H. Su, S. Ultes, and S. Young, *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*, (Valencia, Spain), pp. 438–449, Association for Computational Linguistics, Apr., 2017.

[92] S. Gao, A. Sethi, S. Agarwal, T. Chung, and D. Z. Hakkani-Tür, *Dialog state tracking: A neural reading comprehension approach*, *ArXiv* **abs/1908.01946** (2019).

[93] J. Zhang, K. Hashimoto, C.-S. Wu, Y. Wan, P. S. Yu, R. Socher, and C. Xiong, *Find or classify? dual strategy for slot-value predictions on multi-domain dialog state tracking*, *ArXiv* **abs/1910.03544** (2019).

[94] A. See, P. J. Liu, and C. D. Manning, *Get to the point: Summarization with pointer-generator networks*, in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, (Vancouver, Canada), pp. 1073–1083, Association for Computational Linguistics, July, 2017.

[95] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, *Intriguing properties of neural networks*, in *International Conference on Learning Representations*, 2014.

[96] I. Goodfellow, J. Shlens, and C. Szegedy, *Explaining and harnessing adversarial examples*, in *International Conference on Learning Representations*, 2015.

[97] N. Papernot, P. McDaniel, A. Swami, and R. E. Harang, *Crafting adversarial input sequences for recurrent neural networks*, *MILCOM 2016 - 2016 IEEE Military Communications Conference* (2016) 49–54.

[98] M. Alzantot, Y. Sharma, A. Elgohary, B.-J. Ho, M. Srivastava, and K.-W. Chang, *Generating natural language adversarial examples*, in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, (Brussels, Belgium), pp. 2890–2896, Association for Computational Linguistics, Oct.-Nov., 2018.

[99] D. Jin, Z. Jin, J. T. Zhou, and P. Szolovits, *Is bert really robust? natural language attack on text classification and entailment*, *ArXiv* **abs/1907.11932** (2019).

[100] M. Iyyer, J. Wieting, K. Gimpel, and L. Zettlemoyer, *Adversarial example generation with syntactically controlled paraphrase networks*, in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, (New Orleans, Louisiana), pp. 1875–1885, Association for Computational Linguistics, June, 2018.

[101] R. Jia and P. Liang, *Adversarial examples for evaluating reading comprehension systems*, in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, (Copenhagen, Denmark), pp. 2021–2031, Association for Computational Linguistics, Sept., 2017.

[102] W. Y. Wang and D. Yang, *That's so annoying!!!: A lexical and frame-semantic embedding based data augmentation approach to automatic categorization of annoying behaviors using #petpeeve tweets*, in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, (Lisbon, Portugal), pp. 2557–2563, Association for Computational Linguistics, Sept., 2015.

[103] J. Wei and K. Zou, *EDA: Easy data augmentation techniques for boosting performance on text classification tasks*, in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, (Hong Kong, China), pp. 6382–6388, Association for Computational Linguistics, Nov., 2019.

[104] H. Guo, Y. Mao, and R. Zhang, *Augmenting data with mixup for sentence classification: An empirical study*, *ArXiv* **abs/1905.08941** (2019).

[105] H. Zhang, M. Cissé, Y. Dauphin, and D. Lopez-Paz, *mixup: Beyond empirical risk minimization*, *ArXiv* **abs/1710.09412** (2017).

[106] X. Wu, S. Lv, L. Zang, J. Han, and S. Hu, *Conditional bert contextual augmentation*, *ArXiv* **abs/1812.06705** (2018).

[107] A. Anaby-Tavor, B. Carmeli, E. Goldbraich, A. Kantor, G. Kour, S. Shlomov, N. Tepper, and N. Zwerdling, *Not enough data? deep learning to the rescue!*, *ArXiv* **abs/1911.03118** (2019).

[108] A. Einolghozati, S. Gupta, M. Mohit, and R. Shah, *Improving robustness of task oriented dialog systems*, *ArXiv* **abs/1911.05153** (2019).

[109] M. Cheng, W. Wei, and C.-J. Hsieh, *Evaluating and enhancing the robustness of dialogue systems: A case study on a negotiation agent*, 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)*, (Minneapolis, Minnesota), pp. 3325–3335, Association for Computational Linguistics, June, 2019.

[110] M. Lewis, D. Yarats, Y. Dauphin, D. Parikh, and D. Batra, *Deal or no deal? end-to-end learning of negotiation dialogues*, in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, (Copenhagen, Denmark), pp. 2443–2453, Association for Computational Linguistics, Sept., 2017.

[111] A. W. Yu, D. Dohan, M.-T. Luong, R. Zhao, K. Chen, M. Norouzi, and Q. V. Le, *QANet: Combining Local Convolution with Global Self-Attention for Reading Comprehension*, in *6th International Conference on Learning Representations (ICLR)*, 2018.

[112] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin, *A neural probabilistic language model*, *J. Mach. Learn. Res.* **3** (Mar., 2003) 1137–1155.

[113] G. Chao and I. Lane, *Bert-dst: Scalable end-to-end dialogue state tracking with bidirectional encoder representations from transformer*, in *INTERSPEECH*, 2019.

[114] A. Kumar, P. Ku, A. K. Goyal, A. Metallinou, and D. Z. Hakkani-Tür, *Ma-dst: Multi-attention based scalable dialog state tracking*, *ArXiv* **abs/2002.08898** (2020).

[115] H. T. Le, R. Socher, and S. Hoi, *Non-autoregressive dialog state tracking*, *ArXiv* **abs/2002.08024** (2020).

[116] L. Chen, B. Lv, C. Wang, S. Zhu, B. Tan, and K. Yu, *Schema-guided multi-domain dialogue state tracking with graph attention neural networks*, *Proceedings of the AAAI Conference on Artificial Intelligence* **34** (Apr., 2020) 7521–7528.

[117] Z. Lin, A. Madotto, G. I. Winata, and P. Fung, *Mintl: Minimalist transfer learning for task-oriented dialogue systems*, *ArXiv* **abs/2009.12005** (2020).

[118] S. Mehri, M. Eric, and D. Hakkani-Tur, *Dialoglue: A natural language understanding benchmark for task-oriented dialogue*, *ArXiv* **abs/2009.13570** (2020).

[119] A. K. Lampinen, I. Dasgupta, S. C. Y. Chan, K. Matthewson, M. H. Tessler, A. Creswell, J. L. McClelland, J. X. Wang, and F. Hill, *Can language models learn from explanations in context?*, *ArXiv* **abs/2204.02329** (2022).

[120] K. Cobbe, V. Kosaraju, M. Bavarian, J. Hilton, R. Nakano, C. Hesse, and J. Schulman, *Training verifiers to solve math word problems*, *ArXiv* **abs/2110.14168** (2021).

[121] D. Zhou, N. Scharli, L. Hou, J. Wei, N. Scales, X. Wang, D. Schuurmans, O. Bousquet, Q. Le, and E. Chi, *Least-to-most prompting enables complex reasoning in large language models*, *ArXiv* **abs/2205.10625** (2022).

[122] P. Hase, S. Zhang, H. Xie, and M. Bansal, *Leakage-adjusted simulatability: Can models generate non-trivial explanations of their behavior in natural language?*, in *EMNLP (Findings)*, pp. 4351–4367, 2020.

[123] S. Wiegreffe and A. Marasović, *Teach me to explain: A review of datasets for explainable natural language processing*, in *NeurIPS Datasets and Benchmarks*, 2021.

[124] S. Narang, C. Raffel, K. Lee, A. Roberts, N. Fiedel, and K. Malkan, *Wt5?! training text-to-text models to explain their predictions*, *ArXiv* **abs/2004.14546** (2020).

[125] N. F. Rajani, B. McCann, C. Xiong, and R. Socher, *Explain yourself! leveraging language models for commonsense reasoning*, in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, (Florence, Italy), pp. 4932–4942, Association for Computational Linguistics, July, 2019.

[126] W. Samek, G. Montavon, A. Vedaldi, L. K. Hansen, and K.-R. Müller, *Explainable AI: interpreting, explaining and visualizing deep learning*, vol. 11700. Springer Nature, 2019.

[127] M. Geva, D. Khashabi, E. Segal, T. Khot, D. Roth, and J. Berant, *Did aristotle use a laptop? a question answering benchmark with implicit reasoning strategies*, *Transactions of the Association for Computational Linguistics* **9** (2021) 346–361.

[128] T. Kojima, S. S. Gu, M. Reid, Y. Matsuo, and Y. Iwasawa, *Large language models are zero-shot reasoners*, *ArXiv* **abs/2205.11916** (2022).

[129] X. Ye and G. Durrett, *The unreliability of explanations in few-shot in-context learning*, *ArXiv* **abs/2205.03401** (2022).

[130] B. Wang, "Mesh-Transformer-JAX: Model-Parallel Implementation of Transformer Language Model with JAX." `https://github.com/kingoflolz/mesh-transformer-jax`, May, 2021.

[131] W. L. Johnson, *Agents that learn to explain themselves*, *AAAI* (05, 1994).

[132] L. A. Hendricks, Z. Akata, M. Rohrbach, J. Donahue, B. Schiele, and T. Darrell, *Generating visual explanations*, in *ECCV*, 2016.

[133] O.-M. Camburu, T. Rocktäschel, T. Lukasiewicz, and P. Blunsom, *e-snli: Natural language inference with natural language explanations*, in *NeurIPS*, 2018.

[134] S. Jain, S. Wiegreffe, Y. Pinter, and B. C. Wallace, *Learning to faithfully rationalize by construction*, in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, (Online), pp. 4459–4473, Association for Computational Linguistics, July, 2020.

[135] S. Hochreiter and J. Schmidhuber, *Long short-term memory*, *Neural computation* **9** (1997), no. 8 1735–1780.

[136] T. Mihaylov, P. Clark, T. Khot, and A. Sabharwal, *Can a suit of armor conduct electricity? a new dataset for open book question answering*, in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, (Brussels, Belgium), pp. 2381–2391, Association for Computational Linguistics, Oct.-Nov., 2018.

[137] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. Le Scao, S. Gugger, M. Drame, Q. Lhoest, and A. Rush, *Transformers: State-of-the-art natural language processing*, in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, (Online), pp. 38–45, Association for Computational Linguistics, Oct., 2020.

[138] P. W. Koh and P. Liang, *Understanding black-box predictions via influence functions*, *ArXiv* **abs/1703.04730** (2017).

[139] M.-A. Clinciu, A. Eshghi, and H. Hastie, *A study of automatic metrics for the evaluation of natural language explanations*, in *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, (Online), pp. 2376–2387, Association for Computational Linguistics, Apr., 2021.

[140] M. Kayser, O.-M. Camburu, L. Salewski, C. Emde, V. Do, Z. Akata, and T. Lukasiewicz, *e-vil: A dataset and benchmark for natural language explanations in vision-language tasks*, *2021 IEEE/CVF International Conference on Computer Vision (ICCV)* (2021) 1224–1234.

[141] J. Shin, H. Yu, H. Moon, A. Madotto, and J. Park, *Dialogue summaries as dialogue states (DS2), template-guided summarization for few-shot dialogue state tracking*, in *Findings of the Association for Computational Linguistics: ACL 2022*, (Dublin, Ireland), pp. 3824–3846, Association for Computational Linguistics, May, 2022.

[142] Y. Pruksachatkun, J. Phang, H. Liu, P. M. Htut, X. Zhang, R. Y. Pang, C. Vania, K. Kann, and S. R. Bowman, *Intermediate-task transfer learning with pretrained language models: When and why does it work?*, in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, (Online), pp. 5231–5247, Association for Computational Linguistics, July, 2020.

[143] T. Gao, A. Fisch, and D. Chen, *Making pre-trained language models better few-shot learners*, in *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, (Online), pp. 3816–3830, Association for Computational Linguistics, Aug., 2021.

[144] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, *Pytorch: An imperative style, high-performance deep learning library*, in *Neural Information Processing Systems*, 2019.

[145] X. Zang, A. Rastogi, S. Sunkara, R. Gupta, J. Zhang, and J. Chen, *MultiWOZ 2.2 : A dialogue dataset with additional annotation corrections and state tracking baselines*, in *Proceedings of the 2nd Workshop on Natural Language Processing for Conversational AI*, (Online), pp. 109–117, Association for Computational Linguistics, July, 2020.

[146] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, 2015.

[147] J. Li, M. Galley, C. Brockett, J. Gao, and W. Dolan, *A diversity-promoting objective function for neural conversation models*, *ArXiv* **abs/1510.03055** (2016).