Enhancing Transformer Models for Dialogue Summarization

Bo-Ru Lu

A dissertation

submitted in partial fulfillment of the

requirements for the degree of

Doctor of Philosophy

University of Washington
2024

Reading Committee:
Mari Ostendorf, Chair
Noah A. Smith
Meliha Yetisgen
Hao Cheng

Program Authorized to Offer Degree:
Electrical and Computer Engineering

University of Washington

Abstract

Enhancing Transformer Models for Dialogue Summarization

Bo-Ru Lu

Chair of the Supervisory Committee:

Mari Ostendorf
Electrical and Computer Engineering

Understanding and generating summaries of conversational speech and text are crucial for various applications such as virtual assistants, customer service calls, sales calls, and doctor-patient consultations. Transcribed human-human dialogues often lack explicit structure, making them time-consuming to read and challenging to skim for essential information. This thesis explores the use of natural language processing (NLP) techniques to automatically summarize two-party conversations, improving the efficiency of understanding large volumes of dialogue data. We address the limitations of existing summarization methods by proposing innovative approaches aimed at improving performance and reducing the cost of transformer models.

We introduce two primary types of summarization tasks: corpus-level and conversation-level. Corpus-level summarization aims to provide an overview of speaker behaviors across multiple dialogues by using a graph-based approach, which simplifies the understanding and summarizing of large dialogue corpora. This method highlights common and distinct subdialogues, and can potentially aid in the refinement of agent training and the enhancement of call center analytics. On the other hand, conversation-level summarization focuses on individual dialogues, producing structured summaries that capture essential details for follow-up interactions. These summaries are particularly useful in domains such as customer service and healthcare, where accurate and efficient information extraction is critical.

To address the challenges of summarizing large and complex dialogues, we propose new modeling algo-

rithms and a data collection framework. We enhance transformer models by incorporating additional structured information derived from dialogue graphs and expert-designed schemas, improving both performance and efficiency. Furthermore, we demonstrate that smaller encoder-decoder models can outperform larger decoder-only models in specialized domains, offering faster inference speeds and comparable performance. Our human-language model collaborative data synthesis framework also increases annotation efficiency and reduces costs, particularly for proprietary data. Through these contributions, this thesis advances the field of conversational AI, providing more effective and efficient methods for summarizing dialogues.

Acknowledgements

I want to sincerely thank my advisor, Mari Ostendorf, for her invaluable guidance and insight throughout my PhD studies. Her influence is crucial in helping me think critically and express my ideas clearly. Her constant support and patience allowed me to explore various research paths, making this journey possible.

I am also deeply grateful to Noah A. Smith for his continuous support and valuable research guidance over the past few years. Special thanks to Hao Cheng for meeting with me weekly throughout my entire PhD, brainstorming research ideas, and offering guidance. I am fortunate to have Meliha Yetisgen and Shane Steinert-Threlkeld on my thesis committee. I appreciate their constructive feedback and valuable suggestions.

I also want to express my gratitude to my advisors at National Taiwan University, Yun-Nung Chen and Hung-Yi Lee, for their valuable guidance when I was just starting as a young researcher. My deepest thanks go to Lin-Shan Lee, my inspiring research advisor, for introducing me to this fascinating field. I still remember our dinner at a restaurant on Wenzhou St. in February 2015, where he encouraged me to join his research lab to open my research journey.

Being part of the TIAL Lab and UWNLP is a privilege. I extend my gratitude to all the lab members and alumni: Aaron Jaech, Farah Nadeem, Hao Feng, Junkai Wu, Kevin Everson, Kevin Lybarger, Sitong Zhou, Sara Ng, Trang Tran, Vicky Zayats, and Yi Luan, with whom I have met and collaborate over the past six years. I want to thank my research collaborators, Chien-Yu Lin, Nikita Haduong, Yushi Hu, and Zeqiu Wu (listed alphabetically), whose significant contributions to some of the work in this thesis. I also appreciate the discussion from UWNLP members, including Jungo Kasai, Tao Yu, and Tim Dettmers, who help me throughout this degree.

Beyond research, I am fortunate to have a group of friends in Seattle who provided companionship and

fun. I especially want to thank the friends I met during this PhD journey in Seattle: Alvin Ting, Ang-Ching Chen, Aimee Wu, Bing-Syuan Wang, Chia-Ning Lee, Daphne Chiu, Jing Chao, Man-Lin Chao, Michelle Liu, Ming-Chien Chiang, Pomi Chang, Po-Han Chen, Serena Lin, Shayne Mei, Thomas Wang, Vanessa Chu, Webb Tseng, and Yujung Lu for their consistent care and attention.

I also want to thank my friends from our days at National Taiwan University, Frank Shyu, Wen-Jen Lee and Yu-An Lin, who have given me the confidence and support to believe I can overcome any challenge. Additionally, my friends from junior high and high school—Bo-Yi Wu, Kun-Yan Wu, Ting-Wei Kuo, Yu-An Li, and Wei-Lun Yu—have always been there to encourage me, offering a listening ear and lifting my spirits whenever I felt down. I am also grateful to those who are not mentioned here but whose support has been equally meaningful to me.

I am deeply thankful to my loving family—my mother, Chia-Yen Lee; my father, Yi-Chung Lu; and my two younger sisters, Ting-Xuan Lu and Ting-Yu Lu—for their love and support. They taught me the importance of gratitude, resilience, and humility.

Finally, I want to express my deepest gratitude to my girlfriend, Chih-Ning Ho. Her support, encouragement, and understanding have been my constant source of strength throughout this entire PhD journey. She always be by my side, always believing in me when I found it difficult to believe in myself. Her love and companionship have been essential in helping me navigate the challenges of this journey. I am excited for what the future holds for us, and I look forward to continuing to share life's adventures together.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

A wide variety of applications for language understanding and generation involve conversational speech and text, including human-machine dialogue systems such as virtual assistants [13] and question-answering agents [114], and human-human dialogues such as customer service calls, sales calls, doctor-patient consultation [121]. Transcribed human-human spoken dialogues can be time-consuming to read because there is less explicitly structure text compared to written documents, making it harder for humans to skim and to capture important information from them. Therefore, there is interest in automatically summarizing conversations using natural language processing (NLP).

Previous research in summarization has predominantly concentrated on abstractive [62, 36, 66], extractive [129, 71], and information extraction (e.g., form-filling extraction) approaches for single or multiple documents or dialogues. Abstractive summarization generates new sentences that represent the core ideas of the original text, while extractive summarization selects and combines key sentences or phrases directly from the source text to create a summary. Form-filling information extraction extracts key information based on an expert-designed schema, usually presented as short phrases. In this thesis, we diverge from prior categories by exploring summarization tasks from different perspectives, specifically focusing on corpus-level summarization and conversation-level summarization. Corpus-level summarization aims to produce a set of summaries that describe different kinds of dialogues in an entire dialogue corpus, offering a comprehensive overview of speaker behaviors across various conversations within the corpus. On the other hand, conversation-level summarization targets individual dialogues, making it particularly useful for extracting

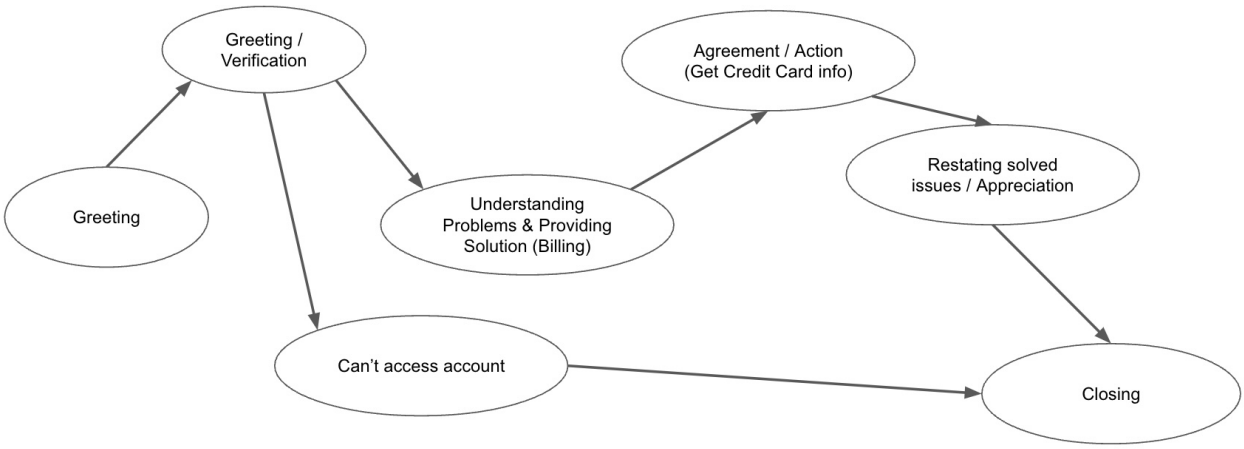


Figure 1.1: An example of simplified graph for corpus-level summarization.

concise notes for follow-up interactions or decisions. The extracted information can be presented as either a short phrase extracted from the dialogues for particular slots, or a full sentence or paragraph description for a specific section. These summarization tasks aim to address the impracticality of reading through large volumes of spontaneous conversations, making it a critical topic in NLP. We discuss more details of these two types of summarization tasks in the following two paragraphs.

Corpus-level summarization aims to condense information from a large volume of dialogues to understand the underlying speaker behavior via a summary graph. By summarizing this information into a graph, it can support call center analytics, which can refine agent training, enhance service quality, and improve automatic systems. Ultimately, this leads to increased customer satisfaction and reduced costs through improved efficiency, all without the need to read through numerous conversations. To present the corpus-level summary, we can view a customer service call as a sequence of subtask-oriented subdialogues derived from a summary graph. Each subdialogue consists of a series of speaker turns that collectively fulfill a functional purpose, including opening/greeting, user identification, problem description, problem-solving, issue restatement, and closing/thank you. Motivated by the desire for simplicity and ease in distinguishing between calls, we use a graph to summarize a corpus of dialogues. We use Figure 1.1 to illustrate a simplified graph derived from our private call center dataset (CALL CENTER) for corpus-level summarization. In this summary graph, each node represents a subdialogue, and each edge represents a transition between subdialogues. This graph illustrates the shared (e.g., greeting, verification, and closing) and distinct (e.g., understanding a problem, problem-solving solutions for billing issues, credit card information gathering, and account access

issues) underlying subdialogues between different calls within the corpus. The advantage of using a graph is that paths can share common prefixes, indicating that the involved speakers have the same subgoal in two different types of conversations. Afterward, these paths may diverge at certain points and merge back into another subdialogue (e.g., question/solution restatement or closing). The primary differences between the two types of calls can act as key indicators and be used for call analytics. For example, in a customer service call, an agent and a user typically greet each other followed by user verification. After this, the user will describe the specific problem they are experiencing. At this point, the conversation diverges based on the individual issues raised. Once these issues are resolved, the conversation converges again in the closing stage, where the agent and user exchange farewells and thanks.

On the other hand, *conversation-level summarization* focuses on providing a concise and structured summary for a single conversation. The structured summary is particularly useful for "information-gathering" dialogues. The predefined structure (e.g., a schema for auto insurance claim calls or for clinical notes in doctor-patient dialogues) enables the summaries to capture the desired information from a conversation for a follow-up interactions or actions. For example, in a car accident claim call, an agent must collect necessary details (e.g., location, vehicle make/model, traffic condition) for the claim and assist a caller in articulating the circumstances of the accident to determine liability. Typically, this extracted information adheres to a structured schema defined by in-domain experts and can be presented as a concise tabular summary of an accident. An adjuster can review the form to determine liability in accidents without going through everything from the entire conversation. In a medical scenario, during clinical visits, physicians are tasked with multiple responsibilities, including reviewing patient records, inquiring about symptoms, conveying test results, and documenting the encounter. Managing these tasks can be challenging, particularly when physicians must see multiple patients daily. An automatic summarization system could reduce the burden of documentation, enabling physicians to concentrate on patient care, attentive listening, and accurate diagnosis.

Recently, pretrained language models (LMs) have demonstrated impressive performance on a variety of NLP tasks [25, 65, 83, 11, 74, 57, 85]. These models have achieved great success but they have some limitations for these conversational summarization tasks. For corpus-level summarization, while current transformer models in NLP primarily generate text, our approach generates a graph. This graph representation of a dialogue corpus, with nodes describing states, offers a clearer overview of the entire corpus. It

highlights how subdialogues transition, which subdialogues are shared, and which are distinct, compared to using text alone. Second, it is challenging for a transformer model to summarize an entire dialogue corpus due to the input length limitation. The maximum input length remains constrained in transformer models, which can only process a limited number of tokens due to the quadratic memory explosion and the use of predefined fixed-length positional embeddings. While there has been some research aimed at mitigating the long input issues, e.g., memory-efficient attention mechanisms [7, 1] and relative position embeddings [96, 44], processing an entire corpus in a single pass remains impractical. In the context of conversation-level summarization, there are two primary challenges. Again, the large size and the complexity of the attention mechanisms of transformers necessitate significant hardware resources. This requirement can render them impractical in resource-constrained environments, such as small businesses with limited budgets or applications on lower-powered devices such as smartphones. Therefore, there is a critical need for optimizations that enhance the computational efficiency of transformer models in these scenarios. In addition, finetuning transformers with domain-specific data hinges on the availability of high-quality, annotated datasets, which are often costly and time-consuming to develop. This is particularly challenging for proprietary data, which often requires specialized in-house and expensive expertise. Given these obstacles to using transformers in both corpus-level and conversation-level summarization tasks, we propose two new modeling algorithms and a new data collection framework to mitigate the aforementioned three challenges.

1.1 Contributions

We summarize the main contributions in this thesis to deal with the challenges of corpus-level and conversation-level summarization tasks.

- We focus on challenging and less explored dialogue domains, including telecommunications service call center dialogues and auto insurance claim call center dialogues, to summarize these conversations for humans to use in more efficient ways via corpus-level summarization and conversation-level summarization.
- We introduce a new task, corpus-level dialogue summarization, which uses a graph to serve as an interpretable corpus-level summary. This graph reduces the burden of reading a large volume of

conversations for humans to analyze the unlabeled dialogue data. Therefore, the graph increases the efficiency of understanding the entire corpus and facilitates the extraction of useful information, such as good negotiation strategies from good salesmen or efficient strategies of experienced agents.

- Inspired by dialogue state tracking, we propose using the dialogue state at the final turn as a conversation-level summary for summarizing an auto-claim call. This structured summary provides comprehensive information, capturing the essential details needed to determine fault and compensation in a car accident without the need to read the entire dialogue. This resulting summary can potentially increases the efficiency of claim processing.

- Instead of solely relying on raw text input for transformer models, we incorporate additional structured information to enhance task performance and efficiency. We use two types of structure: the latent structure of input dialogues and the expert-designed structure of output summaries. The input latent structure is derived from a corpus-level summary graph, where each dialogue is represented as a path on this graph, creating a sequence of subdialogue states. This structure is used as an additional feature to improve encoder-only hierarchical transformers. Moreover, the expert-designed output structure, such as section titles in medical notes or domain-slot names in task-oriented dialogues, divides a main task into smaller, simpler subtasks. This subtasking enables encoder-decoder transformers to share encoded input and process each subtask output in parallel, leading to more efficient transformer decoding.

- We demonstrate that current larger decoder-only transformer models (e.g., LLaMA, LLaMA2, ChatGPT, GPT-4, etc.) are not always the best solution for all NLP tasks, especially when finetuning a transformer model in complex and specialized domains. In this thesis, we demonstrate that smaller encoder-decoder models can outperform these large decoder-only models, offering better or comparable performance and significantly faster inference speeds in specific domains such as dialogue state tracking and summarization.

- We observe that using only these decoder-only transformer models cannot achieve human-level performance in auto-claim summarization tasks both in data synthesis and annotation. The best performance is obtained when human correction is incorporated into the data synthesis process to correct

the models' annotations.

Our view is that transformer models can perform more effectively and efficiently on conversational AI tasks when we design components or frameworks that consider additional structure of dialogue tasks or human knowledge. Our methods not only provide more effective ways to enhance transformer performance on conversational AI tasks but also enhance the efficiency of transformer models in terms of the data interpretability, data annotation cost and model inference speed.

1.2 Dissertation Overview

In chapter 2, we provide background knowledge on the computational models that we build on this work. First, we describe transformer models and a general overview of recent related work on transformer models for NLP. Then, conversational AI tasks discussed in this thesis are reviewed in terms of types of conversations, applications of conversation understanding and summarization, and the current language and speech processing technology in these applications. We also discuss the limitations of the existing work on conversational understanding and summarization tasks.

Chapter 3 introduces an unsupervised structure learning algorithm to learn *latent dialogue structure* from a corpus of dialogues where structure is represented via a hidden Markov model (HMM). The algorithm provides an unsupervised way to learn the latent structure from unlabeled dialogues and generate additional features for transformer models to boost the performance on predictive downstream conversational AI tasks. The resulting summary graph renders an efficient way to understand the underlying dialogue structure of the entire corpus for humans, mitigating the manual effort of reading dialogue transcripts individually. We evaluate our structure learning algorithm on conversation-level prediction tasks using two public datasets, including negotiation dialogues and problem-solving dialogues, as well as on one proprietary call center dialogue dataset.

In chapter 4, we use introduce a new decoding configuration for structured summaries and boost the efficiency of encoder-decoder transformer decoding. The output structure is defined by domain experts, such as the schema in task-oriented dialogues and the eletronic health forms in clinical conversations. The output structure allows us to divide a main task into into multiple smaller and simpler subtasks. Encoder-decoder

transformer models hence encode a single input conversation once and share the key-value cache of the input conversation across multiple subtasks, and decode the all subtasks in parallel. We assess the inference efficiency using two dialogue datasets: one for task-oriented dialogues and the other for doctor-patient dialogues. Additionally, we demonstrate that the proposed configuration is effective not only in summarization tasks but also in the medical question-answering domain. In addition to measuring actual speed-up, we also provide the detailed algorithm analysis of the transformer decoding to support our experimental results.

In chapter 5, we enhance the performance of finetuned encoder-decoder transformer models on the auto insurance claim call center dialogues via a collaborative human-LM framework for data synthesis to emulate the lengthy and complex human-human dialogues. This framework allows us to boost the performance of finetuned models on real data without accessing the real data to generate synthetic data. We incorporate these synthesized dialogues with real dialogues to finetune a transformer model for a conversation-level summarization task on real dialogues, thereby enhancing the model's performance, particularly on rare or unseen slots. In controlled experiments, we also compare training with our human-in-the-loop-synthesized data vs. fully automatically LM-generated data and find that collaborating humans adds value both in the generation and annotation stages.

Finally, in chapter 6, we conclude this thesis by summarizing the approaches, the developed algorithms and the framework, and our experimental findings. After that, impacts of this thesis and future directions for extending work with enhancing transformer models on conversational AI tasks are discussed.

Chapter 2

Background

In this chapter, we provide a background knowledge of transformer models, conversational AI tasks used in this thesis, and the limitations of the prior work. In section 2.1, we review the general architecture of transformer models and three different model variants, i.e., encoder-only, decoder-only, and encoder-decoder transformer models. Encoder-only models are used for the proposed unsupervised structure learning algorithm for corpus-level summarization and the conversation-level predictive tasks in chapter 3. The decoder-only and encoder-decoder tasks are used in conversation-level summarization tasks, such as dialogue state tracking, medical dialogue summarization, dialogue information extraction in chapter 4 and chapter 5. In section 2.2, we introduce the common approaches for pretraining and finetuning adopted in this thesis. After that, an overview of hidden Markov models (HMMs) with discrete observation sequences for language processing is provided in section 2.3. HMMs are used in structure learning for corpus-level summarization in chapter 3. In section 2.4, we introduce the conversational AI tasks used in this thesis for our proposed two algorithms (THETA and PID) and the framework (DIALGEN). Finally, we introduce the current limitations of the transformer models on these summarization tasks.

2.1 Transformer Models

In this section, we first review the general mathematical framework of transformers, which will be relevant to the algorithm discussion in Chapter 4, and then describe three different architectures used in this thesis.

2.1.1 Neural Attention

The concept of neural attention, first proposed by [5], serves as an effective method to handle representations of varying lengths. This function operates by processing a query vector along with a sequence of distinct pairs of key and value vectors. The resulting output vector is formulated through a weighted sum of the value vectors, where the attention weights are determined by comparing the similarity between the query vector and the key vectors.

Multi-head Attention

The multihead attention in transformer models [105] consists of multiple neural attention heads. These attention heads compute the contextualized output $\mathbf{Z}$ in parallel. Usually, the key and value vectors have the dimension d/h, where d is the dimension of input and output vectors and h is the number of attention heads. The query head vectors $\mathbf{Q}$ can be obtained by projecting the input vectors $\mathbf{N}$,

$$\mathbf{N} = (n_1, \ldots, n_{|\mathbf{N}|}) \in \mathbb{R}^{|\mathbf{N}| \times d}$$
$$\mathbf{Q} = \mathbf{X}W^Q \in \mathbb{R}^{|\mathbf{N}| \times d}$$

where $W^Q \in \mathbb{R}^{d \times d}$ is the linear projection for the query. Similarly, the key and value matrices are computed by projecting another input vectors $\mathbf{M}$,

$$\mathbf{M} = (m_1, \ldots, m_{|\mathbf{M}|}) \in \mathbb{R}^{|\mathbf{M}| \times d}$$
$$\mathbf{K} = \mathbf{M}W^K \in \mathbb{R}^{|\mathbf{M}| \times d}$$
$$\mathbf{V} = \mathbf{M}W^V \in \mathbb{R}^{|\mathbf{M}| \times d}$$

where d is the hidden size; $W^K \in \mathbb{R}^{d \times d}$ and $W^V \in \mathbb{R}^{d \times d}$ are the linear projection matrices.

Given the projected matrices for queries ($\mathbf{Q}$), keys ($\mathbf{K}$), and values ($\mathbf{V}$), the scaled dot-product attention mechanism can be expressed as follows:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d/h}}\right)\mathbf{V}.$$

$\mathbf{QK}^T$ computes the dot product between each query and all keys, resulting in a matrix of scores. $\sqrt{d/h}$ is a scaling factor, used to stabilize the gradients during learning. softmax is applied to each row of the scaled scores, converting them into attention scores that sum to 1.

More specifically, we use the self-attention of the T5 model as an example, where the relative position information between token i and j is provided by an inductive bias b_{j-i}. The similarity score between n_i and m_j can be denoted as follows,

$$\alpha_{ij} = \frac{1}{\sqrt{d/h}}(n_i W^Q)(m_j W^K)^T + b_{j-i},$$

where the term b_{j-i} introduces a relative positional bias, which is used to learn the relative encoding in T5. After that, we apply softmax function on the scores associated with the token i and use these normalized scores and value matrix $\mathbf{V}$ to obtain the weighed output o_i at the position i,

$$o_i = \text{softmax}(\alpha_{i1}, \alpha_{i2}, \ldots, \alpha_{im})V.$$

Then, the matrix of the weighted outputs $\mathbf{O} = (o_1, \ldots, o_n) \in \mathbb{R}^{|\mathbf{N}| \times d}$ is projected by a linear matrix $W^O \in \mathbb{R}^{d \times d}$ to obtain the final attention output

$$\mathbf{Z} = \mathbf{O}W^O \in R^{|\mathbf{N}| \times d}.$$

In the self-attention mechanism, the query, key, value are projected from the same input matrix; therefore, $\mathbf{N}$ is equivalent $\mathbf{M}$. On the other hand, in the case of cross-attention, the variable $\mathbf{N}$ originates from the current step in the decoder, while $\mathbf{K}$ and $\mathbf{V}$ is sourced from the key-value cache of the encoder's final layer.

2.1.2 Transformer Architectures

Transformers have become a fundamental model in the field of natural language processing (NLP), dramatically transforming how machines comprehend and produce human language. Compared to a recurrent neural network, which summarizes work history in a single vector at each time step, transformers are pri-

marily characterized by their reliance on the attention mechanism to weigh the importance all words in a sentence. In the following paragraphs, we provide a description of the three main types of transformer models: *encoder-only*, *decoder-only*, and *encoder-decoder* models. These models tend to be used in different NLP tasks.

Encoder-only Transformer Models

Encoder-only transformer models are designed to process input sequences and produce a continuous representation of the input that captures contextual relationships between words. The encoder-only transformer models are widely used for sentence representation or classification tasks, such as sentence classification, language understanding, and named entity recognition. The key mechanism at play is bi-directional self-attention in the encoder. This allows each word to attend to all other words in the input sequence, thereby understanding the context around each word. The self-attention mechanism enables the model to dynamically weigh the relevance of each word in a sentence relative to all other words, enhancing the model's ability to capture contextualized language representation. Popular examples of encoder-only models include BERT [25] and its derivatives like RoBERTa [65].

Decoder-Only Transformer Models

Decoder-only transformer models are primarily used for generative tasks such as text completion and language modeling. The fundamental mechanism governing these models is causal self-attention, also known as auto-regressive attention. In causal self-attention, each word can only attend to previously generated words in the sequence. This limitation guarantees that the generation of each subsequent word is solely based on its predecessors, effectively preventing information leakage during text generation. Such models are excellent at producing coherent and contextually appropriate continuations of text, making them ideal for applications like chat-bots and creative writing aids, but they can also be used in any task involving output text. Examples of decoder-only transformer models include GPT [82, 83, 74] and LLaMA [101, 102].

Encoder-Decoder Transformer Models

Encoder-decoder transformer models combine the functionalities of both encoder-only and decoder-only models to handle tasks that involve both understanding an input sequence and producing an output sequence.

These tasks typically include machine translation, summarization, and question answering. In encoder-decoder models, the encoder first processes the input sequence to create a context-rich representation. This representation is then passed to the decoder, which generates the output sequence. The decoder attends not only to the output it has already produced via self-attention but also to the encoder's output through cross-attention. This dual attention mechanism allows the decoder to adapt its outputs based on both the content it has generated and the input provided by the encoder. The well-known examples of encoder-decoder transformer models are T5 [85] and BART[57].

Both decoder-only and encoder-decoder models excel in generative tasks, each offering distinct advantages depending on the scenarios. Recent research [109] indicates that scaling up decoder-only transformer models enhances in-context learning, which is particularly useful when labeled data is limited, thereby improving generalization to unseen domains. However, these larger decoder-only models come with higher computational costs and may perform worse than smaller, specialized encoder-decoder transformer models [55].

Each of these transformer architectures plays a important role in the field of NLP, leveraging the attention mechanism to perform a wide range of language understanding and generation tasks with remarkable improvement.

2.2 Transformer Pretraining and Finetuning

In this section, we first introduce the common approaches for pretraining transformer models. In chapter 3, we adopt domain-adaptive pretraining to train a BERT model on individual datasets, aiming to achieve better representations for speaker turn embeddings. Although we do not apply in-domain adaptive pretraining for decoder-only and encoder-decoder models, we outline the common training practices to highlight the differences between them and encoder-only transformers. For finetuning, we introduce the objective functions for classification tasks used in chapter 3 and generation tasks used in chapter 4 and chapter 5.

2.2.1 Pretraining

There are many different pretraining methods for each transformer architecture, including encoder-only, decoder-only, and encoder-decoder models. In this section, we use BERT, GPT, and T5 as examples to

illustrate these methods.

Pretraining for Encoder-only Transformer Models

We use bidirectional encoder representations from transformers (BERT) as an example. BERT is pretrained using a bidirectional approach to understand the context of a word from both its left and right sides. Its primary pretraining tasks are masked language modeling (MLM) and next sentence prediction (NSP).

Masked Language Modeling (MLM). In MLM, random words in the input sentence are masked, and the model learns to predict these masked words. Given a sentence $X = (x_1, x_2, \ldots, x_n)$, we randomly mask some tokens to form X_{masked}. The objective is to minimize the loss function:

$$\mathcal{L}_{\text{MLM}} = -\sum_{i \in M} \log P(x_i \mid X_{\text{masked}}),$$

where M is the set of masked token positions.

Next Sentence Prediction (NSP). NSP involves predicting whether two given sentences follow each other in the text. For a pair of sentences (A, B), the objective is to minimize the binary classification loss:

$$\mathcal{L}_{\text{NSP}} = -\log P(\text{isNextSentence} \mid A, B) - \log P(\text{isNotNextSentence} \mid A, B)$$

Pretraining for Decoder-only Transformer Models

Generative pretrained transformer (GPT) is a decoder-only transformer and is pretrained using an autoregressive approach, where the model learns to predict the next token in a sequence. The primary pretraining task is language modeling (LM).

Language Modeling (LM). In LM, given a sequence of tokens $X = (x_1, x_2, \ldots, x_n)$, the model learns to predict the next token x_{t+1} based on the previous tokens $(x_1, x_2, \ldots, x_t)$. The objective is to minimize the negative log-likelihood:

$$\mathcal{L}_{\text{LM}} = -\sum_{t=1}^{n} \log P(x_{t+1} \mid x_1, x_2, \ldots, x_t)$$

Pretraining for Encoder-decoder Transformer Models

Text-to-text transfer transformer (T5) is an encoder-decoder transforms all NLP tasks into a text-to-text format, using text sequences for both input and output. Its pretraining objective, similar to BERT's MLM, is designed as a text generation task aimed at reconstructing a corrupted document.

Corrupted Document Reconstruction. T5 uses a denoising objective where a portion of the input sequence is corrupted, and the model learns to reconstruct the original sequence. Given an input sequence X and a corrupted version X_{corrupt}, the objective is to minimize the reconstruction loss:

$$\mathcal{L}_{\text{T5}} = -\log P(X \mid X_{\text{corrupt}}).$$

In summary, BERT focuses on bidirectional context through MLM and NSP, GPT leverages an autoregressive LM approach for next-token prediction, and T5 unifies all tasks into a text-to-text format with a denoising objective. These pretraining methods are used to better initialize model parameters via self-supervised learning without labeled data and have been proven to be useful when the pretrained models are finetuned on downstream tasks.

2.2.2 Transformer Finetuning

After pretraining, transformer models are commonly finetuned on specific downstream tasks to adapt them to various applications. Finetuning involves training the pretrained model on a smaller, task-specific dataset. Below, we discuss finetuning strategies for these models, focusing on classification tasks for encoder-only and both classification and generation tasks for decoder-only and encoder-decoder models.

Finetuning Encoder-only Model for Classification

In chapter 3, we use BERT as a backbone to initialize our hierachical BERT model. BERT is often finetuned for classification tasks using the special `[CLS]` token. During pretraining, the `[CLS]` token is added at the beginning of every input sequence, and its corresponding output representation is used for classification.

Given an input sequence $X = (\texttt{[CLS]}, x_1, x_2, \ldots, x_n \texttt{[SEP]})$, the output representation of the `[CLS]` token $h_{\texttt{[CLS]}}$ is fed into a classifier to predict the class label. The classification loss is minimized as follows:

$$\mathcal{L}_{\text{classification}} = -\log P(y \mid h_{\texttt{[CLS]}}),$$

where y is the true class label.

Common classification tasks for finetuning BERT include document topic classification, successful call prediction, extractive-based question answering, sentiment detection and spam detection, etc.

Finetuning Decoder-only and Encoder-decoder Models for Generation and Classification

In chapter 4 and chapter 5, we use LLaMA2 and T5 are primarily finetuned for generation and classification tasks. These generative models can also be used as a classifier if labeled data, proper instructions, and post-processing are given.

Generation Task. For generation tasks, LLaMA2 and T5 are finetuned to produce a sequence of tokens based on the given input. The objective is to minimize the negative log-likelihood of the target sequence $Y = (y_1, y_2, \ldots, y_m)$ given the input sequence X:

$$\mathcal{L}_{\text{generation}} = -\sum_{t=1}^{m} \log P(y_t \mid X, y_1, y_2, \ldots, y_{t-1})$$

Common generation tasks for finetuning LLaMA2 and T5 include machine translation, generation-based question answering, summarization, story generation, dialogue systems, and code generation, etc.

Classification Task. For classification tasks, both LLaMA2 and T5 can be fine-tuned by providing the model with labeled data, appropriate instructions, and post-processing. In LLaMA2, a classification token or prompt can be added to the input sequence, and the model's output is used to predict the class label. Similarly, T5 can be finetuned by framing the classification task as a text-to-text problem, where the input is a description of the task and input data and the output is the predicted class label. The objective function is the same as generation task.

In summary, while BERT is typically finetuned for classification tasks using the `[CLS]` token, LLaMA2 and T5 are versatile and can be finetuned for both generation and classification tasks, depending on the nature of the labeled data and the instructions provided.

2.3 Hidden Markov Models

An hidden Markov models (HMM) is a classifcal statistical tool for modeling time series data where the states of the process are not directly observable but can be inferred through observable events. They were popular for early work on unsupervised learning of continuous speech recognition, part-of-speech tags, and other tasks where the goal is to infer hidden discrete state from observable sequences. An HMM assumes the unobserved state sequence is a Markov process and each hidden state produces an observation that depends only on that state. The sequence of these observations gives clues about the sequence of hidden states. Observations may be discrete-valued or continuous. In this work, we will use discrete observations. Let $s_t \in \{1, \ldots, N\}$ be the state at time $t \in \{1, \ldots, T\}$, and the observation be $c_t \in \{1, \ldots, K\}$.

An N-state HMM is characterized by three sets of parameters:

Initial state probability (π)

An initial state probability, $\pi(s_1)$, is a K-dimension vector, indicating how likely it is that the model will start in any given state.

Emission probability (η)

These define the probability (an $N \times K$ matrix) of observing a particular symbol (c_t) from a hidden state (s_t), denoted as $\eta(c_t \mid s_t)$. For example, in the context of POS tagging, this might represent the probability that a noun will generate a particular word. For example, the probability that a noun will generate the word "cat" can be written as

$$p(c_t = \text{cat} \mid s_t = \text{noun}).$$

Transition probability (γ)

A transition probability indicates the likelihood of moving from one state (s_t) to another state (s_{t+1}). In POS tagging, this could model the probability of transitioning from one POS tag to another. The transition

probability is an $N \times N$ matrix and can be denoted as $\gamma(s_{t+1} \mid s_t)$. The topology of the state space can be represented with a graph corresponding to non-zero values of π and γ.

In this thesis, we use an HMM to model the hidden dialogue state sequence $S = (s_1, \ldots, s_T)$ given the observed sequence $C = (c_1, \ldots, c_T)$ with the sequence length T. The state S_t can be thought of as a subtask or subtopic in a dialogue. The observation c_t is a discrete representation of a user turn (automatically learned) that is intended to capture speaker intent, which can be thought of as a fine-grained dialogue act. An HMM is a statistical model that characterizes an observation sequence C in terms of a discrete, latent (hidden) Markov state sequence S,

$$\begin{aligned} P(C) &= \sum_{\text{all } S} P(C, S) \\ &= \sum_{\text{all } S} \pi(s_1) \prod_{t=1}^{T} \eta(c_t|s_t)\gamma(s_{t+1}|s_t), \end{aligned}$$

where s_{T+1} is a dummy stopping state.

Given a specified topology—defining the structure of a hidden Markov model (HMM), including the number of hidden states, the initial emission probabilities, and the initial transition probabilities between these states—the parameters will be learned using the Expectation-Maximization (EM) algorithm [72]. The Viterbi algorithm [28] can then be applied to find the most likely state sequence during inference. We will discuss more details of how to automatically learn the topology given a dialogue corpus in chapter 3.

2.4 Conversational AI Tasks

In this thesis, we explore various conversational AI challenges, specifically focusing on two types of summarization tasks: *corpus-level* and *conversation-level* summarization. We use three datasets for our experiments, including a public negotiation dataset, a public customer service dataset, and private call center data from a U.S. telecommunications company. Due to the lack of available gold summary labels, which are impractical to obtain in real-world settings, we instead leverage the derived latent dialogue structure from the graph as additional features for predictive tasks at the conversational level. In chapter 4 and chapter 5, we focus on conversation-level summarization tasks such as dialogue state tracking (DST), medical dia-

logue summarization, and the information extraction from auto insurance claim calls. We categorize both dialogue state tracking and dialogue information extraction as conversation-level summarization tasks, since the dialogue state represents as a tabular summary of user needs up to the current interaction, while dialogue information extraction serves as a tabular summary of the entire conversation. Unlike DST, the information extracted from each turn is collected to create a summary of the call rather than to generate a virtual agent's response or make an API call. In addition, the summary includes entities that are associated with attributes (slots) and values. To evaluate models on this IE task, we introduce entity-centric scoring methods that allow partial matching of multiple and descriptive values.

In the following paragraphs, we introduce the different types of dialogues in the thesis and their evaluation metrics.

Negotiation Dialogues

Negotiation dialogues involve two parties trying to come to agreement on terms of a transaction, contract, policy, etc. In this thesis, we look at dialogues proposed in He et al. [39] involving transactions or agreements, often focusing on price negotiation for goods. The dialogues involve different strategies like persuading the other party by highlighting the quality of an item. By analyzing and summarizing patterns and strategies across multiple negotiations, one can extract insights into what constitutes a compelling sales strategy, thereby enabling sellers or buyers to enhance their negotiation skills and effectiveness in future interactions. For evaluation, this thesis builds on prior research [39] by using the "sale price ratio" to measure the success of a seller. A higher ratio indicates that a seller sold the product at a better price.

Customer Service Dialogues

Customer service dialogues are structured conversations between a customer and a service agent, focused on resolving issues, addressing inquiries, or offering support. These dialogues are goal-oriented and often follow specific protocols, covering a range of types from problem-solving dialogues to information-gathering dialogues. Analyzing and summarizing these interactions are essential for refining agent training, improving dialogue system design, boosting customer satisfaction, and streamlining the process for any necessary follow-up actions. In this thesis, we focus on three different datasets for customer service dialogues, including one public dataset, action-based conversations dataset (ABCD) [15], and two private datasets, call center

dialogues from an US telecommunications company (CALL CENTER) [67] and call center dialogues from a car insurance company (AIC) [68].

The ABCD and CALL CENTER datasets are chosen for corpus-level graph-based summarization aimed at improving agent training by distinguishing between effective and ineffective dialogues. We assess the quality of the resulting graphs used to generate corpus-level graph-based summaries by incorporating the state sequence as an additional feature for predictive tasks, such as failure call prediction and call type classification. Further details will be discussed in chapter 4. The auto insurance dataset (AIC) is used for conversation-level summarization to automate the claims process, which helps in saving time for subsequent actions and facilitates fault determination. We evaluate this task by calculating the F-measure scores for all required information. More comprehensive details of the task and evaluation metric will be provided in chapter 5.

Task-oriented Dialogues

Task-oriented dialogues [111, 110, 13] in NLP are designed to accomplish specific tasks such as booking tickets or scheduling appointments following a predefined schema associated with a database. This schema plays a critical role in structuring the dialogue, guiding it toward the efficient collection of user requests. A popular task in task-oriented dialogues is dialogue state tracking, where the system maintains a state of the conversation based on past interactions, where the state consists of values of all slots in the ontology. This dialogue state acts as a dynamic summary that helps the system understand the current context and user requirements. Using this summary, the agent can either call the appropriate API to fulfill the user's requests or ask additional questions to clarify the user's needs, thereby enhancing the system's ability to respond accurately and effectively.

Multiple evaluation metrics have been used for assessing system performance. Joint goal accuracy (JGA) is a primary metric that measures the system's ability to accurately predict the entire set of slot values at each turn of a dialogue including null values. This metric is stringent as it requires correct predictions across all goal aspects to be considered successful. Turn-level belief (TLB) accuracy assesses the system's capability to extract an user's intentions at a particular turn.

Medical Dialogues

Medical dialogues occur between a patient and a provider, such as a physician. During these exchanges, patients generally present their symptoms, medical history, and concerns, while physicians pose specific questions, perform assessments, and offer treatment recommendations. This thesis centers on one specific application: distilling a dialogue into a structured summary that contains all essential information discussed under the format of electronic health record (EHR) notes. Automated summarization can facilitate the effective documentation of patient interactions and supports ongoing care continuity. We use the data proposed by Yim et al. [121].

The evaluation typically involves comparing the automatically generated summaries against a human-written reference summary. In prior work, the method for evaluating the quality of these summaries is using the ROUGE metric [61]. ROUGE measures the overlap of n-grams, word sequences, and word pairs between the automated summary and the reference summaries. It includes different variants such as ROUGE-N (which compares n-grams), ROUGE-L (which focuses on the longest common subsequence).

2.5 Limitations of Related Work of Transformer Models

High Cost of Encoding and Generating Long Sequences

Transformer models, particularly those developed based on the foundational architecture introduced by Vaswani et al. [105], encounter a significant challenge related to the length of input data. The self-attention mechanism inherent in these models causes the computational complexity to increase quadratically with the length of the sequence. This results in high computational costs and substantial memory requirements when dealing with long sequences. Such challenges are particularly evident in tasks that involve translating or summarizing lengthy documents/dialogues. To mitigate the issue, researchers have innovated attention mechanisms that conserve memory, such as the Performer [17] and the Linformer [107], which simplify the attention mechanism to make its computational demand scale linearly with the length of the input, thus easing both computational and memory burdens. Although the prior advancements have focused on addressing these length constraints, the challenges are still acute when dealing with real-world data, such as lengthy dialogues from call centers that may last over an hour. These extended conversations result in very long input sequences that increase the computational difficulties. In chapter 3, we address the challenge of

input length constraints by using an encoder-only hierarchical transformer model. In chapter 4, we leverage the expert-designed structure to divide the output sequence into multiple smaller and simpler subtasks and reduce the burden of duplicate encoding of long input sequence by sharing the common input.

Limitation of Generating Corpus-Level Summaries

Current transformer models in NLP excel at generating text. However, using text alone may not be the most efficient way to distinguish between types of dialogues within an entire corpus. This inefficiency arises because different types of dialogues may share common subdialogues, such as greetings, verification, and closings in goal-oriented dialogues, while diverging in other subdialogues, such as problem-solving for billing or user account access issues. Conversely, a graph can better represent these characteristics. Each node in the graph represents a specific type of subdialogue, and the edges represent transitions between subdialogues. Each path in the graph can represent a specific type of dialogue. The shared common subdialogues (nodes) can be reused across different types of dialogues (paths), clearly illustrating the differences between dialogue types. Additionally, the aforementioned limitation of input length in transformer models makes it impractical for them to process entire dialogues to generate corpus-level summaries. In chapter 3, we address the challenge of generating corpus-level graph-based summaries by integrating representations produced by an encoder-only transformer model to generate a graph via an unsupervised structure learning algorithm.

Considering Raw Text Only without Leveraging Structured Information.

Most current studies on transformers handle input and output dialogue tasks as raw, flattened text, overlooking the advantages of utilizing structured information in either the input or output text. However, incorporating structure into transformers can significantly enhance their performance and efficiency. Typically, work on conversational systems treats dialogues as linear sequences of text, akin to written text. This thesis explores unsupervised learning strategies for integrating structured information into a state-of-the-art hierarchical transformer-based text model. Moreover, the decoding phase of transformers has relatively low operation intensity compared to the training phase. During decoding, only one token is generated at a time, yet all projection matrices and the key-value cache must be loaded into memory to compute a single new token, leading to slower response times and lower throughput in real-time applications. The data in conversation-

level summarization task comes with structured output. This expert-designed structure can break down a complex task into smaller, more manageable subtasks, referred to as "decomposable tasks." For example, dialogue state tracking can be seen as a summary of the user's needs up to the current turn, with the output dialogue state divided based on schema structure, such as slot names. In the medical domain, a patient's electronic health record is often organized into sections like subjective, objective exam/results, plan, and assessment. Currently, popular methods either treat all subtasks in an example as a large text chunk, requiring the model to generate all output at once, or contextualize each subtask and input dialogue/document separately, leading to duplicate input encoding and inefficiency. In chapter 3 and chapter 4, we discuss how to leverage latent input dialogue structure and expert-designed output structure to enhance transformer task performance and its efficiency.

High Costs of Data Annotation for Conversation-level Summarization in Private Dialogues

The scarcity of high-quality annotated data is particularly evident in domains involving private human-human interactions, such as call center dialogues. These interactions are inherently private and sensitive, leading to significant restrictions on the availability of these natural problem-solving dialogues for public research purposes. Moreover, even when private datasets are available, the cost of annotation is prohibitively high. Specialized in-house expertise is required to ensure the accuracy and relevance of annotations, further exacerbating already limited training resources.

To reduce data collection costs, researchers have explored the use of language models (LMs) to generate synthetic training data [63, 108, 6, 59, 4]. Synthesized data can target long-tail phenomena [18, 127, 123, 40] and allow for public release of data that closely emulates real-world privacy-constrained domains, such as the medical domain [78]. Although LMs can follow instructions to generate text that closely resembles human writing, there can be challenges to ensure that the data are diverse and not too simplistic [99, 63]. In addition, they still suffer from incoherence and consistency issues [19, 26]. To mitigate the shortcomings of LMs, human-LM collaboration can offer a robust solution, leveraging the strengths of both humans and machines [95, 103]. In chapter 5, we discuss our approach, a human-LM collaborative framework that leverages a layperson and an LM to synthesize more realistic dialogues that improve the performance of the downstream models on real-world call center data.

Chapter 3

Dialogue Latent Structure Learning

3.1 Introduction

Increasingly, language understanding applications involve conversational speech and text. Much attention has recently been directed at human-agent dialogue systems, including virtual assistants, interactive problem solving, and information seeking tasks (e.g., conversational question answering). However, automatic understanding of human-human conversations is also of interest for problems such as call-center analytics, conversation outcome prediction, meeting summarization, and human-agent interaction involving multiple people. The focus of this chapter is on summarizing the human-human conversations at the corpus level and providing a graph-based summary of dialogue structures. The resulting structure is leveraged to extract additional turn-level features to enhance the performance of encoder-only transformer models on predictive tasks.

Like written documents, goal-oriented conversations tend to have ordered structure (openings, context setting, problem solving, etc.). However, in human-human conversations (both text and speech), participant roles factor into the structure, and the structure is less rigid due to the need to accommodate miscommunications and varying objectives. Yet, most work on conversational systems treats dialogues like written text, i.e., the dialogue history is a linear sequence of text. In this paper, we explore unsupervised learning strategies for adding structural information to a state-of-the-art hierarchical encoder-only transformer model.

Linguistic analysis of conversations often involves associating speaker utterances with dialogue acts

(DAs), e.g., question, statement, backchannel, clarification, etc. [47, 20], and segmenting the conversation into nested subsequences of participant turns that reflect a common topic or conversational goal [34]. Past studies have explored using such structure, particularly DAs, to improve automated human-agent dialogues. Here, we use hierarchical structure (both turn-level DA labels and sub-dialogue states) to improve classification of human-human conversations. Specifically, we introduce **T**hree-stream **Hie**rarchical **Tra**nsformer (THETA), which integrates transformer representations of the DA and sub-dialogue state sequences into a hierarchical transformer (HET) [92, 76] operating on the original text. In addition to improving performance, the use of discrete structural cues in classification can support conversation analysis. For example, we can identify sales strategies that are more likely to lead to a successful outcome or use the sub-dialogue state sequence to summarize frequently visited states in unsuccessful interactions.

Since hand-annotation of structure can be costly and inventories vary across tasks, there is substantial interest in unsupervised learning of structure for specific task domains. Here, the approach to structure learning involves two steps. First, we use a clustering algorithm to learn a mapping of utterance embeddings to discrete categories, which serve as an unsupervised version of DAs. Each conversation is then represented by the discrete sequence of cluster identifiers (IDs) associated with the sequence of utterances. Using the collection of discretized conversations, we automatically learn the topology of a latent finite-state model over these sequences, i.e., a hidden Markov model (HMM), using a greedy state-splitting algorithm that maximizes the likelihood of the sequence data without requiring any annotations. The states of the HMM correspond to different sub-dialogues that may be associated with specific topics, strategies or subtasks. The sub-dialogue structure of a new conversation is identified by finding the most likely state sequence given that discretized utterance sequence.

In this chapter, the learned structure is assessed in experiments on three conversation-level classification tasks: i) buyer/seller negotiation outcomes on CRAIGSLISTBARGAIN [39], ii) conversation category in the Action-based Conversations Dataset (ABCD) [15], and iii) client callback prediction in a private call center corpus. In each task, we find that a combination of both utterance-level category and sub-dialogue state information lead to improved performance. Further, we use automatically generated descriptions of the clusters and sub-dialogue states to provide an interpretable view of the finite-state topology and a summarized view of a conversation. Anecdotally, we find that this structure lends insights into how participant

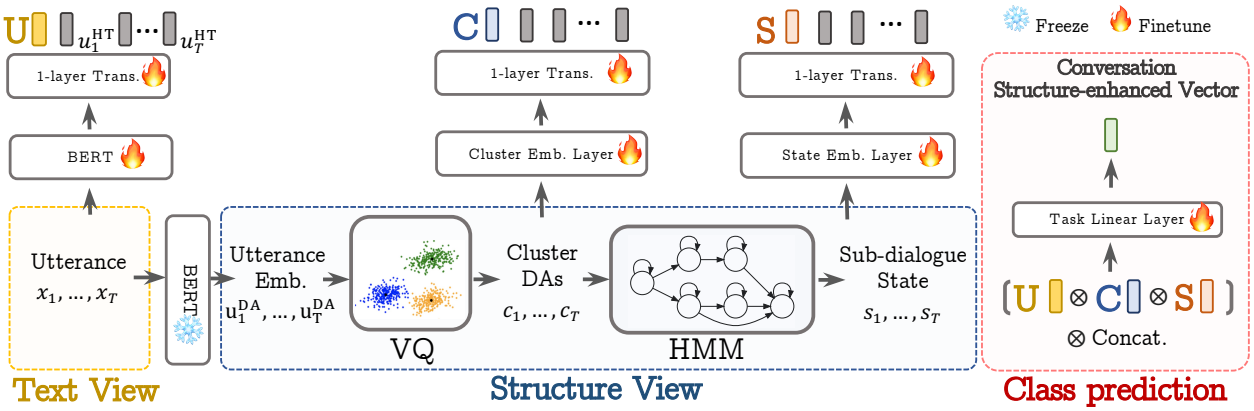


Figure 3.1: Overview of THETA conversation encoding. The text of each utterance text is encoded by BERT, and a 1-layer transformer further contextualizes utterance embeddings to generate the text vector **U**. For structure, utterances are mapped to K-means dialogue acts (DAs), which are input to an HMM to decode sub-dialogue states. 1-layer transformers are applied to sequences of DAs and sub-dialogue states, yielding cluster vector **C** and state vector **S**. The concatenation of **U**, **C** and **S** is fed into a linear layer to obtain the structure-enhanced vector for the predictive task. For simplicity, Emb. and Trans. stand for embedding and transformer, respectively.

strategies (state paths) are associated with different conversation outcomes.

The contributions of this work are as follows. First, we introduce a simple unsupervised approach to learn a hierarchical representation of conversation structure that includes turn-level labels and sub-dialogue segmentation, accounting for participant role. Using the three conversation-level classification tasks, we demonstrate that integrating the structural information into a state-of-the-art hierarchical transformer consistently improves performance. Lastly, we show how the discrete representation of structure combined with automatic summarization can provide a mechanism for interpreting what the model is learning or for conversation summarization and analytics.

This chapter contains material that was originally published in the previous work [67]. I was the main contributor of this work, and Yushi Hu helped with setting up unsupervised summarization for HMM graphs. We discussed and wrote the paper with the other collaborators throughout the project.

3.2 Method

As shown in Figure 3.1, THETA represents the sequence of turns in a conversation using: i) a hierarchical encoder-only transformer (HET) operating on a turn-segmented word sequence, ii) an encoder-only trans-

former operating on a sequence of turn-level DAs, and iii) a separate encoder-only transformer operating on a sequence of sub-dialogue states derived from the DAs. The conversation-level vectors produced by the three transformers are concatenated and used in a final task-specific layer for conversation classification tasks. At the time of this work, the HET alone was the state-of-the-art model for conversation-level tasks. The DA and sub-dialogue states comprise the structural information that enhances the HET for improving performance of the end task. In addition, the discrete nature of the structure representation provides a mechanism for analyzing the conversation classes via summarization of utterances associated with the DA labels or sub-dialogue states.

3.2.1 Model Components

Definitions

More formally, each dialogue consists of a sequence of words (or tokens) $X = [x_1, \ldots, x_T]$ associated with T customer/agent (or seller/buyer) utterances, where x_t is the subsequence of words associated with the tth utterance.[1] The word sequence is decorated with three special tokens: `[CLS]`, `[PTY]` and `[SEP]`, where `[PTY]` indicates the utterance speaker role (`[AGT]` for agent/seller and `[USR]` for customer/buyer). The word sequence X is mapped to two sequences of utterance-level embeddings $U^v = [u_1^v, \ldots, u_T^v]$, $v \in \{\text{HET}, \text{DA}\}$. The vector u_t^{HET} is output from the last layer of the HET that is used to derive the text-based conversation-level vector $\mathbf{U}$. The vector u_t^{DA} is the output of a separate transformer, which is then mapped to a DA category c_t to produce the sequence $C = [c_1, \ldots, c_T]$. The sequence C is associated with a hidden subdialogue sequence that is represented using the HMM state sequence $S = [s_1, \ldots, s_T]$. Additional transformers derive conversation-level vectors $\mathbf{C}$ and $\mathbf{S}$ from C and S, respectively. THETA enhances the conversation representation by concatenating $\mathbf{U}$, $\mathbf{C}$ and $\mathbf{S}$ together for input to a task-specific layer.

Hierarchical Encoder-only Transformer

The hierarchical encoder-only transformer [76] has been shown to be useful for classifying long documents (like customer support conversations), which exceed the length limits placed on encoder-only transformer models due to the quadratic complexity of the self-attention module. At a high level, two encoder-only

[1] We use the term "utterance" although some conversations involve text-based interactions.

transformer blocks, a lower utterance encoder-only transformer and an upper conversation encoder-only transformer are stacked together for encoding dialogues. Here, the utterance-level encoder-only transformer first encodes utterances into utterance embeddings, one for each utterance. In this case, the first contextualized token embedding as the utterance embedding, which corresponds to the sentence-level `[CLS]` token. The sequence of utterance embeddings augmented with a conversation-level `[CLS]` token are then fed as inputs to another one-layer conversation-level transformer to further contextualize the vector sequence. We use the output vector associated with the conversation-level `[CLS]` token as the conversation representation.

Dialogue Act Sequence Module

To obtain the DA labels, we first derive an utterance embedding u_t^{DA} by mean pooling the final layer of the BERT transformer.[2] The resulting embedding is mapped to a DA class c_t using a vector quantization (VQ) approach: K-means clustering is used to learn the classes, and vectors are labeled at inference time by minimizing the Euclidean distance to cluster means. The number of clusters is treated as a hyperparameter of the overall model. We apply K-means clustering separately for utterances from the two different participant roles, so the DA index reflects the role. This simple approach is motivated by prior work on unsupervised learning of DA categories [12], which showed that K-means clustering gives a performance that is only slightly worse than HMM-based learning.

In linguistic analyses, a turn can contain a sequence of DAs. Our work assigns a single DA to a user turn, as in other work using unsupervised learning as well as the negotiation data set that we report results on. Since the prior work often uses "dialogue act" for turn-level labels, we have chosen to use the DA term here, acknowledging the abuse of terminology. For complex tasks like the call center data (and other data with real users), the turns will involve multiple dialogue acts, in which case a large number of clusters is useful.

Sub-Dialogue Sequence Module

The DA sequence C is input to a hidden Markov model (HMM) to derive the sub-dialogue structure. An HMM is a statistical model that characterizes an observation sequence C in terms of a discrete, latent

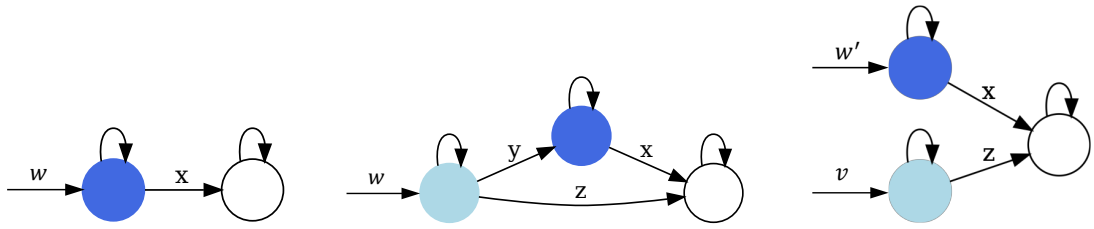
[2] We also experimented with using the `[CLS]` token, but mean pooling gave better results.

(hidden) Markov state sequence S,

$$P(C) = \sum_{\text{all } S} P(C, S)$$
$$= \sum_{\text{all } S} \pi(s_1) \prod_{t=1}^{T} \eta(c_t|s_t)\gamma(s_{t+1}|s_t),$$

where π, η, and γ are start-state, observation, and transition distributions, respectively. s_{T+1} is a dummy stopping state. The HMM is used to decode the hidden sub-dialogue state sequence S, which provides a segmentation of the conversation into different stages or sub-tasks in problem solving or negotiation. The HMM topology and parameters are derived using unsupervised learning as described in the next section.

3.2.2 Sub-Dialogue Structure Learning



(a) Before split. **(b)** Temporal split. **(c)** Contextual split.

Figure 3.2: The design of two split methods. The dark-blue state is chosen to be split. The white state is the end node. The light-blue state is the new state after split. Transitions to other states are omitted for simplicity.

Given a specified topology, inference and training algorithms for HMMs are well established [72]; the Viterbi algorithm gives the most likely state sequence, and the Expectation-Maximization (EM) algorithm is used for parameter estimation. To automatically learn the HMM topology, we apply a greedy state splitting algorithm [75], which learns a left-to-right topology by constraining states to inherit the transition constraints of their parent. The standard objective is maximum likelihood of the DA sequence, which is unsupervised with respect to the conversation-level task.

Topology learning is outlined in Algorithm 1 The initial model has a 3-state left-to-right topology, initialized (assuming 15% of utterances in first state, 70% of utterances in the second state, 15% utterances in the final state). All states are initialized with a transition to the end state. The parameters are updated itera-

Algorithm 1 State-splitting Algorithm for Topology Learning

Input : n: number of splits. τ_i: topology after i split.
Initialization: Run the EM algorithm on 3-state initial topology τ_0.
for $i = 1$ *to* n **do**
  Select state $s \in \tau_{i-1}$ to split based on max entropy of observation distribution $\eta_{i-1}(c|s)$
  Apply temporal split and get new topology $\tau_{i,t}$
  Apply contextual split and get new topology $\tau_{i,c}$
  Run the EM algorithm on $\tau_{i,t}$ and $\tau_{i,c}$
  Select the topology with higher likelihood as τ_i

tively until the improvement is lower than a fixed threshold or the iteration count exceeds some number. At each iteration, the state with the highest entropy of emission probability is chosen to be split. The topology can change into two new configurations corresponding to temporal and contextual splits (Figure 3.2). The EM algorithm is applied again on each configuration and the topology that leads to the higher likelihood is chosen. Transitions with probability below a threshold (1×10^{-4}) are pruned. We iteratively conduct the splitting until the total number of states reaches the desired value (a hyperparameter). In the previous study [75], the HMMs used continuous observation distributions. The splitting approach described below was designed for discrete distributions.

Temporal split

The temporal split provides more detailed sequential structure along a path. Figure 3.2b shows the result of a temporal split on the selected state (dark-blue) in Figure 3.2a. The light-blue node is the new child state that inherits the observation distribution and all incoming edges (e.g., w) [3], and the associated transition probabilities of the dark-blue state. The incoming edges are removed from the dark-blue node. The light-blue node also inherits all outgoing edges of the dark-blue node but the outgoing edges are not removed from the dark-blue node. A new edge y is added from the light blue state to the dark-blue state. To ensure the sum of outgoing node is one, the probabilities of the edges y and z are initialized as $p_x/2$.

Contextual split

The contextual split allows for alternate sub-dialogue paths. Figure 3.2c illustrates the contextual split applied on the dark-blue state. The light-blue state inherits everything but the observation distribution of

[3] The other states in the graph are not shown in the picture for simplicity.

dark-blue state. All the incoming and outgoing edges are copied. The outgoing edges inherit the transition probabilities $p_z = p_x$. The incoming edge probabilities will come from the same state, so $p_{w'} = p_v = p_w/2$. With the aim of modeling different types of paths, when copying the observation probabilities to the light-blue state, we omit the top emission probability of the dark-blue node and set it to 0 and normalize the rest of probabilities.

3.2.3 Pre-Training and End-Task Training

Both for initializing the HET and for deriving the DAs, we use the encoder-only transformer, BERT model [25], for encoding individual utterances u_t, pre-trained using masked language modeling and next-sentence prediction. Due to the style differences of dialogue data vs. written text, we apply domain-adaptive pretraining (DAPT) [38] to adapt BERT for dialogue applications. As shown later (section 3.4), adapting BERT with DAPT provides substantial improvement in terms of predictive power as well as optimization stability. For the HET alone, supervised training involves learning the weights of the final task-level linear layer, the utterance-level transformer, and the word-level transformer.

For THETA, supervised training involves learning the weights of the cluster- and state-level transformers, in addition to all updates associated with the HET component described above. The cluster sequences are obtained using the word-level transformer with DAPT and the associated cluster mapping obtained from unsupervised learning, i.e., without task-level finetuning. Similarly, there are no task-level supervision updates to the parameters associated with the HMM that is used to derive the state sequence.

3.3 Datasets and Evaluation Metrics.

We use three datasets with conversation-level classification tasks to evaluate our model. The data statistics are summarized in Table 3.1.

CRAIGSLISTBARGAIN

The work by He et al. [39] presents a public negotiation dataset where buyers and sellers negotiate the prices of items on sale. In each conversation, the buyer has a target price in mind and attempts to reach an agreement with the seller. The negotiation scenarios are sourced from `https://craigslist.com`,

	CRAIGSLIST BARGAIN	ABCD	CALL CENTER
train set # dialogues	4828	8034	711310
dev. set # dialogues	561	1004	95540
test set # dialogues	567	1004	142560
# turns / dialogue	9.2	22.1	71.6
# tokens / turn	15.5	9.2	16.3
# tokens / dialogue	142.6	202.5	1167.1

Table 3.1: Data statistics of the datasets.

including a product description, optional product photos, and the listing price. The buyer is given a private target price that is strictly lower than the listing price. The data was collected on the Amazon Mechanical Turk (AMT) platform, where two Turkers role-played with each other. The seller aims to maximize profit, while the buyer tries to purchase the product at a price close to their private target price. Following previous work [131, 46], we use the same list of 14 handcrafted utterance DAs and the 5-class sale-to-list price ratio labels provided in their code base. The 14 handcrafted utterance DAs are used as comparison to evaluate if our unsupervised version of DAs is learning good representations. Classification of sale-to-list price ratio is used as the downstream task, with accuracy as the evaluation criterion. We follow all original data preprocessing scripts for CRAIGSLISTBARGAIN[4].

ABCD

The work [15] is a public customer support dataset that is introduced to study customer service dialogues. In each conversation, an agent follows guidelines to help a customer solve their issue. Conversations are categorized with flows and subflows. Flows are broad categories, such as shipping issue, account access, or purchase dispute. Subflows comprise 96 fine-grained labels, for example, shipping status question, recover password, or invalid promotion code. Each conversation is annotated with a flow and a subflow. We use classification of the subflows as our conversation-level task. Macro and micro F1 scores are used to reflect the performance of imbalanced subflow classes. We follow all original data preprocessing scripts for ABCD[5].

[4] https://github.com/rishabhjoshi/DialoGraph_ICLR21.
[5] https://github.com/asappresearch/abcd.

CALL CENTER

The dataset is a private collection of customer service conversations. Phone calls are automatically transcribed and private user information is anonymized. Conversations are annotated with a binary indicator as to whether or not there will be a callback within two days. (Such callbacks are an indicator that the problem was not solved in the call.) For the task of callback prediction, we measure area under the ROC curve (ROC AUC).

3.4 Experimental Results

3.4.1 Implementation Details & Experimental Setup

We pretrain and finetune on BERT [25] downloaded from `Huggingface` Transformers [113][6] and use the uncased base model of BERT in most of our experiments. To feed lengthy conversations to the model, we employ gradient checkpoint and `DeepSpeed` [87], a deep learning optimization library, to reduce GPU memory usage and accelerate the training process.

The details of the model hyperparameters are as follows. 1-layer and 2-head transformers with 300 hidden size are applied to encode sequences of utterance-level embeddings in text view and sequences of clusters and states in structure view. Thus, the total number of parameters of our best system THETA, including base model of BERT and 3 one-layer transformers, is about 113M. Models are selected by the best score on the development set for each dataset.

For in-domain adaptation pretraining (DAPT), we use dynamic whole-word masking (WWM) on 128-token segments for each dataset and use 5×10^{-5} as learning rate and 5000 steps for CRAIGSLISTBARGAIN and ABCD and 30000 steps for CALL CENTER. 0.1 epochs are used as warm-up steps with linear learning rate decay. Gradient accumulation and `PyTorch` [79] distributed data parallel GPU training are applied to achieve the equivalent training batch size 4096.

For finetuning, we set 1×10^{-5} as the learning rates, 4 epochs in total and 0.1 epochs for warm-up steps with linear decay. The equivalent training batch size is 16 during finetuning. Besides, the layer-wise learning rate decay is utilized to stabilize the training results; the rates are from $0.7, 0.8, 0.9$ and the 0.9

[6] `https://github.com/huggingface/transformers`

leads to the best performance. For the rest of the training hyperparameters, we follow the default values in `HuggingFace`'s training script.

For K-means, we use `Faiss` [45][7] with GPU to speed up clustering process for large private corpus. For HMMs, we develop our splitting algorithm via `Pomegranate` [93],[8] a Python package that implements fast and flexible probabilistic models, to build our topology learning algorithm. The predefined numbers of clusters (numbers of DAs) vary for different datasets. The size of the HMM state space are chosen separately for each dataset based on development set performance. To compare with handcrafted DAs provided in CRAIGSLISTBARGAIN, we define number of clusters $k = 14$ for each party. For customer service domain, we set $k = 60$ for ABCD and $k = 120$ for CALL CENTER. For all datasets, we try the number of states from 5 to 20 and find the best numbers of states are 8, 12, and 12 for CRAIGSLISTBARGAIN, ABCD, and CALL CENTER, respectively. Each training run takes at most 2 hours on 2 Nvidia GeForce RTX 2080Ti GPUs for CRAIGSLISTBARGAIN and ABCD and 54 hours on 8 GPUs on CALL CENTER. All models are saved based on the best performance on the development sets. For each experiment on CRAIGSLISTBARGAIN and ABCD, we conduct 15 random runs and report the median and standard deviation. Due to the computation limitations and the size of corpus, we only conduct a single run for CALL CENTER for each experiment setting. The total number of GPU hours for all experiments, including different runs with random seeds, is 1536 hours approximately.

3.4.2 Comparison Systems

We use the hierarchical encoder-only transformer (HET) as a baseline for all datasets in comparison to THETA. For CRAIGSLISTBARGAIN, we also include three additional baselines from two works [131, 46] that employ the DAs extracted by heuristic methods; our systems use K-means to obtain primitive DAs.

FST-enhanced hierarchical encoder-decoder model (FeHED)

FeHED [131] uses an RNN-based sequence-to-sequence model with finite-state transducers for encoding sequences of strategies and DAs.

[7] `https://github.com/facebookresearch/faiss`
[8] `https://github.com/jmschrei/pomegranate`

Hierarchical encoder-decoder (HED) + RNN or transformer

HED encodes dialogue utterances with a transformer (initialized from pretrained BERT), and the decoder generates the next response. An RNN or transformer encodes strategies and DAs. HED + RNN is based on the dialogue manager of He et al. [39]; Joshi et al. [46] replace the RNN with a transformer.

DIALOGRAPH [46]

The state-of-the-art HED-based model on CRAIGSLISTBARGAIN dataset leverages graph attention networks (GAT) [106] to encode strategies and DAs.

3.4.3 Prediction Results

Model	% Acc.
FeHED	42.3
HED + RNN	47.9
HED + transformer	53.7
DIALOGRAPH	53.1
HET	$54.1_{\pm 2.4}$
THETA	$\mathbf{66.1}_{\pm 1.0}$

Table 3.2: Results on the test set of CRAIGSLISTBARGAIN in accuracy. For models studied in this paper (lower part), the median number is reported with standard deviation calculated based on 15 random runs.

Performance on Negotiation Dialogues

Table 3.2 reports the results of different systems on the test set of CRAIGSLISTBARGAIN dataset. All models are based on the BERT-base model. HET with only text outperforms the state-of-the-art DIALOGRAPH which leverages a graph-based representation of conversation structure. This verifies our hypothesis that DAPT with target data indeed improves BERT for dialogue tasks. Compared with HET, THETA achieves better prediction accuracy and smaller variance, which suggests that integrating the structure view helps stabilize training with different random seeds. THETA provides a 24.5% relative gain in accuracy over DIALOGRAPH, setting a new state of the art. This further validates the advantage of our learned conversation structure for a predictive task.

	ABCD			CALL CENTER
	F1			
Model	Micro	Macro	Weighted	ROC AUC
HET	52.2	25.4	45.7	69.6
THETA	**62.8**	**39.1**	**59.9**	**71.3**

Table 3.3: Results on the test sets of ABCD and CALL CENTER datasets.

Performance on Customer Support Domain

Similar to the results on the negotiation dialogue domain, Table 3.3 shows that conversation structure effectively enhances the performance in the customer service domain, ABCD and CALL CENTER.

	CRAIGSLISTBARGAIN	ABCD			CALL CENTER
		F1			
Model	Accuracy	Micro	Macro	Weighted	ROC AUC
HET w/o DAPT	48.0	15.4	4.2	9.4	68.4
HET	50.3	52.2	26.9	46.3	71.2
THETA (cluster only)	60.2	59.8	35.3	55.7	72.2
THETA (state only)	51.7	58.8	32.8	54.1	72.1
THETA	**61.3**	**62.6**	**38.6**	**59.5**	**72.8**

Table 3.4: Ablation on the development sets of CRAIGSLISTBARGAIN, ABCD and CALL CENTER datasets. All models with structure are statistically better than HET. THETA is better ($p < 0.01$) than the cluster-only alternative except for the CALL CENTER.

Ablation

Table 3.4 reports the results of ablating different components of THETA on the validation sets of all datasets. The first rows show that DAPT is useful on all tasks particularly for ABCD with its skewed class distribution. We also observe that THETA consistently achieves the best performance over all tasks. The cluster-based DA sequence provides more information than the sub-dialogue states, but incorporating all three views together leads to the best performance. Statistical significance is tested using bootstrap resampling [27, 8].

Prior work [131, 46] on CRAIGSLISTBARGAIN use domain knowledge in rule-based annotation of DAs. To assess the use of K-means clusters for learning DAs, we also trained an HMM using the provided DAs. The resulting model obtained 66.5% accuracy on the test data, which is not significantly different the 66.1%

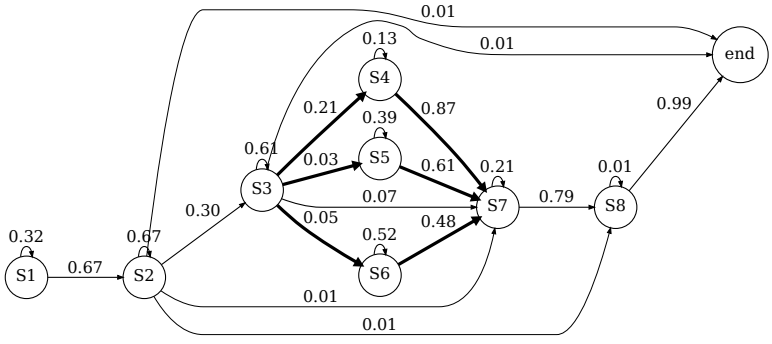
results obtained using K-means (cf. Table 3.2).

3.5 Interpretation and Analysis

In this subsection, we leverage automatic summarization of clusters and states to derive insights into the learned conversation structure, both for interpretability of the model and for applications such as conversation analytics and summarization. To achieve better interpretation, we use unsupervised summarization for generating human-readable summaries for different clusters and states to augment the learned corpus-level topology graph for analyzing dialogue patterns. As an example, we analyze fine-grained components from the learned topology, i.e., most frequent paths and individual state n-grams, to investigate their associations with different dialogue characteristics.

3.5.1 Topology Graph with Summary

Our hierarchical conversation structure produces a human-readable HMM topology graph for each dialogue corpus. Such human-readable topology graph helps reader comprehend the overall content of large amount of unlabeled conversation transcripts. It also provides a tool for interpreting the effects of common dialog patterns. For example, we can discover the conversational state leads to successful negotiations in trades or effective strategies experienced agents employ in customer support. We apply graph-based unsupervised summarization [10, 94] over utterances in each state (decoupling participant roles) and in each cluster. On CRAIGSLISTBARGAIN and ABCD, this leads to more than 3x reduction in conversation length.



S1	buyer: hi i am interested seller: hi how are you interested ?
S2	buyer: how about its condition ? seller: it s in great condition .
S3	buyer: i can do <price>. i can pick it up seller: that s too low. i can go with <price>
S5	buyer: i can do <price>. <offer> seller: the lowest i can do is <price>
S6	buyer: that s a deal. i can do that seller: great. that s a deal
S8	buyer: <accept> seller: <accept>

Figure 3.3: The 8-state topology graph on CRAIGSLISTBARGAIN dataset. The thicker edges indicate higher levels of negotiation success; in contrast, the thinner edges represent lower levels. The detailed topology graph with all cluster and state summaries is in Figure 3.4.

State Topology Graph and Path Analysis in Negotiation Dialogues (CRAIGSLISTBARGAIN)

Figure 3.3 is the simplified graph of Figure 3.4. For easier reading, we start from explaining the simplified graph and only show the state summary. We shows the detailed graph with cluster and state summaries in Figure 3.4.

The simplified graph Figure 3.3 shows the 8-state topology graph of CRAIGSLISTBARGAIN with selected state summaries. Based on the summaries, it is easy to see that `S1` and `S8` capture opening and closing DAs, respectively, while `S5` and `S6` correspond to different negotiation strategies. We also find that conversations with shorter paths are likely to involve a less experienced seller or lower buyer interest, e.g., 92% conversations with path `S1-S2-S8` lead to under listing sells. On the other hand, sellers that say offers are too low are more likely get better prices, e.g., 91% conversations with path `S1-S2-S3-S5-S7-S8`.

Figure 3.4 shows the detailed graph with both cluster and sub-dialogue state summaries. For each sub-dialogue state, we add the cluster summaries with top 3 emission probabilites and the sub-dialogue state summaries for the buyer and the seller. The thickness of edges indicates the levels of negotiation success and the edges with probabilities lower than 0.01 are pruned for simplicity.

Party	Utterance	Cluster Summary	State Summary
Agent	Welcome to AcmeBrands! How can I help you?	How can I help you?	A: How can I help you today? C: I want to check my order.
Customer	Hello, I would like to change my shipping details as they have changed recently due to a move	I want to check my shipping	
Agent	I would be happy to help you with that	I can help you with that	A: Can I have your account/order? C: My account/order is _____
Agent	Is there an outstanding order?	How long have you been waiting?	
Agent	Or is this just an update to your account?	I have pulled up your account.	
Customer	Yes my order id is 4870952797	My order/account ID is _____	
Agent	What is your name please?	What is your name?	
Customer	Crystal Minh	<name>	
Agent	What is the shipping status of the order?	What is the shipping status?	
Customer	In Transit	In store/ In transit	
Agent	Next I need to validate your purchase. I will need your username and email.	I need your name	
Customer	cminh948, cminh948@email.com	<email>	
Agent	Thank you	Thank you	
Agent	and the new address please?	Can you tell me _____?	A: Can I have the address? C: My address is _____
Customer	9756 Primrose Street Newark, MI 85971	<address>	
Agent	All taken care of!	Your order has been updated	
Agent	Is there anything else today?	Anything else?	A: Anything else I can help? C: That's all. Thank you.
Customer	Thank you that is all	That's all. Thank you.	
Agent	Have a great one!	Have a good one!	

Table 3.5: An example of ABCD with cluster and state summaries. A and C stand for agent and customer, respectively.

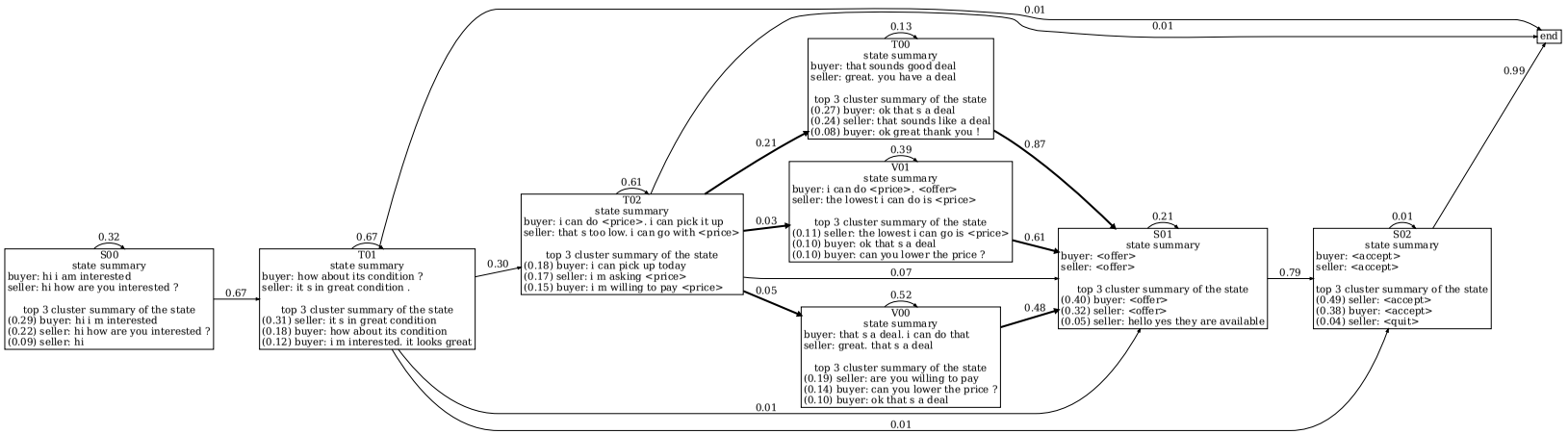


Figure 3.4: The 8-state full topology with cluster and sub-dialogue state summaries on CRAIGSLISTBARGAIN dataset.

Clustering and State Summaries in Customer Support Dialogue (ABCD)

The example in Table 3.5 shows the summaries of a conversation from cluster and state views in customer support domain. The state summaries provide a sketch of the conversation and cluster summaries render the template-like summaries. Based on the cluster summaries, we see that K-means learns typical DAs associated with customer service, e.g., information requests from the agent and the corresponding customer replies (`<name>, <email>, <address>`). States correspond to sub-dialogues where the agent follows certain protocols in resolving a subtask (e.g., verifying account information). Alignment of flow labels with the most frequent paths through the HMM topology graph shows that paths are highly indicative of the corresponding dialogue flow. The high confusions are among certain flows, such as `storewide_query` and `single_item_query`, which one would expect to have have similar DAs.

Most impactful features for class 4 (Good seller)
(seller: That's too low. I can go with <price>) $\longrightarrow$ (buyer: That's a deal)
(seller: That's too low. I can go with <price>)
(buyer: <offer>) $\longrightarrow$ (seller: <accept>)
(seller: That's a deal)
(seller: That's too low. I can go with <price>) $\longrightarrow$ (buyer: I can do <price>)

Most impactful features for class 0 (Rookie seller)
(buyer: how much do you want?)
(buyer: are you willing to take <price> ?)
(buyer: That's a deal) $\longrightarrow$ (seller: That's a deal)
(seller: It's in great condition)
(seller: It's in great condition. I am asking for <price>)

Table 3.6: Top 5 most impactful uni-state or bi-state features in XGBoost evaluated by SHAP values. Each feature is represented by its state summary.

Impact of Specific States in CRAIGSLISTBARGAIN

To identify sub-dialogue states linked to successful dialogues, we trained an XGBoost classifier on the CRAIGSLISTBARGAIN task using unigram and bigram features. SHAP values highlighted the most impor-

tant features. For predicting successful sellers (class 4), three of the top five features involved the seller stating the offer is too low. For predicting successful buyers (class 0), the top feature was the buyer asking, "how much do you want?" and two features involved the seller saying, "It's in great condition." When buyers ask for the seller's price, they control the negotiation, potentially putting sellers on the defensive and revealing their price first. Overemphasizing the item's condition might make buyers skeptical, causing them to test the seller's price flexibility, assuming the seller is compensating for other shortcomings.

Path Analysis in CALL CENTER Dataset

Figure 3.5 shows the relative frequency of customer call backs for different paths through the graph. A

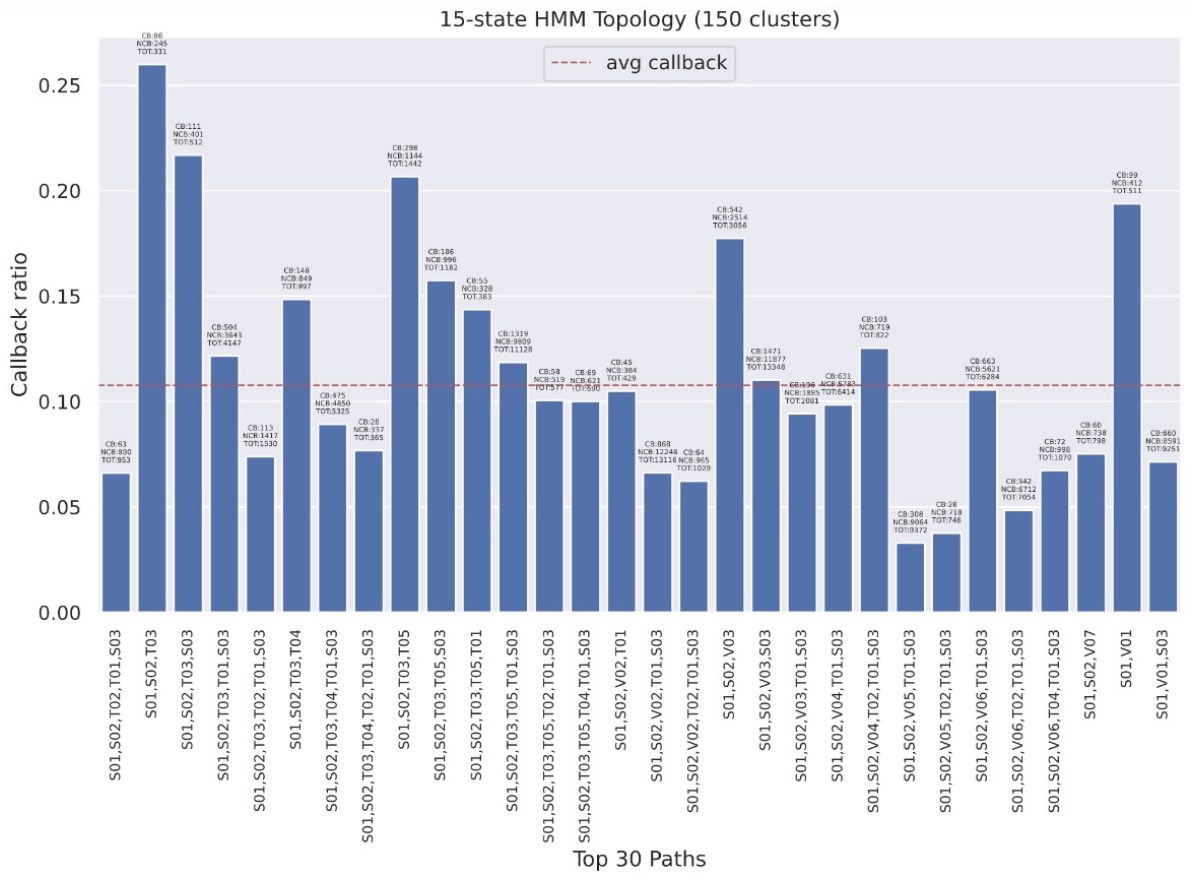


Figure 3.5: Top 30 most frequent paths and their callback ratios extracted from the 15-state HMM graph of the CALL CENTER dataset. A higher ratio indicates a more problematic path where agents fail to resolve the user's issue, leading to an increased likelihood of a callback. Conversely, a lower ratio suggests a path with a reduced chance of a callback.

lower callback frequency corresponds to more successful interactions. We randomly select two paths to

analyze: S01,S02,T03,S03 (high callback ratio) and S01,S02, V05,T02,T01,S03 (low callback ratio). The shared states S01, S02, S03 between the two paths represent greeting, verification, and closing, respectively. These states are common across most calls and provide minimal information for distinguishing between different calls. In contrast, states T03, V05, T02, T01 highlight the differences between paths. For example, in paths with a high callback ratio, T03 is identified as problematic, involving account access issues that require full identification before assistance can be provided; hence, automating user identification before transferring to an agent could enhance efficiency. Conversely, for paths with a low callback ratio, the graph reveals successful strategies employed by agents. The strategy can be extracted and documented for the other agents to follow in the future. For instance, V05 pertains to payment issues, where agents provide solutions and update payment information in T02. In T01, the agent summarizes the resolved issues, and the user expresses appreciation.

3.6 Discussion & Limitations

Methodological Limitations and Future Directions

We use K-means and HMMs for deriving the conversation structure, both of which require dataset-specific hyperparameters that are unlikely to transfer well to new datasets. Additionally, we only study a late fusion strategy for combining discrete structure and text-based representations. A more tightly integrated approach might be more effective. For example, our K-means DA is based on a single utterance; however, sequence models have been important for past work on unsupervised learning of DAs.

Direct Evaluation on Graph Quality

In this chapter, we indirectly evaluate the graph quality by utilizing cluster and state sequences as additional features to enhance model performance on predictive tasks. A more direct evaluation of the graph might provide a better reflection of the graph summary quality.

3.7 Summary

In this chapter, we introduced a novel concept of corpus-level, graph-based summarization, using a graph to represent various types of dialogues within an entire corpus. The graph is constructed by combining two unsupervised learning approaches: K-means clustering and HMM topology design, based on embedded utterance representations. We explicitly choose to use discrete representations of latent structure, with the goal of using automatic summarization to make the structure interpretable. The K-means clusters are intended to approximate DAs and the HMM is intended to learn sub-dialogue structure. In addition, this hierarchical representation of conversation structure enhances the performance of a hierarchical transformer in three conversation-level classification tasks. Unlike prior work in this area, the sub-dialogues build on DA sequences rather than unigram/bigram statistics, and the HMM incorporates forward-moving dialogue flow constraints in topology learning, with the goal of capturing sub-dialogue function.

Chapter 4

Structure-aware Efficient Encoder-Decoder Transformer Decoding

4.1 Introduction

The transformer architecture [105] is the backbone of many successful NLP models, but they have high latency and computational costs. To reduce costs, researchers have investigated multiple approaches, including: i) model compression (e.g., distillation [41, 32, 104], quantization [122, 23, 115, 24, 126], and mixture of experts [51]); ii) decoding strategy (e.g., speculative decoding [56, 14] and parallel decoding [91, 73]); iii) algorithm-level attention optimizations (e.g., sparse attention [90, 64], fewer number of key-value heads [97, 2] and Hydragen [48]); iv) kernel-level attention optimizations (e.g., FlashAttention [21] and FlashInfer [119]); and v) memory management (e.g. paged attention[52] and radix attention [128]). Our proposed method falls into the category of algorithm-level attention optimizations via key-value prefix sharing. We aim to upcycle existing encoder-decoder models with the goal of reducing redundant computations and increasing hardware utilization via key-value prefix sharing.

Recent research [109] indicates that scaling up decoder-only transformer models facilitates effective in-context learning, which is particularly beneficial when labeled data is scarce, making the models more generalized to unseen domains. However, these larger decoder-only models, despite their generalizability to unseen domains, incur higher computational costs and may offer inferior performance compared to smaller,

specialized encoder-decoder transformer models [55]. Motivated by this, we propose a new configuration for encoder-decoder models to further improve training and inference efficiency in *decomposable* tasks that involve multiple prompts over the same document (or dialogue). These tasks include scenarios where multiple users are querying the same document with different requests (e.g., question answering, [120, 48]), as well as scenarios where it is useful to decompose a complex task into simpler subtasks (e.g., abstractive summarization of long/multiple documents/dialogues [31, 70, 124, 121] and dialogue information extraction [68]).

In the multi-user question-answering scenario, decoupling questions and the corresponding document allows reusing the shared document embeddings for questions from different users. The shared embeddings can significantly reduce duplicate computation of the shared prefixes and increase operational intensity during training and inference. For summarization and information extraction, decomposing a long target output into multiple shorter sequences mitigates attention degeneration issues [29, 130], leading to a boost in accuracy and efficiency. However, existing methods put the prompts in the encoder, resulting in a high computation cost because contextualizing each shared document and its prompts results in the duplicate encoding of the same shared prefixes. Instead, we propose **prompt-in-decoder (PID)**: an encode-once, decode-in-parallel strategy that avoids duplicate encoding costs by sharing inputs and increases decoding efficiency by reducing memory access.

In this chapter, we demonstrate the effectiveness of our structure-aware decoding strategy through experiments on two conversation-level dialogue summarization tasks: dialogue state tracking and abstractive medical dialogue summarization. Furthermore, we show that our proposed method generalizes well to extractive medical question answering on medical notes. Our models achieve comparable or higher performance (98-101%) than the current state of the art. At the same time, we observe a 2-10x computation reduction, depending on the number of subtasks, and up to 4.6x speed-up for shorter subtask outputs.

This work has been published on arXiv and has been submitted to a peer-reviewed conference. As of the writing of this thesis, it is currently under review.

4.2 Encoder-Decoder Framework

As described in subsection 2.1.2, the encoder-decoder is a general framework that has been used to address a wide variety of problems in NLP. Given an input word sequence, a desired result is obtained by first encoding the input and then iteratively generating an output word sequence. Mathematically, in general-purpose models, the input $\mathcal{X}$ is optionally combined with a prompt[1] $\mathcal{Z}$ that specifies the task:

$$\mathcal{Y} = \text{decoder}(\text{encoder}(\mathcal{X}, \mathcal{Z})).$$

The input $\mathcal{X}$ is any form of text, e.g., a sentence, article, or transcript of a conversation, and $\mathcal{Z}$ can be an instruction or a question. The output $\mathcal{Y}$ could be information extracted from an article, a summary of a conversation, or a response to a question.

State-of-the-art encoder-decoder systems are built on transformers. This section overviews the general framework to introduce notation and set up the multi-subtask inference problem that we address.

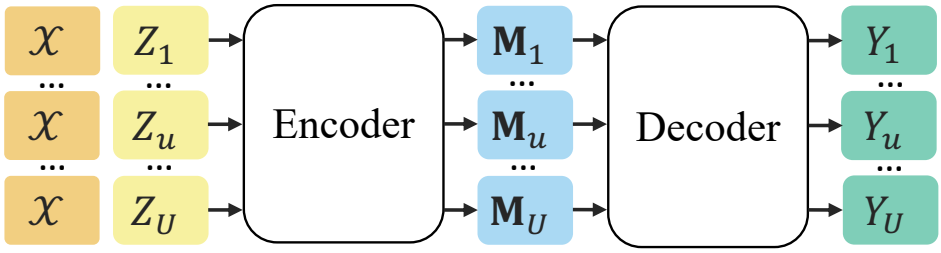
4.2.1 Multi-Prompt Decoding

In this paper, we tackle tasks that can be framed in terms of multiple prompts over the same input $\mathcal{X}$. Specifically, the output is a list of subtasks (or answers) $\mathcal{Y}$, where each subtask/answer is Y_u, e.g., $\mathcal{Y} = (Y_1, \ldots, Y_u, \ldots, Y_U)$, and U is the total number of subtasks/answers. Each subtask Y_u is associated with a specific prompt Z_u, so $\mathcal{Z} = (Z_1, \ldots, Z_u, \ldots, Z_U)$. The scenario involves running inference multiple times to generate Y_u:

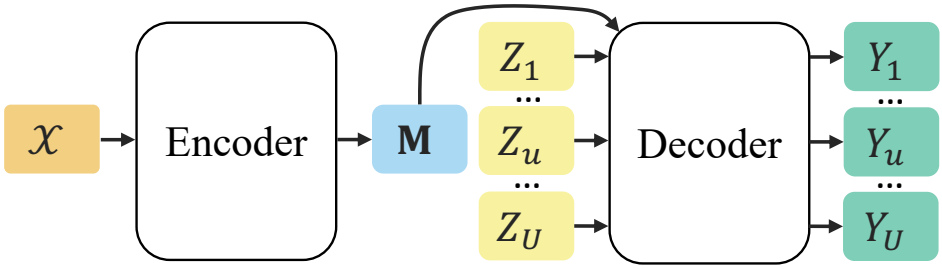
$$Y_u = \text{decoder}(\text{encoder}(\mathcal{X}, Z_u)),$$

then combining all outputs Y_u to form the final $\mathcal{Y}$. We refer to this as the **prompt-in-encoder (PIE)** approach. Figure 4.1a illustrates how PIE tackles a single instance ($\mathcal{X}$, $\mathcal{Y}$ and $\mathcal{Z}$). PIE involves U encodings of $\mathcal{X}$, one for each prompt Z_u.

[1] We use prompt to avoid confusion between the "query" and the "query vector" in subsection 4.3.2 and subsection 4.3.3.



(a) Prompt-in-encoder (PIE).



(b) (Ours) Prompt-in-decoder (PID).

Figure 4.1: Given a task where a single input document $\mathcal{X}$ is used to generate multiple outputs Y_u associated with different prompts Z_u, PIE creates unique encodings $\mathbf{M}_u$ for every prompt Z_u. In contrast, PID uses a single shared $\mathbf{M}$ for each prompt. thus requiring less memory access and resulting in higher computational efficiency.

4.2.2 Encode Once and Decode in Parallel

To avoid the redundant encoding of $\mathcal{X}$ in PIE and improve inference efficiency when decoding Y_u, we propose placing prompt Z_u in the decoder, allowing us to encode $\mathcal{X}$ once and decode Y_u in parallel. We refer to this method as **prompt-in-decoder (PID)**. By moving Z_u from the encoder (in PIE) to the decoder, the encoder only encodes $\mathcal{X}$ once, generating a single sequence of embeddings that is reused throughout the decoding process for each prompt Z_u. As shown in Figure 4.1b, $\mathcal{X}$ is only encoded once, and the embeddings $\mathbf{M}$ are reused for U prompts during decoding to generate all subtask outputs $(Y_1, \ldots, Y_u, \ldots, Y_U)$. Formally, the equation can be represented as

$$Y_u = \text{decoder}(\text{encoder}(\mathcal{X}), Z_u).$$

4.3 Performance Analysis

In this section, we show how the PID model improves inference efficiency over the PIE model by quantifying memory access and the number of arithmetic operations.

4.3.1 Operational Intensity

Memory bandwidth peak performance per second (byte/s) and the number of floating-point operations per second (FLOP/s) are used to compute the operational intensity:

$$\text{Operational Intensity} = \frac{\text{FLOP/s}}{\text{byte/s}} = \frac{\text{\# operations}}{\text{memory access}},$$

which provides a measure for the hardware efficiency of operations [112].

To carry out calculations, accelerators must access and move data between global memory and registers, which can be a bottleneck because modern hardware accelerators such as GPUs/TPUs often have significantly greater capacity for computations compared to memory bandwidth. For example, an NVIDIA A100 GPU [16] has an operation capacity of 312 Tera FLOP/s versus a memory bandwidth of 2 Giga byte/s. The attainable FLOP/s of a device is determined by the formula:

$$\text{Attainable FLOP/s} = \min(\text{Peak FLOP/s}, \underbrace{\text{Operational Intensity}}_{\text{\# operations per byte}} \cdot \underbrace{\text{Peak Memory Bandwidth}}_{\text{bytes per second}}),$$

where the peak memory bandwidth is fixed and the peak FLOP/s is the maximum arithmetic operations the accelerator is capable of performing. If the operational intensity is too low, the accelerator idles, waiting for data to move to registers instead of running computations. This often occurs in models where memory access is more intensive than arithmetic operations, i.e., incremental decoding in transformers [97]. By decreasing memory access, the operational intensity is increased, i.e., efficiency improves.

4.3.2 Multi-head Attention in Transformer

Transformers [105] have two types of multi-head attention: self-attention and cross-attention. Usually, the attention key and value tensors have the dimension d/h, where d is the dimension of input and output vectors

and h is the number of attention heads. For simplicity, we consider the operations of all heads together such that the dimension of $\mathbf{Q}$, $\mathbf{K}$, $\mathbf{V}$ (query/key/value) in the following section is denoted as d.

During attention, the query/key/value vectors can be obtained by projecting the corresponding input vectors $\mathbf{N} \in \mathbb{R}^{n \times d}$ or $\mathbf{M} \in \mathbb{R}^{m \times d}$, where n and m can be either n_s (input source length) or n_t (output target length). More formally, the equations are $\mathbf{Q} = \mathbf{N} \cdot W^Q \in \mathbb{R}^{n \times d}$, $\mathbf{K} = \mathbf{M} \cdot W^K \in \mathbb{R}^{m \times d}$ and $\mathbf{V} = \mathbf{M} \cdot W^V \in \mathbb{R}^{m \times d}$ where the projection matrices are $W^Q, W^K, W^V \in \mathbb{R}^{d \times d}$.

In the self-attention, $\mathbf{N}$ is equivalent to $\mathbf{M}$; hence, $m = n = n_s$ in the encoder or $m = n = n_t$ in the decoder. On the other hand, in the case of the cross-attention, the variable $\mathbf{N} \in \mathbb{R}^{n_t \times d}$ originates from the decoder, while $\mathbf{M} \in \mathbb{R}^{n_s \times d}$ is sourced from the output of the encoder.

The simplified equation of the attention mechanism is represented as follows,

$$\mathbf{O} = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d/h}}\right)\mathbf{V} \cdot W^O, \tag{4.1}$$

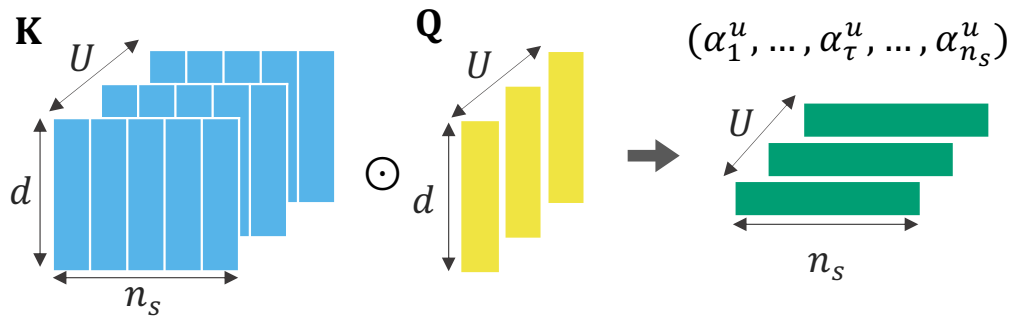
where $\mathbf{O} \in \mathbb{R}^{m \times d}$ is the final atention output.

4.3.3 Performance Analysis for PIE and PID

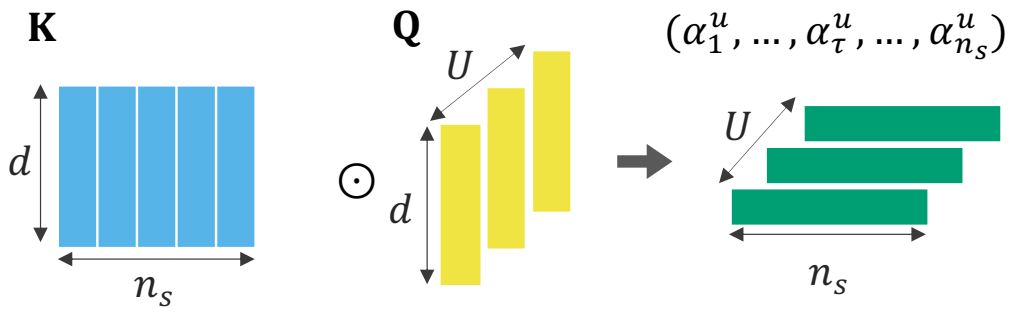
Figure 4.2 shows the dot product operation in the cross-attention for the two models. In the PIE model, the input is contextualized with each prompt, so the encoder's output tensor differs for each prompt. Thus, at each decoding step τ, $\mathbf{K}$ is read U times to generate U different sets of attention weights α_τ^u to decode Y_u. In contrast, in the PID model, $\mathbf{K}$ is shared across all prompts since the input is encoded independently of the prompts. Thus, at each decoding step τ, the dot product operation in the cross-attention shares and broadcasts $\mathbf{K}$ and computes the dot product of $\mathbf{K}$ and $\mathbf{Q}$, resulting in lower memory access but the same number of arithmetic operations compared to PIE.

	Encoder's Self-attention		Decoder's Self-attention		Decoder's Cross-attention	
Model	Memory	Operations	Memory	Operations	Memory	Operations
PIE-T5	$Ubn_sd + d^2$	Ubn_sd^2	$Ubn_t^2d + n_td^2$	Ubn_td^2	$Ubn_sn_td + Ubn_td + n_td^2$	Ubn_td^2
PID-T5	$bn_sd + d^2$	bn_sd^2	$Ubn_t^2d + n_td^2$	Ubn_td^2	$bn_sn_td + Ubn_td + n_td^2$	Ubn_td^2

Table 4.1: Inference computation comparison between PIE and PID, where U, b, n_s, n_t, d are the number of prompts, batch size, input source length, output target length, and hidden size, respectively.



(a) Cross-attention of PIE.



(b) Cross-attention of PID.

Figure 4.2: An illustration of cross-attention dot product operations ($\mathbf{QK}^\top$ in Equation 4.1) for PIE and PID for a single inference step. U, d, n_s are the number of prompts, hidden layer dimension, and input length, respectively. $\odot$ is the dot product operation, and the resulting scalars of $\mathbf{QK}^\top$ are α_τ^u, where $\tau = \{1, \ldots, n_s\}$, at the decoding step τ with respect to the prompt Z_u.

We approximate the memory access and operations based on the dominant terms in the self- and cross-attention and ignore the constant terms. For a single input, the memory access of $\mathbf{M}$ and $\mathbf{N}$ is $n_s d$ and $n_t d$. The memory access of the matrices W^Q, W^K, W^V, W^O is d^2. The number of operations in both attention mechanisms is dominated by the matrix projections which are used to obtain $\mathbf{Q}$, $\mathbf{K}$, $\mathbf{V}$, and $\mathbf{O}$; thus, the number of operations is approximated as $n_s d^2$ or $n_t d^2$.

The comparisons of memory access and operation counts on different inference components for our PIE and PID implementations are presented in Table 4.1, explained in further detail below. In the analysis, we assume that a batch of b inputs with the same set of U subtasks are processed together. The encoder input length and the decoder output length differ depending on whether the prompt is in the decoder. To present a higher-level overview of the analysis, we begin by simplifying the analysis in this section, assuming that the prompt terms are negligible. Following this, in section 4.4, we provide a detailed justification.

Encoder's self-attention. In PIE, to run inference on a single instance, the model encodes U prompts (Z_u) with input ($\mathcal{X}$). Considering a batch with b instances, PIE takes $Ubn_sd + d^2$ for memory access and Ubn_sd^2 for the number of operations. In contrast, PID only encodes the input once, so the memory access and the number of operations remain $bn_sd + d^2$ and bn_sd^2. Thus, the encoders of both models have similar operational intensity, but PIE requires more memory access and more arithmetic computations.

Decoder's self-attention. In both PIE and PID, the decoder computes the self-attention for U prompts. For each decoding step, the memory access and the number of operations are Ubn_td and Ubd^2. Thus, for n_t steps, the resulting memory access and the number of operations are Ubn_t^2d and Ubn_td^2. In this case, PIE and PID have roughly the same operational intensity.

Decoder's cross-attention. Cross-attention dominates the inference cost. For each decoding step, we load the encoded input embeddings $\mathbf{M} \in \mathbb{R}^{m \times d}$, where $m = n_s$, from the encoder for each $(\mathcal{X}, Z_u)$ input for PIE and $\mathcal{X}$ for PID. For PIE, the resulting b $(\mathbf{M}_1, \ldots, \mathbf{M}_U)$ is fed into the decoder's cross-attention for b instances with U prompts. On the other hand, PID shares all $\mathbf{M}$ for all U prompts of each instance, resulting in feeding b $\mathbf{M}$ to the decoder. Therefore, for each step, the memory access for loading encoded $\mathbf{M}$ is Ubn_sd and bn_sd for PIE and PID, respectively. The memory access of loading $\mathbf{Q}$ and $\mathbf{O}$ is Ubd. Loading projection matrices takes d^2. We multiply all memory access cost by a factor of n_t steps.

4.4 Detailed Performance Analysis

In the previous section, we provided a high-level overview of the performance analysis, disregarding the prompt length. In this section, we delve into a detailed performance analysis, taking the effect of prompt length into account and comparing the operational intensity ratios of PIE and PID.

To simplify operational intensity equations, we follow the previous work [97, 22, 2], using the *inverse operational intensity* to compare the inverse ratios of all modules in encoder-decoder models.

Lower inverse ratio (higher operational intensity) brings more efficient matrix computation in modern accelerators such GPUs/TPUs, hence better performance. We discuss memory access and number of floating-point operations of encoder's self-attention, decoder's self-attention and decoder's cross-attention

for PIE and PID, respectively. We denote n_s, n_t and n_p as input source length, output target length and prompt length. U is number of prompts/subtasks. d is the joint dimension of all heads of key/query/value vectors. We approximate the memory access and the number of operations based on the dominant terms in the self- and cross-attention and ignore the constant terms. For a single input, the memory access of the key or value tensors $\mathbf{M}$ or $\mathbf{N}$ are $n_s d$ and $n_t d$. The memory access of the projection matrices of key/query/-value/output tensors W^K, W^Q, W^V, W^O is d^2. As described in Shazeer [97], the number of operations in both attention mechanisms is dominated by the matrix projections which are used to obtain projected query/key/value/output tensors (**Q/K/V/O**); thus, the number of operations is approximated as $n_s d^2$ or $n_t d^2$.

We denote inverse operational intensity as $\mathcal{R}^{\text{module}}_{\text{config}}$ in the following paragraphs. The module can be one of the following types: encoder's self-attention (Enc-self), decoder's self-attention (Dec-self), decoder's self-attention within prompts (Dec-self, prompt), or decoder's self-attention within generated tokens (Dec-self, gen-tkn). Additionally, the configuration (config) can be either PIE or PID.

4.4.1 Encoder's self-attention

The inverse operational intensity of PIE's encoder can be written as follows,

$$\begin{aligned}\mathcal{R}^{\text{Enc-self}}_{\text{PIE}} =& \underbrace{Ub(n_s+n_p)d + d^2}_{\text{memory access}} \Big/ \underbrace{Ub(n_s+n_p)d^2}_{\text{\# operations}} \\ =& \frac{1}{d} + \frac{1}{Ub(n_s+n_p)},\end{aligned}$$

where PIE's encoder individually encodes U prompts $(Z_1, \ldots, Z_U)$ with input $\mathcal{X}$. $\mathcal{R}^{\text{Enc-self}}_{\text{PIE}}$ is a low ratio given the fact that n_s is usually an hundred or a thousand tokens and d is usually near a thousand. Different from PIE's encoder, PID's encoder only encodes input $\mathcal{X}$ and leaves prompts in the decoder. Thus, the memory access is lower than PIE by a factor of U and only the input length n_s is considered. More formally

the inverse operational intensity of PID's encoder is denoted as

$$\begin{aligned}\mathcal{R}_{\text{PID}}^{\text{Enc-self}} &= \underbrace{bn_s d + d^2}_{\text{memory access}} \Big/ \underbrace{bn_s d^2}_{\text{\# operations}} \\ &= \frac{1}{d} + \frac{1}{bn_s}.\end{aligned}$$

Again, n_s and d is around a thousand, resulting in $\mathcal{R}_{\text{PID}}^{\text{Enc-self}}$ is also a low ratio. Compared to PIE, PID requires fewer number of operations, saving more memory usage and enabling faster computation.

4.4.2 Decoder's self-attention

In PIE, prompts are encoded in the encoder, the decoder only accounts for generating target tokens. The inverse operational intensity of PIE encoder's self-attention is denoted as

$$\begin{aligned}\mathcal{R}_{\text{PIE}}^{\text{Dec-self,gen-tkn}} &= \underbrace{Ubn_t^2 d + n_t d^2}_{\text{memory access}} \Big/ \underbrace{Ubn_t d^2}_{\text{\# operations}} \\ &= \underbrace{\frac{n_t}{d}}_{\text{dominant term}} + \frac{1}{Ub},\end{aligned}$$

where $\frac{n_t}{d}$ is the dominant term that causes the issue of slower incremental decoding.

In PID, the decoder encodes all tokens of a prompt simultaneously and incrementally generates output tokens. Thus, we decouple the analysis of encoding the tokens in the prompt and generating new output tokens. The ratio of encoding a prompt can be written as

$$\begin{aligned}\mathcal{R}_{\text{PID}}^{\text{Dec-self,prompt}} &= \underbrace{Ubn_p d + d^2}_{\text{memory access}} \Big/ \underbrace{Ubn_p d^2}_{\text{\# operations}} \\ &= \frac{1}{d} + \frac{1}{Ubn_p}.\end{aligned}$$

Obviously, the $\mathcal{R}_{\text{PID}}^{\text{Dec-self,prompt}}$ is a low ratio (high operational intensity); thus the encoding prompts in the decoder part is efficient. In addition to encoding prompts, PID's decoder also needs to generate output tokens. The ratio of generating tokens of in PID ($\mathcal{R}_{\text{PID}}^{\text{Dec-self,gen-tkn}}$) is the same as $\mathcal{R}_{\text{PIE}}^{\text{Dec-self,gen-tkn}}$.

4.4.3 Decoder's cross-attention

In encoder-decoder transformer architecture, the decoder's cross-attention is the key issue that cause the computation inefficiency since the inference is incremental and cross-attention needs to read the huge chunk of key and value cached tensors from the encoder. The inverse operational intensity of PIE decoder's cross-attention can be denoted as follows,

$$\begin{aligned}\mathcal{R}_{\text{PIE}}^{\text{Dec-cross}} &= \frac{\overbrace{Ub(n_s+n_p)n_td + Ubn_td + n_td^2}^{\text{memory access}}}{\underbrace{Ubn_td^2}_{\text{\# operations}}} \\ &= \underbrace{\frac{n_s+n_p+1}{d}}_{\text{dominant term}} + \frac{1}{Ub},\end{aligned}$$

where a prompt is encoded in the encoder; hence the length of cached tensors is the sum of input source length and the prompt length ($n_s + n_p$). The dominant term results in a serious bottleneck especially when long input n_s is fed into the model. Similar to the analysis of PID decoder's self-attention, we consider encoding tokens in a prompt and generating new output tokens separately in PID's decoder's cross-attention. The ratio for encoding tokens in the prompt is as follows,

$$\begin{aligned}\mathcal{R}_{\text{PID}}^{\text{Dec-cross,prompt}} &= \frac{\overbrace{bn_sd + Ubn_pd + d^2}^{\text{memory access}}}{\underbrace{Ubn_pd^2}_{\text{\# operations}}} \\ &= \underbrace{\frac{1}{d}\cdot\left(\frac{n_s}{Un_p}+1\right)}_{\text{dominant term}} + \frac{1}{Ubn_p}.\end{aligned}$$

In the dominant term, the input source length n_s is divided by the factor of Un_p. On the other hand, the ratio of generating output target tokens is denoted as,

$$\mathcal{R}_{\text{PID}}^{\text{Dec-cross,gen-tkn}} = \frac{\overbrace{bn_s n_t d + Ubn_t d + n_t d^2}^{\text{memory access}}}{\underbrace{Ubn_t d^2}_{\text{\# operations}}} = \underbrace{\frac{1}{d} \cdot \left(\frac{n_s}{U} + 1\right)}_{\text{dominant term}} + \frac{1}{Ub}.$$

Similarly, in the dominant term, the input source length n_s is divided by the factor of U. Overall, in PID decoder's cross-attention, the dominant term of the incremental decoding is reduced by a factor of U or Un_p since PID shares the same input key-value cache, only move the cached tensor once and broadcast the matrix operations while performing the cross-attention for generating individual tokens of all subtasks.

4.5 Datasets & Metrics

4.5.1 Datasets & Task Performance Metrics

	MultiWoZ 2.4	ACI-Bench	RadQA
Data	Task oriented	Medical	Medical
Input type	Dialogue	Dialogue	Document
Task	Dialog state tracking	Summarization	Question Answering
# examples	9887	207	6148
Input length	$289_{\pm 108}$	$1725_{\pm 511}$	$137_{\pm 157}$
Output length	$56_{\pm 26}$	$693_{\pm 200}$	$28_{\pm 49}$
Prompt type	Fixed	Fixed	Free-form
# prompts	30 slots or 5 domains	4 sections	–

Table 4.2: Statistics are calculated on the each full data set. Input and output lengths are calculated based on Huggingface T5 tokenizer.

In terms of data preprocessing, we follow the previous works [116, 121, 55] to process MultiWoZ 2.4, ACI-Bench and RadQA, respectively. The dataset statistics are shown in Table 4.2. We describe each dataset and its task performance as follows.

Dialogue State Tracking (DST).

Multi-domain Wizard-of-Oz dataset (MultiWoZ; 13) is a task-oriented dialogue dataset. We selected the most recent version of MultiWoZ 2.4 [117] due to its refined validation and test set annotations. MultiWoZ 2.4, designed for task-oriented dialogue, involves dialog state tracking with 9,887 examples. The input and output lengths average 289 ± 108 and 56 ± 26 tokens, respectively, with prompts being fixed across 30 slots or 5 domains. For comparison to other work, joint goal accuracy (JGA) is adopted as the evaluation metric. The input $\mathcal{X}$ is the dialogue history, $\mathcal{Y}$ is the dialogue state, the subtask prompts Z_u are the domain-slot name, and the associated outputs Y_u are slot values.

Summarization.

We use ACI-Bench [121], a dataset containing clinical notes associated with conversations between doctors and patients. The clinical notes have structured output with distinct sections. ACI-Bench, focused on medical dialogue summarization tasks, contains 207 examples with significantly longer inputs and outputs, averaging 1,725 ± 511 and 693 ± 200 tokens, respectively, and uses fixed prompts categorized into four sections. We use ROUGE-L score [61], denoted as R-L, to evaluate models. The input $\mathcal{X}$ is the doctor-patient dialogue, $\mathcal{Y}$ is the full clinical note, the subtask prompts Z_u are section indicators, and the associated outputs Y_u are section notes.

Question Answering.

RadQA [98] is an extractive question-answering dataset on radiology reports with 3k questions posed by experts. A single report can have multiple questions. RadQA includes 6,148 examples, with input and output lengths averaging 137 ± 157 and 28 ± 49 tokens, respectively, employing a free-form prompt type We use exact match (EM) as our evaluation metric. The input $\mathcal{X}$ is the radiology report, the subtask prompts Z_u are specific questions, and the associated outputs Y_u are extracted responses.

4.5.2 Efficiency Metrics

Floating point operations (FLOPs)

refer to the number of arithmetic operations required for model inference, i.e., the computational complex-

ity.[2] Note that FLOP reduction may not correlate with wall-clock speed, as it tends to ignore overheads from memory access (IO) [21].

Latency.

To account for extra IO costs, e.g., GPU memory bandwidth, we also report latency, which measures the wall clock time for a single instance inference. Specifically, we report the average time for a single instance inference, where instances are processed sequentially.

Latency w/ Batching.

For real applications, multiple instances are computed in a batch fashion to fully utilize the computing device, especially for cloud serving. We thus report the average time for a single instance, where a batch of instances is computed simultaneously.

To assess the FLOPs and latency, we randomly chose 512 samples from MultiWoZ 2.4 test set (due to the large test set) and used the full test sets for ACI-Bench and RadQA. We report the average latency and the average latency w/ batching at the optimal batch size.

4.6 Experimental Results

	MultiWoZ 2.4		ACI-Bench		RadQA	
	PID-T5$_{\text{base}}$	PID-T5$_{\text{large}}$	PID-T5$_{\text{base}}$	PID-T5$_{\text{large}}$	PID-T5$_{\text{base}}$	PID-T5$_{\text{large}}$
Batch size	4	1	1	2	4	2
Grdient accumulation	64	32	16	16	16	32
Effective batch size	64	64	32	32	64	64
# epochs	6	4	100	100	50	15
Max input length	1024	1024	3072	3072	1024	1024
Max output length	24	24	1024	1024	92	92
Max prompt length	8	8	6	6	36	36
# outputs	30	30	4	4	2-6	2-6
# beams	1	1	4	4	1	1

Table 4.3: The PID-T5 hyperparameters used in the training and testing.

[2] We use `calflops` [118] to compute FLOPs.

4.6.1 Training Details

Table 4.3 presents the hyperparameters selected for the training and testing phases. We executed a search for the optimal learning rate across the following set of values: $\{5\times10^{-4}, 3\times10^{-4}, 1\times10^{-4}, 7\times10^{-5}, 5\times 10^{-5}, 3\times10^{-5}\}$ to identify the most effective learning rate for each dataset. All reported values represent the medians of three different random runs. The rest of the hyperparameters are the same as the default values in HuggingFace transformer package. Training time varies because of dataset size, model size, model configuration and training procedure. Our experiments, which utilize $T5_{base}$ as the primary framework, are carried out using a single NVIDIA A40. Training $T5_{base}$ and PID-$T5_{base}$ on the MultiWoZ 2.4 dataset typically requires approximately 5 GPU hours, whereas it takes around 46 GPU hours for PIE-$T5_{base}$. In the case of ACI-Bench, where the dataset is relatively small, $T5_{base}$, PIE-$T5_{base}$, and PID-$T5_{base}$ require roughly 4 GPU hours each. On the other hand, for RadQA, PIE-$T5_{base}$ takes 2 hours, while PID-$T5_{base}$ requires 3 GPU hours. When switching to $T5_{large}$, the required GPU training time increases by a factor of 2 to 3 times compared to the $T5_{base}$ models.

We use t5-base[3] and t5-large[4] checkpoints downloaded from HuggingFace as initialization for MultiWoZ 2.4 and ACI-Bench. For RadQA, we follow the previous work [55] to use pretrained clinical T5 models.[5]

4.6.2 Compared Systems

T5

T5 [84] is adopted as the encoder-decoder model in all experiments. We use T5-base and T5-large models from HuggingFace for MultiWoZ 2.4 and ACI-Bench. For RadQA, we follow the previous work [55] and use a pretrained clinical T5. We denote base and large models in following tables as $T5_{base}$ and $T5_{large}$. We use T5 as the backbone to compare two different subtasking strategies: prompt-in-encoder (PIE) and our proposed prompt-in-decoder (PID); thus the model size keeps the same as the standard T5. The sizes of $T5_{base}$ and $T5_{large}$ are 220M and 770M, respectively.

[3] https://huggingface.co/google-t5/t5-base
[4] https://huggingface.co/google-t5/t5-large
[5] https://physionet.org/content/clinical-t5/1.0.0/

LLaMA2

LLaMA2 [102] is a popular open decoder-only language model. Due to computation limitations, we adopt low-rank adaptation (LoRA) [42] to efficiently finetune LLaMA2 7B, resulting in approximately 80M trainable parameters with a rank of 64.

Current State-of-the-art Models

We report the current best published results for in-context learning and full finetuning for every dataset. The source papers for the results of each dataset are documented in the caption of Table 4.6.

4.6.3 Training Procedure

The standard finetuned T5 and LLaMA2 data is represented as $(\mathcal{X}, \mathcal{Y})$, i.e. no prompts are used. For PIE-T5, the data is $(\mathcal{X}, Y_u, Z_u)$, since $\mathcal{X}$ is separately contextualized with each subtask prompt Z_u and a subtask output Y_u, effectively increasing the dataset size by a factor of U and substantially extending the time required for training. In contrast, PID-T5 has the flexibility to be trained using either data representation, as it uses shared inputs. We choose $(\mathcal{X}, \mathcal{Y}, \mathcal{Z})$ for PID-T5 because it is more efficient to put all output $\mathcal{Y}$ from the same shared input $\mathcal{X}$ in the same batch to save encoding processing time during gradient update. The model selection criterion is the highest score on the validation set. Hyperparameters can be found in subsection 4.6.1.

Model	MultiWoZ 2.4 JGA ↑	ACI-Bench R-L ↑	RadQA EM ↑
$T5_{base}$	71.5	47.9	–
PIE-$T5_{base}$	76.3	53.8	53.7
PID-$T5_{base}$	75.5	54.0	52.4
$T5_{large}$	72.5	52.5	–
PIE-$T5_{large}$	77.5	54.7	55.4
PID-$T5_{large}$	76.5	55.2	54.6
LLaMA2 $_{LoRA}$	66.1	53.8	54.4

Table 4.4: Task performance comparison between the baseline models and PID-T5 over three public evaluation tasks.

4.6.4 Results

Task Performance

Table 4.4 presents the task performance results for three different full finetuning scenarios (T5, PIE-T5, and PID-T5), as well as a LoRA finetuned LLaMA2 language model. The results of MultiWoZ 2.4 and ACI-Bench indicate that subtasking and multi-prompt decoding help, since both PIE-T5 and PID-T5 outperform the standard T5 and LLaMA2 models. Although larger model size often comes with better performance in moving from $T5_{base}$ to $T5_{large}$, there are mixed results for LLaMA2 on these domains. One possible reason could be that the number of trainable parameters is fewer than that of full finetuning. More importantly, similar or greater gains in performance are obtained with smaller, more efficient models that leverage task structure.

	MultiWoZ 2.4				ACI-Bench				RadQA			
Model	JGA ↑	FLOPs ↓	Sp↑	Sp^{batch} ↑	R-L ↑	FLOPs ↓	Sp↑	Sp^{batch} ↑	EM ↑	FLOPs ↓	Sp↑	Sp^{batch} ↑
PIE-$T5_{base}$	76.3	1.0x	1.0x	1.0x	53.8	1.0x	1.0x	1.0x	53.7	1.0x	1.0x	1.0x
PID-$T5_{base}$	75.5	**0.1x**	**1.9x**	**4.6x**	54.0	**0.4x**	**1.1x**	**1.1x**	52.4	**0.6x**	**1.3x**	**1.3x**
PIE-$T5_{large}$	77.5	1.0x	1.0x	1.0x	54.7	1.0x	1.0x	1.0x	55.4	1.0x	1.0x	1.0x
PID-$T5_{large}$	76.5	**0.1x**	**2.8x**	**4.2x**	55.2	**0.4x**	1.0x	**1.5x**	54.6	**0.5x**	**2.8x**	**1.3x**
LLaMA2 $_{LoRA}$	66.1	0.7x	0.2x	0.5x	53.8	4.8x	0.3x	0.3x	54.4	26.8x	0.2x	0.1x

Table 4.5: Task performance and inference efficiency comparison between PIE-T5, PID-T5 and LLaMA2 over three public evaluation tasks. Sp and Sp^{batch} represent the relative speed-up in single-instance and batching scenarios computed on NVIDIA A100. We present the relative ratios of FLOPs, Sp and Sp^{batch} compared to PIE-$T5_{base}$ or PIE-$T5_{large}$, across other models. Overall, PID-T5 achieves best computation efficiency across all tasks and achieves comparable task performance on MultiWoZ 2.4 and RadQA and better task performance on ACI-Bench.

Computation Efficiency

Since PIE-T5 and PID-T5 achieve better results than standard T5, we compare PIE-T5 and PID-T5 in Table 4.5. Regarding computational efficiency, measured in FLOPs, PID-T5 significantly outperforms PIE-T5 in reducing the number of arithmetic operations because it processes each input only once. PID-T5 achieves superior speed-up in both single-instance and batching scenarios across three datasets and two model sizes, while obtaining similar performance (98-101%) to PIE-T5. Additionally, PID-T5 offers greater reductions in computational costs (2-10x) and further accelerates efficiency when dealing with a larger number of subtasks—for example, managing 30 slots in MultiWoZ 2.4 versus 4 sections in ACI-Bench and

2-6 questions in RadQA.

	MultiWoZ 2.4			ACI-Bench			RadQA		
Learning Method	Model	Size↓	JGA ↑	Model	Size↓	R-L ↑	Model	Size↓	EM ↑
In-context learning SOTA	$\text{Codex}_{\text{davinci}}$	175B	62.4	GPT4-32k	1760B	54.3	GPT3	175B	36.2
Full finetuning SOTA	T5_{XXL}	11B	75.9	$\text{Bart}_{\text{large}}$	4×406M	48.6	T5_{large}	**770M**	**55.0**
Our PID	T5_{large}	**770M**	**76.5**	T5_{large}	**770M**	**55.2**	T5_{large}	**770M**	54.6

Table 4.6: We choose previous state-of-the-art (SOTA) generative models from the literature with the most comparable model sizes. In-context learning SOTA results for MultiWoZ 2.4, ACI-Bench, and RadQA are reported in the studies [43, 121, 55], respectively. For full finetuning SOTA, the results are reported in the studies [125, 121, 55].Yim et al. [121] use four $\text{Bart}_{\text{large}}$ models (one for each section), resulting in quadruple the size of a single $\text{Bart}_{\text{large}}$ (406M).

Comparison Between PID-T5_{large} and State-of-the-art Models

Table 4.6 illustrates the comparison between our method, PID-T5_{large}, and existing state-of-the-art approaches. Our PID-T5_{large} outperforms in-context learning methods on all three datasets with much smaller models.

Compared to full finetuning, our model outperforms on MultiWoZ 2.4 and ACI-Bench, but slightly underperforms on RadQA. This discrepancy could be attributed to the fact that the prompts in RadQA are less structured compared to the fixed types of prompts in tasks with structured outputs.

We omit the inference cost of state-of-the-art models since access to the language models for in-context learning is restricted to API calls, and some fine-tuned models' checkpoints are unavailable. Nevertheless, we can assume that the inference cost exceeds that of the proposed PID-T5 due to the larger model sizes or inference techniques employed. The T5_{XXL} inference in MultiWoZ 2.4 follows the standard T5, while the inference approaches for $\text{Bart}_{\text{large}}$ and T5_{large} in ACI-Bench and RadQA are consistent with those in PIE-T5.

Comparison Between In-context Learning and Finetuned Models in the Low-resource Setting

Figure 4.3 shows the comparison between in-context learning and full finetuned models. We use the same 1%, 5% and 10% training set provided in Hu et al. [43]. PID-T5_{base} surpasses $\text{Codex}_{\text{davinci}}$ when the 5% training ($\approx$ 374 examples) is available with 0.1% model size.[6] The result suggests that the small, finetuned model is still useful when a reasonable amount of training data is available.

[6]Some papers mention that $\text{Codex}_{\text{davinci}}$ is 175B, but OpenAI does not officially confirm that.

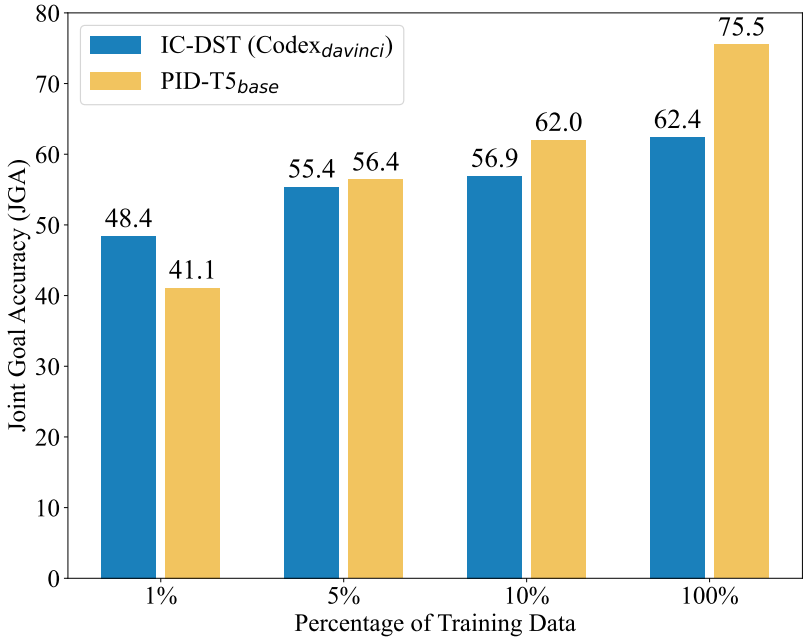


Figure 4.3: Comparison between in-context learning and full finetuning on MultiWoZ 2.4 test set. The JGA scores are reported at 1%, 5%, 10% and 100% of training data. We use the 3 random splits of the training data provided by IC-DST[43].

Efficiency under the Batching Scenario

The efficiency of batching is reflected in latency w/ batching. Figure 4.4 shows the computational efficiency and the task performance. PID-T5 outperforms the standard T5 model and achieves similar task performance to PIE-T5, while maintaining the same scale of FLOPs as the standard T5. Furthermore, the latency w/ batching of PIE-T5 is more sensitive to model size than either T5 or PID-T5.

	Data	Training FLOPs ↓	JGA ↑
T5$_{\text{base}}$	$(\mathcal{X}, \mathcal{Y})$	1.5×10^{17}	71.5
PIE-T5$_{\text{base}}$	$(\mathcal{X}, Y_u, Z_u)$	5.0×10^{18}	76.3
PID-T5$_{\text{base}}$	$(\mathcal{X}, \mathcal{Y}, \mathcal{Z})$	1.2×10^{17}	75.5

Table 4.7: Comparison of training cost across models and different training procedure on MultiWoZ 2.4. The training computational costs are reported in FLOPs, and the JGA scores are reported on the test set.

Training Efficiency

Table 4.7 shows the training costs associated with models and training strategies described in subsection 4.6.3. PIE-T5$_{\text{base}}$ incorporates prompts within its encoder, necessitating the enumeration of all prompts

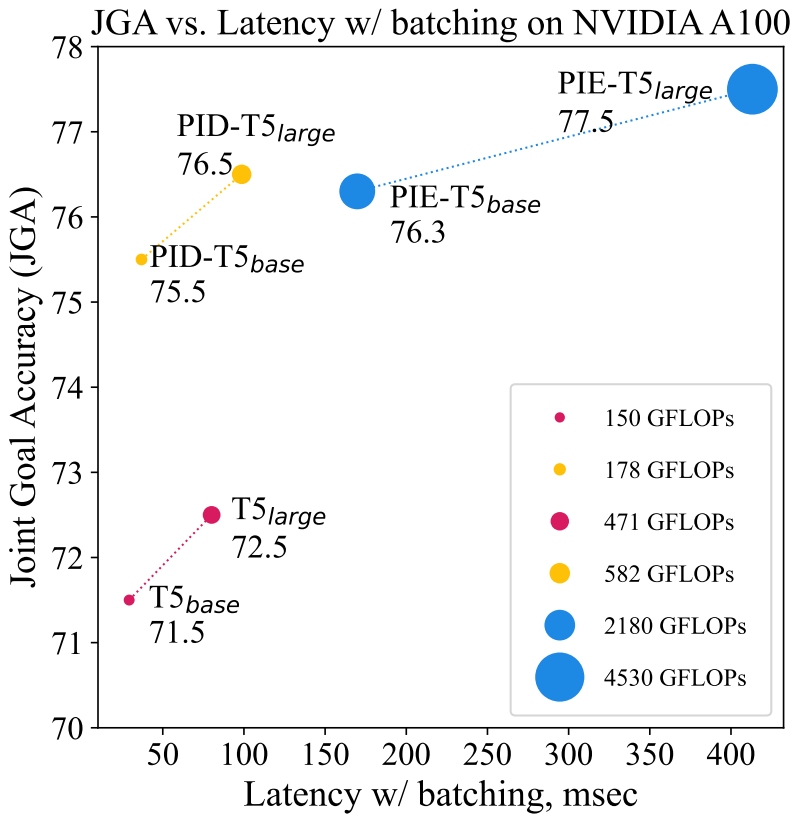


Figure 4.4: Comparison of joint goal accuracy (JGA) and latency w/ batching on the MultiWoZ 2.4 test set. Models positioned in the upper left corner indicate superior task performance coupled with faster decoding. A larger bubble indicates the model requires more FLOPs to complete a instance.

$\mathcal{Z} = (Z_1, \ldots, Z_U)$ for every input $\mathcal{X}$ during both training and testing phases. Consequently, PIE-T5$_{\text{base}}$ requires more FLOPs, i.e., longer training duration, as the total number of training samples is increased by a factor of U. Conversely, PID-T5$_{\text{base}}$ allows the reuse of the same input across all prompts, keeping the total number of training examples the same as T5. PID-T5$_{\text{base}}$ not only maintains a comparable JGA score and has efficient inference but also reduces training costs.

When creating batches for training, we pad input and output of all examples as the same length in the same batch. Since vanilla T5 predicts all output in a single sequence, it is more likely to get long output sequences in a batch. In contrast, since our prompt-in-decoder (PiD) divides the output sequence into smaller pieces (subtasks), the padding issue is mitigated. Although vanilla T5 takes slightly more FLOPs than that of PiD, the numbers are still of the same scale.[7]

[7]We follow the standard way in HuggingFace toolkit to create batches and acknowledge that there could be a more optimized way to create a batch. We will raise this issue in the discussion.

	JGA↑	L_{A100} ↓	Sp ↑	L_{2080Ti} ↓	Sp ↑
PIE-$T5_{base}$	76.3	146	1.0x	209	1.0x
PID-$T5_{base}$	75.5	78	**1.9x**	91	**2.3x**
PIE-$T5_{large}$	77.5	413	1.0x	625	1.0x
PID-$T5_{large}$	76.5	147	**2.8x**	163	**3.8x**

Table 4.8: Comparison of latency (measured in msec) between different levels of GPUs on MultiWoZ 2.4. L_{A100} and L_{2080Ti} stand for latency on NVIDIA A100 and RTX 2080Ti, respectively. Sp represent the relative speed-up relative to PIE-$T5_{base}$ or PID-$T5_{large}$.

Latency on Different Levels of GPUs

Table 4.8 illustrates that our PID-$T5_{base}$ surpasses PIE-$T5_{base}$ in efficiency under a single-instance scenario. This difference becomes more pronounced when using a consumer-grade GPU, e.g., NVIDIA RTX 2080Ti, and inferencing on the larger model.

Model	Subtask	JGA ↑	FLOPs ↓	Sp ↑
$T5_{base}$	All	71.5	1.0x	1.0x
PID-$T5_{base}$	Domain	72.5	2.3x	12.3x
PID-$T5_{base}$	Domain-Slot	75.5	2.5x	5.9x

Table 4.9: Comparison between different subtask scales, i.e., 30 domain slots or 5 domains, on MultiWoZ 2.4. Sp represents the relative speed-up in latency computed on NVIDIA A100. The model with the domain subtask predicts all active slot values in that domain.

Effect of Different Subtask Granularity

In MultiWoZ 2.4, T5's output can be broken down into subtasks according to either domains (where the output is a sequence of observed slots and their values) or domain-slot pairs (where the output is the slot value). Each domain or domain-slot pair is associated with a individual prompt. As demonstrated in Table 4.9, the system PID-$T5_{base}$ reveals that employing multi-prompt decoding can enhance both the inference speed and the joint goal accuracy (JGA) score, regardless of the granularity of the subtask units. While utilizing domains as subtask units leads to more speed-up, the use of slots as subtask units yields the best JGA.

Effect of Subtasking for Long Output

In ACI-Bench, we adopt the structure proposed by [121] for organizing the summary output into four distinct parts: subjective, objective examination, objective findings, and assessment with planning. The mean section

		Section (ROUGE-L)			
Model	All	1	2	3	4
$T5_{base}$	47.9	34.3	28.8	28.4	17.9
PIE-$T5_{base}$	53.8	**36.9**	57.2	50.9	35.4
PID-$T5_{base}$	**54.0**	36.6	**57.7**	**58.9**	**35.9**

Table 4.10: Comparison between T5, PIE-T5 and PID-T5 on ACI-Bench test 1 set.

lengths are specified as 285, 98, 35, 254 tokens, respectively. The first section details the patient's medical history, while the fourth section focuses on assessment and planning; these sections surpass the second and third sections in terms of length.

Table 4.10 reveals that the standard $T5_{base}$ model underperforms with longer outputs, especially when the generation reaches the end of the sequence, i.e., performance of later sections are much worse than for the other models. Both PIE-$T5_{base}$ and PID-$T5_{base}$ models demonstrate improved performance with subtasking, allowing the model to "focus" on one subtask at a time.

4.7 Related Work

Improving the efficiency of transformer decoding hinges on minimizing memory access and reducing redundant computations via two primary ideas, increasing operational intensity and the reuse of key-value tensors.

Attention optimizations. Increasing operational intensity can be achieved through hardware-friendly attention functions or model architectures, which increase the number of operational operations per memory access. FlashAttention [21] and FlashInfer [119] use kernel-level optimization to fuse the attention mechanism into a single kernel function. Multi-query attention [97] and grouped-query attention [2] modify transformer architectures to employ a single (or fewer) key-value attention head for multiple query heads to reduce the memory access.

Key-value tensor caching. Another method to enhance decoding efficiency is the reuse of key-value tensors of shared common prefixes. This technique reduces redundant computations for subsequent requests

with the same shared prefix. Methods like paged attention [52] and radix attention [128] mitigate the redundant storage of overlapping key-value cache. While paged attention and radix attention address the memory fragmentation issue and enable memory reuse for shared prefix, they are less than optimally efficient because their implementation lacks compute-level memory optimization. Consequently, the key-value cache of the shared prefixes still needs to be loaded multiple times during computation.

While most current research focuses on decoder-only models, our work emphasizes encoder-decoder models, as they offer superior performance in specialized domains. Our method is compatible with these kernel-level efficiency techniques, e.g., FlashAttention [8] and paged attention [9], in principle leading to further efficiency gains when used in concert.

Hydragen [48] is the most closely related concurrent research to our work. Both our method and Hydragen aim to increase operational intensity by sharing key-value tensors for shared prefixes. However, the main difference between our method and Hydragen are listed as follows. First, Hydragen mainly focuses on decoder-only models but we focus on encoder-decoder model as our experiments shows that encoder-decoder models perform better than decoder-only models when trained on specialized domains. Second, Hygragen emphasizes speed-up metrics in question answering whereas we evaluate both speedup and accuracy metrics across various applications, including dialogue state tracking, summarization, and question answering.

4.8 Limitations

The types of tasks where our method is applicable are currently limited to decomposable tasks with a shared input document. The subtasking strategies in our datasets were designed by humans, e.g., according to structured output sections. Instead of using human-designed subtasking rules, a potential avenue for exploration is to allow a model to learn how to subtask, which can additionally make more tasks possible. While our subtasking experiments use only encoder-decoder models, our strategy of sharing an embedding and decomposing a task should work with decoder-only models, but experimental analysis is left to future

[8] `FlashT5` enhances T5 by incorporating FlashAttention. `https://github.com/catie-aq/flashT5`

[9] As of the work published on ArXiv May 22th 2024, vLLM was planned to support encoder-decoder models but was still under development.

work.

4.9 Summary

In this chapter, we study settings for decomposable tasks in NLP where multiple subtask prompts are applied to the same document or dialogue. The subtasking approach allows an encoder-decoder model to individually address smaller and simpler components of the main task, leading to improved task performance. The strategy of moving the prompts from the encoder to the decoder allows our PID configuration to reduce computational costs by encoding the input just once and then sharing it to multiple subtasks to decode outputs in parallel, which further speeds up inference time while either maintaining or improving task performance. Our approach achieves higher efficiency and comparable accuracy to existing approaches, which is particularly valuable in scenarios where computational resources are scarce.

Chapter 5

Collaborative Human-LM Data Synthesis

5.1 Introduction

Rapid advances in natural language processing have driven interest in its use in a wide variety of domains. However, applications that involve human-human interaction, such as call center dialogues, have had limited success [53, 3, 80, 67]. One reason is that natural problem-solving dialogues are not typically publicly available for privacy reasons, restricting opportunities for researchers to explore methods in advancing applications for these domains. Further, annotating private datasets can be expensive because of the need for in-house expertise, so training resources are limited. In this chapter, we introduce a method to fill the data gap using synthetic data generated by a collaborative human–language model (human-LM) framework for the dialogue information extraction task. Specifically, we experiment with a task of extracting information from auto insurance call center dialogues, using public synthetic data to improve performance on a private dataset. The extracted information is utilized to create a summary of the critical details from the dialogue. This summary helps the auto insurance adjuster quickly comprehend the key points of the accident without having to read the entire conversation. By focusing on the essential information, the adjuster can efficiently assess the situation and make decisions.

Many available dialogue datasets are designed for training *virtual agents*, collected using pairs of humans to perform a task [13, 88, 15]. Designing for human-machine interaction results in dialogues that lack the complexity of human-human dialogues. Additionally, human-only data collection can have limited

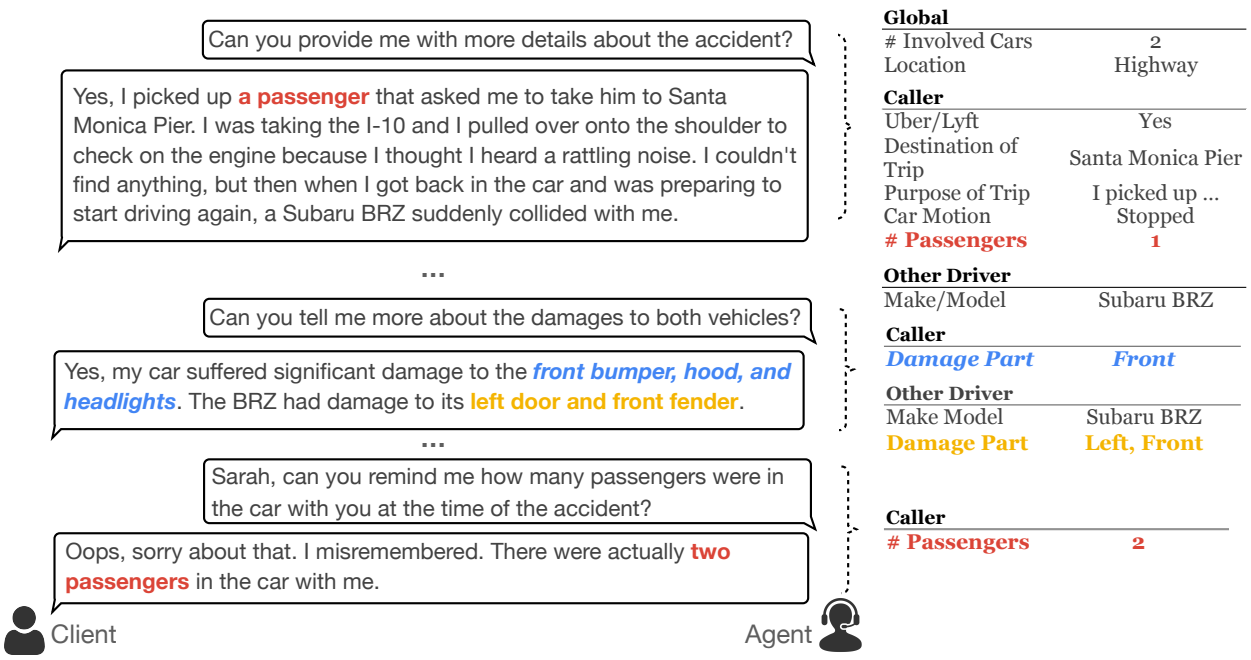


Figure 5.1: An illustrative snippet of our dialogue with entity-slot-value triples. Yellow is the slot with multiple values. *Italic blue* and yellow are the same slot (*Damage Part*) with different entities (e.g., *Caller* and *Other Driver*). Red is a slot with a value update.

content diversity, result in imbalanced training sets, and does not scale to more complex tasks, due to the high cost of employing domain experts [37, 30, 81].

To reduce data collection costs, researchers have explored the use of language models (LMs) to generate synthetic training data [63, 108, 6, 59, 4]. Synthesized data can target long-tail phenomena [18, 127, 123, 40] and allow for public release of data that closely emulates real-world privacy-constrained domains, such as the medical domain [78]. Although LMs can follow instructions to generate text that closely resembles human writing, there can be challenges to ensure that the data are diverse and not too simplistic [99, 63]. In addition, they still suffer from incoherence and consistency issues [19, 26]. To mitigate the shortcomings of LMs, human-LM collaboration can offer a robust solution, leveraging the strengths of both humans and machines [95, 103].

The idea of integrating human intelligence with artificial intelligence was initially introduced in Licklider [60]. Recent research has highlighted the proficiency of human-LM collaboration in generating a variety of data; however, most of the study focuses on short dialogues and text [63, 9]. In contrast, we investigate using a human-in-the-loop framework to create **lengthy and complex dialogues**.

Our work proposes a human-LM collaborative framework for dialogue generation (DIALGEN) that leverages the scalability and creativity of generative models, yet retains controllability through humans. Human collaborators edit the synthesized dialogues, which we use to boost information extraction performance on real-world call center data.

Many call center dialogues involve problem solving where customers provide information to an agent through question-answer pairs and clarifications that need to be interpreted in the context of the dialogue history. Our information extraction (IE) task is thus framed as an iterative information update after each agent-customer exchange, analogous to dialogue state tracking (DST) in task-oriented dialogues. However, unlike DST, the information extracted from each turn is collected to create a summary of the call rather than to generate a virtual agent's response or make an API call. In addition, the summary includes entities that are associated with attributes (slots) and values. To evaluate models on this IE task, we introduce entity-centric scoring methods that allow partial matching of multiple and descriptive values.

We demonstrate the effectiveness of DIALGEN by generating data in auto insurance calls, a domain with privacy restrictions that prevent public release of actual call recordings, and by performing information extraction. We work with a private dataset containing 34 dialogues with an average 197 utterances per dialogue and synthesize 235 dialogues with an average 46 utterances per dialogue. Experiments in our IE task show that additional synthetic data relatively improves model performance by 25% in the full F_1 score.

To summarize, our main contributions in this chapter are:

- We design DIALGEN, a collaborative human-LM framework for generating complex dialogues in domains where privacy constraints have previously prevented data sharing with the research community. Synthetic data, training documentation and prompts are released.[1]
- We present DIALGEN-AIC, a custom dataset designed to illustrate the complexity of real-world auto insurance call center data. While not intended as a benchmark, DIALGEN-AIC aims to provide a demonstration of the complex nature of real conversations and the challenges faced in this domain, including linking information with different entities and tracking multiple values in a single slot.
- We propose an entity-centric scoring methodology that considers information links to different entities, allows for multiple slot values, and provides partial match scores for descriptive values.

[1] https://boru-roylu.github.io/DialGen

- We compare our DIALGEN framework against a fully automatic LM framework and find that human collaboration adds value during both in generation and annotation.

This chapter contains material that was originally published in Lu et al. [68]. Nikita Haduong and I were the main contributors of this work. I initiated the project idea, implemented the collection framework and design experiments. Nikita conducted the data collection process with annotators and ran the experiments on the private data. We discussed the project and wrote the paper with the other collaborators throughout the project. We made equal contributions in conducting experiments and analysis.

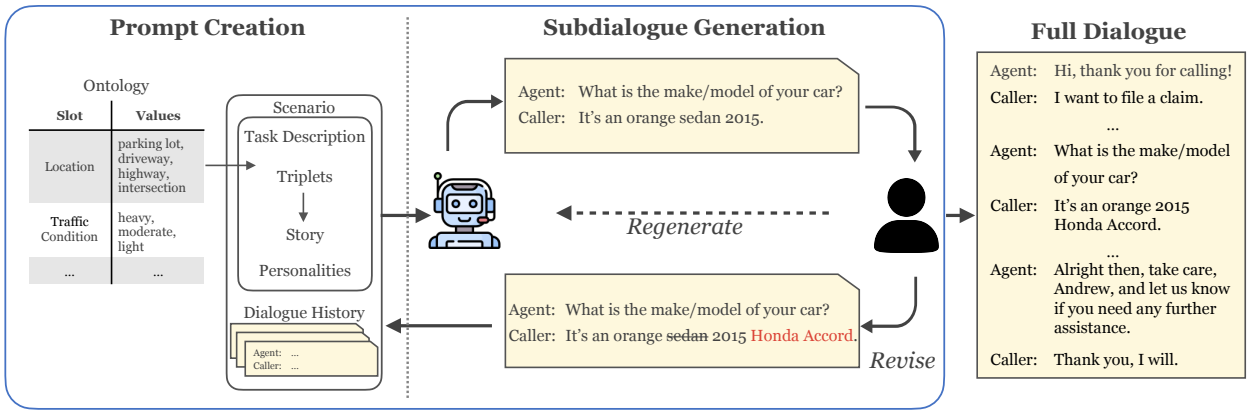


Figure 5.2: In the DIALGEN framework, a language model (LM) and a human reviewer collaborate to generate a dialogue. First, a story is created by the LM, using randomly sampled entity-slot-value triplets from the ontology. Second, the LM generates a subdialogue, using a task description, triplets, story, personalities, and dialogue history. The reviewer evaluates how the subdialogue fits with the task requirements and dialogue history. If not satisfied, the reviewer can have the LM regenerate the subdialogue before revising it. The revised subdialogue is added to the dialogue history for generating the next subdialogue. This iterative process continues until the dialogue is complete.

5.2 Dialogue Generation (DIALGEN)

As shown in Figure 5.2, our DIALGEN framework is designed to generate schema-guided dialogues through human-LM collaboration. An LM is selected as the backbone, then the data generation process begins with an initial task prompt consisting of natural language description for the desired dialogue (e.g., task description, desired slots, story, and personalities) and dialogue history. During each iteration, the LM first proposes a candidate subdialogue based on the history (the initial task prompt and the generated conversation so far). A human reviewer with sufficient domain knowledge then validates, edits and annotates the subdialogue,

before requesting a continuation via an updated prompt to the LM. The reviewer can optionally augment

Instruction	Count
Have CALLER describe more car accident details with complex reasoning that involves two cars' motion.	23
Have CALLER's response be less specific. have AGENT asks for more details.	18
Split AGENT's questions into multiple turns	18
Have CALLER's response be less specific. have AGENT asks for more details. have AGENT asks a question for car accident details.	15
Have AGENT ask for permission to record the call.	15
Ask for email address and home address	14
Have CALLER ask AGENT questions about her insurance coverages in multiple turns	13
Have AGENT ask CALLER more questions about the accident details	12
Have CALLER misremember the details. AGENT double check with CALLER.	12
Explain coverages	12
Have CALLER corrects wrong information. have AGENT asks for clarification.	12
Break this conversation down into multiple turns of dialogue	11
Have AGENT ask for contact information	10
Break these turns down into multiple turns of back and forth dialogue	10
AGENT needs to split up her questions.	10

Table 5.1: Instructions with a frequency of 10 or more times used by humans to regenerate a subdialogue.

the prompt with a specific instruction (see Table 5.1) related to the desired dialogue flow. This process repeats until the dialogue is complete. At a high level, the human-in-the-loop mechanism ensures that the resulting dialogues are coherent and consistent with the prompt, covering desired content and fulfilling style specifications from domain experts. In the following, we describe each component of DIALGEN in detail.

5.2.1 Prompt for Dialogue Generation

The prompt for generating synthetic dialogues includes: the task description, entity-slot-value triplets, story, personality and dialogue history. An example of a full prompt is given in Table 5.2.

Task Description. Similar to the task descriptions given to humans in Wizard-of-Oz setups [49], the template-based task description gives the information about the dialogue participants and the task scenario for the conversation, such as having the LM role-play as a user calling to file a claim with an agent at an insurance company, e.g., "*Role play car accident claim call. One person is Alice, an agent for a car insurance company, and the other is Bob, the caller who wants to file a claim.*"

Entity-slot-value Triplets. We randomly sample entity-slot-value triples from the expert-authored ontology to steer the LM to generate required content in the dialogue, allowing precise coverage of specific information, e.g., (*Caller, Injury, Neck*).

Story. Kim et al. [50] synthesize social dialogues from triples of common sense knowledge by first using a social narrative to set up the scenario. We similarly use the randomly sampled triplets to generate a

```
<short_summary>
story
Bob Parkhurst had a busy day at work, and all he wanted to do was to go grocery shopping. As he backed out of her parking spot in
      the Office Depot parking lot, he failed to notice the gray MAZDA B-Series Extended Cab driven by Spencer Tullar as he turned
       into the same aisle from the opposite direction.
Spencer, who was on his way to run some errands, had been driving down the parking lot in extremely slow speed when suddenly he
      saw Bob's yellow car backing out of his spot. He didn't think much of it and was about to just drive behind her when, at the
       last minute, he noticed that Bob seemed to be backing out without looking around. Spencer slammed on his brakes, but it was
       too late. The front right of his truck smashed hard into the back passenger side of Bob's car.
The impact of the collision caused Bob's car to spin around and come to a stop. He immediately felt a sharp pain in her neck and
      knew that something was wrong. As he tried to get out of the car, he realized that he couldn't move his neck without
      experiencing excruciating pain.
Spencer got out of his truck and approached Bob's car, he asked if Bob was okay. Bob told him that he was hurt and needed medical
      attention. Spencer called 911 immediately while also trying his best to comfort Bob until help arrived.
When emergency services arrived shortly after, they found Bob slumped over in her seat, clutching his neck in agony. The
      responders helped her out of the car and placed a neck brace around him so he wouldn't move his head while they examined her
       injuries. They then transported him by ambulance to the hospital for further medical attention.
Meanwhile, police were already on their way. Upon arrival at the scene, they took statements from both drivers as well as any
      witnesses who may have seen what happened. Unfortunately, no one at the time had a clear view of the incident, but both
      drivers agreed that they didn't see each other before the collision.
Since both cars were still in the parking lot when the accident happened, there was no need to redirect traffic. However, the
      officers still had to direct people away from the incident site to prevent any further accidents. They also checked Spencer'
      s license and found that it was valid.
The investigation into what caused the accident was inconclusive. Neither driver was certain about who was at fault, as they both
      believed the other driver failed to observe their movements. Since no one appeared to be at fault, no tickets or
--------
entity-slot-value triplets
Accident details: (accident location, office depot parking lot), (damage part, unsure), num of passengers, witnesses, date of
      accident, time of accident, subjective fault, airbag deployed.
Evidences of the car accident: police report, (pictures, no picture), police report number, police department name, tickets
      citations.
Traffic condition: weather visibility, (obstructions to view, no).
Caller's driver action: car motion, speed, traffic controls obeyed, turn signal, (horn, no).
Caller's car information: (make/model, dodge stratus), make year, color, car mileage.
Caller's injury details: body part injured, injury type, medical treatment.
--------
task description
Have role play car accident claim call. One person is an agent Alice from a car insurance company and the other is the caller Bob
      who wants to file a claim.
At beginning of the call, have Alice ask for Bob's permission to record the call and proceeds with the conversation.
Within some <p> </p>, have simulate poor phone connection. Have Alice and Bob can not hear each other and need to repeat what they
       said.
Have Alice verify Bob personal information to access account information at the beginning of the call.
Have Bob describe the car accident by using story and tuples above to describe the accident.
Have Alice confirm new information with Bob during the call to ensure consistency.
Have Alice and Bob engage in small talk with each other.
Have Alice explain the insurance coverages to Bob.
--------
personality
Bob is impatient, feeling frustrated with the claim process or the speed at which it is progressing, may express irritation or
      urgency in their language.
Alice is conversational, personable, patient, empathetic, sympathetic and professional.
--------
instructions
Use the story, information, and personality to create a role play script and follow the task description.
</short_summary>
<div>
<p class="Alice" title="Auto Accident"> Thank you for calling! This is Alice. How may I help you today? </p>
<p class="Bob" title="Auto Accident"> Hello. This is Alice. I am calling for a car accident. </p>
</div>
Have Alice ask a question for car accident details.
<div>
```

Table 5.2: Example prompt used to generate the first subdialogue in DIALGEN-AIC. Subsequent subdialogues are generated by appending the previously completed subdialogue to this prompt. Similar to Park et al. [77], we use HTML tags to denote different dialogue elements, i.e., <p> for turns and for the subdialogue.

story with the LM before the dialogue generation. For example, the aforementioned entity-slot-value triple will be converted into the snippet of a story: "*The impact of the collision caused Bob's car to spin around and come to a stop. He immediately felt a sharp pain in his neck and knew that something was wrong*."

Personality	Description
Aggressive	Feeling angry and confrontational about the accident, may place blame on others or use aggressive language.
Analytical	Focused on the details and logistics of the claim process, may ask for precise information and explanations.
Confused	Unsure about what happened during the accident or what to do next, may ask a lot of questions.
Cooperative	Willing to work with the insurance company and other parties involved in resolving the claim.
Defensive	Feeling the need to justify their actions or place blame on others, may be unwilling to take responsibility for the accident.
Emotional	Experiencing strong emotions related to the accident, may be crying or struggling to maintain composure during the call.
Evasive	Hesitant to provide information or answer questions about the accident, may be trying to conceal something.
Impatient	Feeling frustrated with the claim process or the speed at which it is progressing, may express irritation or urgency in their language.
Reassuring	Trying to maintain a positive and optimistic outlook during the call, may express gratitude for the assistance being provided.
Upset	Feeling distressed or frustrated due to the accident and its consequences.

Table 5.3: The list of the predefined callers' personalities.

Personality. To enrich the diversity of callers, we randomly sample a personality from the predefined list (Table 5.3) for each dialogue, e.g., "*Bob is feeling distressed or frustrated due to the accident and its consequences.*" For the agent, we use the same personality for all dialogues, e.g., "*Alice is conversational, personable, patient, empathetic, sympathetic and professional.*"

Dialogue History. The LM uses the full dialogue history to generate subdialogue turns that are consistent with the flow of the conversation. During the subdialogue generation process, we append completed subdialogues before generating the next subdialogue. The initial dialogue history is always one exchange, e.g., "*Alice: Hi, thank you for calling DialGen Insurance! This is Alice. How may I help you today?*" followed by "*Bob: I am calling regarding a car accident.*"

5.2.2 Human-in-the-loop Subdialogue Generation

The dialogue is generated iteratively where each subdialogue is revised by a human reviewer. Subdialogues are individually revised by a human trained to correct common LM errors such as those described by Dou et al. [26], verify that required information is present (the sampled triples), and edit the text to meet stylistic criteria (e.g., adjusting tone). The reviewer can either revise individual turns directly or instruct the LM to regenerate specified turns, e.g., "*Have the caller correct earlier incorrect information*" (more examples in Table 5.1). The LM may try to end the dialogue by including termination signals such as "*good bye*." If the LM ends the dialogue without covering the required triplets, the reviewer can delete and regenerate the

turns.

5.2.3 Dialogue Annotation

After a subdialogue is generated, a human annotator are asked to label spans in the dialogue that have information tuples associated with the task ontology. If a tuple in turn t has a slot with the same referent and a different value than a previous turn, the human annotators are asked to resolve the duplication by indicating whether the new value is a correction UPDATE, KEEP, or additional detail to be concatenated with the previous value CONCAT. This annotation step is optional and can be decoupled from the framework depending on the target tasks or domains.

5.3 Problem Definition and Evaluation

An auto insurance call center dialogue involves a customer working together with an agent to address an issue or submit a claim. As the conversation progresses, the extracted information must be iteratively updated. This updating process is similar to the concept of DST used in task-oriented dialogues. However, unlike in a task-oriented dialogue, the extracted information is used to summarize the call, not to make API calls or generate responses by a virtual agent.

5.3.1 Problem Definition

Extracted structured information is typically represented as a collection of tuples $\{(s, v), s \in \mathcal{S}\}$, where s is a slot label, v is the associated value, and $\mathcal{S}$ is the full set of slots in the ontology. Values can be associated with a slot-dependent restricted set $\mathcal{V}_s$ or free-form text (e.g., a home address) or null. For multi-domain systems where different domains share some but not all slots (e.g., many domains have a date slot), the domain d is separately tracked: $\{(d, s, v), d \in \mathcal{D}, s \in \mathcal{S}\}$. The full set of tuples is updated after each agent-user exchange to support construction of application calls needed to complete the task.

We formalize our information extraction task as follows. Ignoring domain for brevity, define $(A, U)_t$ as the pair of agent and user turns in exchange t. Given a sequence of exchanges between an agent and a user, $\{(A, U)_1, \ldots, (A, U)_t\}$, find the dialogue state $\{(s, v), s \in \mathcal{S}_t\}$, where $\mathcal{S}_t$ is the subset of slots active at time

t (i.e., having non-null values). The state associated with the final turn T effectively provides a summary of the information extracted from the user in the dialogue.

5.3.2 Definition of Extracted Information

To accommodate the complexities of our dialogues, we augment the DST problem in three ways. First, we introduce the notion of a "referent", either with the global context or with the entity with which the extracted information is associated. Second, we allow slots to take on multiple values. Lastly, we allow slot values to be updated in multiple ways: a value can be corrected by the user, a new value can be added to form a list, or an existing value can be augmented, e.g., with details expanding on a free-form slot. Figure 4.1 provides an example of an agent gathering information about an accident together with the extracted tuples. There are three referents (*Global* context, *Caller*, and *Other Driver*); the number of passengers in the caller's vehicle was corrected from one to two; and the other driver's car has multiple *Damage Parts* (left and front).

With these changes, we describe our notation as follows, using the arrow diacritic to indicate cumulative state elements, upper case to indicate tuples and lower case to indicate labels or values, boldface to indicate a set of tuples, and calligraphic font to indicate a set of values. The initial dialogue state $\mathbf{X}_0$ is empty. The cumulative belief (CB) state $\overleftarrow{\mathbf{X}}_t$ (for $t > 0$) could be predicted directly or via a recursive state update: $\overleftarrow{\mathbf{X}}_t = update(\overleftarrow{\mathbf{X}}_{t-1}, \mathbf{X}_t)$, where only new/updated state values are predicted in the turn-level belief (TLB) $\mathbf{X}_t$ and the update function adds new slots and replaces updated slots. In the direct approach, it is possible to correct errors made by the model in previous turns, as well as introduce errors. A potential advantage of the update approach is that TLBs are shorter and therefore easier to predict.

Formally, $\overleftarrow{\mathbf{X}}_t$ and $\mathbf{X}_t$ are defined as follows. Define $\overleftarrow{\mathcal{R}}_t$ as the set of referents mentioned in a dialogue up through turn t, and $\mathcal{R}_t \subseteq \overleftarrow{\mathcal{R}}_t$ as the subset of referents associated with information updates in turn t.[2] The dialogue state and TLB after turn t, $\overleftarrow{\mathbf{X}}_t$ and $\mathbf{X}_t$, respectively, can both be represented as a set of referent-associated sets of active slots:

$$\overleftarrow{\mathbf{X}}_t = \{(r, \overleftarrow{\mathbf{S}}_{rt}), r \in \overleftarrow{\mathcal{R}}_t\} \text{ and } \mathbf{X}_t = \{(r, \mathbf{S}_{rt}), r \in \mathcal{R}_t\}$$

where $\mathbf{S}_{rt} = \{S_{r1}, \ldots, S_{rn_{rt}}\}$, n_{rt} is the number of active slots for referent r updated at turn t, and $\overleftarrow{\mathbf{S}}_{rt}$

[2] Our application uses a finite set of types $\overleftarrow{\mathcal{R}}_t \subseteq \mathcal{R}$, but it could be an open set, e.g., based on names.

denotes the cumulative set of slots. An active slot is defined as $S_{rj} = (s_{rj}, \mathcal{V}_{rj})$, where $s_{rj} \in \mathcal{S}$ is the jth slot linked to the referent r, $\mathcal{S}$ is the set of slot (or domain-slot) types, and $\mathcal{V}_{rj}$ is a set of one or more values v (categorical or free form text) associated with that slot. For our generated data, annotators are asked to provide state updates.

5.3.3 Evaluation

In IE tasks, precision, recall, and the F-measure are commonly used, while DST is based on joint goal accuracy (JGA) and slot accuracy. Similar to DST, our IE task updates extracted information across turns. However, directly adopting DST metrics for dialogue-based IE is not ideal for two reasons. First, JGA is useful for DST because DST tasks require database queries that are built from the detected slots and values. Hence, accurate prediction is needed for all domains and slots, including null-valued instances. For a complex ontology, where many slots will be unfilled, JGA effectively emphasizes precision over recall. In contrast, for an IE task, the goal is to evaluate extraction quality, for which it is useful to look at precision/recall tradeoffs. Minor errors (e.g., an additional word in a non-categorical slot) should not significantly impact the readability of the extracted information. Second, in DST, queries are issued after most turns, so evaluating average performance at all turns makes sense. In contrast, in our IE task, information is accumulated (and corrected) for a final summary. In this case, turn averaging overemphasizes earlier parts of a conversation. For that reason, our IE metric evaluates the full state (CB) at specific dialogue points (quarter, half, three-quarters, end), and turn averaging is used for evaluating the prediction of state changes (TLB).

Our task requires the scoring to handle multi-value and extended free-form text responses. For scoring purposes, we treat multi-value slots as multiple instances of a slot. For free-form values, we adapt the multi-span setup in [58] and enumerate all possible alignments between the predicted and gold values. Each gold value is aligned to one predicted value at most, and percentage match is computed based on the longest common substring (LCS) to give a partial-credit score in $[0, 1]$ (rather than requiring exact match, i.e., $\{0, 1\}$ score) for use in measuring precision and recall.

Cumulative Belief (CB) State Scores (evaluating $\overleftarrow{\mathbf{X}}$). A cumulative belief (CB) state score m is

computed for a particular turn (specific index t or dialogue-final turn) in the nth dialogue as follows:

$$m_{\text{CB}}(n,t) = \frac{1}{|\overleftarrow{\mathcal{R}}_{nt}|} \sum_{r \in \overleftarrow{\mathcal{R}}_{nt}} m(\hat{\overleftarrow{\mathbf{S}}}_{nrt}, \overleftarrow{\mathbf{S}}^*_{nrt}).$$

where m can be precision (P) or recall (R). Overall scores are obtained by averaging over all dialogues $\mathcal{N}_t = \{n : \overleftarrow{\mathcal{R}}_{nt} \neq \emptyset\}$.[3] For example, precision is given by:

$$\text{CB-}P(t) = \frac{1}{|\mathcal{N}_t|} \sum_{n \in \mathcal{N}_t} P_{\text{CB}}(n,t).$$

We compute the F_1 score after getting the average precision and recall.

Turn Update Scores (evaluating X). Several scores are computed at the turn level, all of which are based on averaging over all N dialogues in the test set as follows:

$$\frac{1}{N} \sum_n \frac{1}{|\mathcal{T}_n|} \sum_{t \in \mathcal{T}_n} m_{\text{TYPE}}(n,t)$$

where $\mathcal{T}_n = \{t : \mathcal{R}_{nt} \neq \emptyset\}$ and $\text{TYPE} \in \{\text{TLB}, \text{R}, \text{RS}, \text{SV}\}$ denotes diagnostic score type. Specific scores (m_{TYPE}) are based on:

$$\begin{aligned}
m_{\text{TLB}}(n,t) &= \frac{1}{|\mathcal{R}_{nt}|} \sum_{r \in \mathcal{R}_{nt}} m(\hat{\mathbf{S}}_{nrt}, \mathbf{S}^*_{nrt}) \\
m_{\text{R}}(n,t) &= m(\hat{\mathcal{R}}_{nt}, \mathcal{R}^*_{nt}) \\
m_{\text{RS}}(n,t) &= \frac{1}{|\mathcal{R}_{nt}|} \sum_{r \in \mathcal{R}_{nt}} m(\hat{\mathcal{S}}_{nrt}, \mathcal{S}^*_{nrt}) \\
m_{\text{SV}}(n,t) &= m\left(\bigcup_{r \in \mathcal{R}_{nt}} \hat{\mathbf{S}}_{nrt}, \bigcup_{r \in \mathcal{R}_{nt}} \mathbf{S}^*_{nrt}\right)
\end{aligned}$$

where $\mathcal{S}_{nrt}$ is the set of slot labels associated with referent r in turn t of the n-th dialogue. For each turn, the m_{TLB} indicates performance over the TLB; m_{R} indicates how well referents are recognized; m_{RS} indicates how well referents are associated with slots ignoring values; and m_{SV} gives performance of slot-value detection ignoring referents.

[3] In the first turns, it is possible that there is nothing to extract and no false predictions, in which case $\overleftarrow{\mathcal{R}}_{nt} = \emptyset$.

5.4 Datasets

	AIC	DIALGEN-AIC	AUTOGEN-AIC
# dialogue	34	235	195
# turns / dialogue	197 ± 98	46 ± 8	31± 7
# tokens / dialogue	4195 ± 2404	1128 ± 230	947±214
# user tokens / turn	18 ± 27	22 ± 17	27±17
# agent tokens / turn	25 ± 31	27 ± 14	35±16
# referent-slot pair	1622	8844	–
# unique referent-slot	109	152	–
# referent-slot pair / dialogue	48 ± 24	38 ± 8	–
% dialogue w/ updates	50.0%	14.5%	–
% dialogue w/ multiple values	50.0%	19.1%	–

Table 5.4: Statistics are calculated on the full dataset. Tokens are calculated with Huggingface T5 tokenizer.

We were provided with a private dataset of 34 natural auto insurance claim calls (AIC). In each call, the agent's task is to gather detailed information about an auto accident. The calls were human transcribed and labeled using a schema with six referents and sixty possible slots from ten domains shown in Table 5.5. Calls had high variance in length and complexity, as shown in Table 5.4. Additionally, 50% of dialogues had multiple values for at least one active slot. We split the calls into 7/4/23 for train/val./test sets aiming for a slot count split of 20/10/70.

We show the full ontology in Table 5.5 including domains, slots, and possible values. Possible referents in the AIC ontology: *Global*, *Caller*, *Other Driver*, *Caller's Passenger*, *Other Driver's Passenger*, and *Witness*. All referents could be associated with every domain/slot, although in practice certain information is almost always associated with a particular referent, e.g., Traffic Conditions (heavy, medium, light) always have a *Global* referent.

Using AIC as a target dataset for augmentation, we apply DIALGEN with ChatGPT as the LM backbone to create DIALGEN-AIC, which contains 235 labeled dialogues (three samples are shown in Table 5.6, Table 5.7, and Table 5.8). Reviewers completed a one-hour training to become familiar with the task and practiced generating one dialogue under supervision. Full training was complete after they received feedback for their first 3–5 dialogues. They were instructed to aim to generate dialogues with ≈ 50 turns. On average, each dialogue comprises 8±4 subdialogues, with 38% of turns receiving edits and 20% of turns

Domain	Slot	Possible Values
Adjuster	Explain Coverages	[]
Adjuster	Permission to Record	[yes, no]
Adjuster	Set up Inspection	[photo claim, field assignment]
Adjuster	Set up Rental	[yes, no]
ContactInfo	First Name	[]
ContactInfo	Last Name	[]
ContactInfo	Home Address	[]
ContactInfo	Phone Number	[]
ContactInfo	Email Address	[]
ContactInfo	Policy Number	[]
ContactInfo	Date of Birth	[]
DriverActions	Car Motion	[traveling forward, backing, turning, changing lanes, stopped, other, unsure]
DriverActions	Speed	[]
DriverActions	Distractions	[cellphone, animals, smoking, passengers, traffic, eating, not paying attention, other, unsure, no distraction]
DriverActions	Brake	[yes, no, unsure]
DriverActions	Horn	[yes, no, unsure]
DriverActions	Turn Signal	[yes, no, unsure]
DriverActions	Traffic Controls Obeyed	[yes, no, unsure]
Evidences	Police Report	[yes, no, unsure]
Evidences	Police Department Name	[]
Evidences	Pictures	[at scene, after accident, no picture, unsure]
Evidences	Tickets Citations	[caller party cited, other party cited, no party cited, multiple parties cited, unsure, no ticket]
Evidences	Police Report Number	[]
Evidences	Skid Marks	[yes, no, unsure]
InjuryDetails	Ambulance	[yes, no, unsure]
InjuryDetails	Body Part Injured	[head, neck, shoulder, chest, abdomen, back, limb, other]
InjuryDetails	Injury Type	[bruise, broken fracture, cut scratch, bleeding, strain sprain, sore, other, no injury]
InjuryDetails	Medical Treatment	[MRI, surgery, CAT scan, hospitalization, ER, x-ray, other]
AccidentDetails	Damage Part	[front, right, back, left, front right, front left, back left, back right, other, unsure]
AccidentDetails	Accident Location	[parking lot, driveway, highway, roadway, intersection, other]
AccidentDetails	Num of Passengers	[0, 1, 2+, unsure]
AccidentDetails	Witnesses	[yes, no, unsure]
AccidentDetails	Num of Involved Cars	[1, 2, 3, 4+, unsure]
AccidentDetails	Children Involved	[yes, no, unsure]
AccidentDetails	Airbag Deployed	[yes, no, unsure]
AccidentDetails	Towed	[yes, no, unsure]
AccidentDetails	Pedestrians Involved	[yes, no, unsure]
AccidentDetails	Date of Accident	[]
AccidentDetails	Time of Accident	[]
AccidentDetails	Subjective Fault	[caller, other driver]
CarInfo	Make/Model	[]
CarInfo	Make Year	[]
CarInfo	Color	[]
CarInfo	Car Mileage	[]
CarInfo	Rideshare (Uber/Lyft)	[yes, no, unsure]
Trip	Destination of Trip	[]
Trip	Purpose of Trip	[]
Trip	Origin of Trip	[]
TrafficEnvironment	Weather Visibility	[clear, cloudy, rainy, snowy, foggy, windy, other, unsure]
TrafficEnvironment	Obstructions to View	[yes, no, unsure]
TrafficEnvironment	Road Condition	[dry, wet, slippery, debris, potholes, straight, curved, tunnel, steep incline, flat, other, unsure]
TrafficEnvironment	Traffic Signal	[stop sign, yield sign, green light, yellow light, red light, other, unsure, no signal or sign]
TrafficEnvironment	Description of Lanes	[normal, turn lane, shoulder, other, unsure]
TrafficEnvironment	Num of Lanes	[1, 2, 3, 4+, unsure]
TrafficEnvironment	Traffic Condition	[heavy, moderate, light, other, unsure]
TrafficEnvironment	Speed Limit	[]
TrafficEnvironment	Traffic Flow	[one-way, two-way, other, unsure]
TrafficEnvironment	Parking Lot Type	[angled, straight, other, unsure]

Table 5.5: AIC ontology. Empty lists indicate free-form extractive values.

Agent	Thank you for calling DialGen Insurance! This is Greta. How may I help you today?
User	Hello. This is Suzanne. I am calling for a car accident.
Agent	Alright, Suzanne. First, I need to verify some information. May I please have your full name and policy number?
User	Yes, my name is Suzanne Luna and my policy number is 1234567890.
Agent	Thank you, Suzanne. And can you please provide me with your phone number and address?
User	Yes, of course. My phone number is (123) 456-7890, and my address is 123 Main Street.
Agent	Thank you, Suzanne. Now, could you please describe the accident for me?
User	Certainly. I was driving alone on my way to the library when I approached an intersection with a yield sign. I slowed down and checked for oncoming traffic but didn't see any cars, so I began to turn left. Suddenly, I heard a loud crunching sound, and my car jerked to a stop. I got out of my car to assess the damage and saw that a black Nissan Maxima had hit the back of my car.
Agent	Thank you for letting me know, Suzanne. Do you know the name of the street you were on when the accident happened?
User	Sure, the accident happened at the intersection of Main Street and Broad Avenue.
Agent	And I believe you said you were driving alone, is that correct?
User	Yes, I was alone in my car.
Agent	Were there any witnesses at the scene who provided their contact information?
User	Yes, there were several witnesses at the scene who provided their contact information, but I didn't get their names.
Agent	Do you remember the date and time of the accident?
User	Yes, the accident occurred a few nights ago at around 11 pm.
Agent	Do you know the specific day?
User	Let's see, it would have been Wednesday–no, Thursday night.
Agent	Thank you, Suzanne. Can you tell me if your airbags deployed during the accident?
User	No, the airbags did not deploy.
Agent	Alright, and can you describe the traffic conditions at the time of the accident? Was there a traffic light at the intersection?
User	Yes, there was a traffic light and a yield sign.
Agent	I see. Did you notice anything else about the road or lanes that might have contributed to the accident?
User	No, nothing else that I recall. It all happened so fast.
Agent	Do you remember if you were going at a particular speed when the accident happened?
User	No, I was not speeding. I don't think the other driver was either.
Agent	Alright. Can you tell me about any damage your car sustained?
User	Yes, my car sustained significant damage on the left side.
Agent	I'm sorry to hear that. Do you know if a police report was filed?
User	Yes, the Houston Police Department took our statements, and the report number is 1234567890.
Agent	Do you know if either of you received a citation or ticket as a result of the accident?
User	Yes, the other driver, Homer Shepherd, received a citation.
Agent	Alright, thank you for letting me know. And can you provide me with the make and model of your car?
User	Yes, my car is a brown sedan, a Toyota Corolla.
Agent	Thank you, Suzanne. And can I ask if you sustained any injuries as a result of the accident?
User	Yes, unfortunately, I did sustain some injuries. I needed a CT scan at the hospital after the accident.
Agent	Can you tell me where you were injured, Suzanne?
User	My neck and back.
Agent	What was the diagnosis at the hospital?
User	The diagnosis was muscle strain and some minor bruising.
Agent	I'm sorry to hear that, Suzanne. How are you feeling now?
User	I'm doing a bit better, thank you for asking.
Agent	That's good to hear. Now, let me explain your coverage options to you.
User	Okay, thank you.
Agent	First, let me go over the details of your policy with you. You have liability insurance, which covers bodily injuries and property damage to others if you are at fault in an accident, although it looks like that's not applicable here. You also have collision coverage, which pays for damages to your vehicle in the event of an accident.
User	Okay, got it.
Agent	Additionally, you have personal injury protection, which will cover your medical expenses related to injuries sustained in the accident, regardless of who was at fault. However, please note that there may be limits on the amount of coverage provided by this policy.
User	Okay, thank you for explaining that to me.
Agent	Suzanne, can you tell me if your car was towed from the scene of the accident?
User	No, it wasn't. I was able to drive it home myself.
Agent	Okay, thank you for letting me know. I will just note that in the report. That should be everything for the preliminary details. Is there anything else you'd like to add, Suzanne?
User	No, that's all. Thank you for your help, Greta.
Agent	You're welcome. We will process your claim and be in touch with you soon about next steps.
User	Okay, thank you so much.
Agent	Have a good day, Suzanne.
User	You too, Greta. Bye.

Table 5.6: Sample DIALGEN-AIC dialogue 1.

Agent	Thank you for calling DialGen Insurance! This is Loan. How may I help you today?
User	Hello. This is Joshua. I am calling for a car accident.
Agent	I'm sorry to hear that you were in an accident. What happened?
User	It was last Monday morning in a tunnel. There were several cars involved, and my car was hit from the back by a yellow Dodge Ram. I was injured and had to be taken to the hospital by ambulance.
Agent	Oh my, I'm sorry to hear that. Did you have any passengers in your car?
User	No, I was the only one in the car.
Agent	Do you know how many cars were involved total?
User	I think there were about four cars involved.
Agent	Okay, thank you for that information, Joshua. Can you describe the damage to your car?
User	The back was heavily damaged, and my car is undrivable.
Agent	Was your car towed from the scene?
User	Yes, it was. Almost all of them had to be.
Agent	Did the police come to the scene of the accident?
User	Yes, they did. They took statements from witnesses, and they also created an accident report that documented all involved parties' details.
Agent	Great, do you happen to have the police report number and the name of the police department?
User	Yes, I have them right here. The police report number is 12345678, and it was the Philadelphia Police Department.
Agent	Thank you, Joshua. Was anyone cited or received a ticket at the scene?
User	No, the police report stated that no party was cited.
Agent	Okay, thank you for letting me know. Can you describe the traffic conditions at the time of the accident?
User	Traffic was flowing smoothly in the three-lane road. There was a car that stopped in the lane to my right, and the car behind them swerved into my lane.
Agent	Did you notice any traffic signals or signs that may have contributed to the accident?
User	No, there weren't any traffic signals or signs at all in the tunnel.
Agent	I see. Can you describe your car's make and model? What year was it made? And what color was it?
User	It's a white sedan, a 2018 Honda Accord.
Agent	Thank you for that information, Joshua. Were there any witnesses to the accident?
User	Yes, there were several people who saw the accident happen. Some good Samaritans helped me after the accident and called 911.
Agent	That's good to hear. Now, can you tell me about your injuries? What kind of medical treatment did you receive?
User	I dislocated my shoulder. They performed a CT scan at the hospital to ensure that there were no internal injuries.
Agent	One more thing, Joshua. Can you remind me of the exact date and time of the accident?
User	It was on Monday morning, around 8:30 am.
Agent	Okay, just to confirm, that would be the 22nd, correct?
User	Oh, wait. I think I may have remembered it wrong. It was actually last Tuesday.
Agent	Thank you for clarifying the date, Joshua. Can you also tell me how fast were you driving when the accident occurred?
User	I was driving around 35 miles per hour.
Agent	Thank you for that information, Joshua. Do you have the contact information for any of the other drivers?
User	Yes, I got Steve Woods' phone number. She was driving the yellow Dodge Ram that hit my car.
Agent	Joshua, can you confirm how the accident occurred from your perspective? I'm a little unclear on some of the details.
User	Sure, I was driving in the middle lane and noticed a stopped car in the other lane. The yellow Dodge Ram swerved into my lane to avoid it. I couldn't change lanes because the other lane was bumper to bumper. The Dodge Ram hit my back bumper, bounced off, and hit the car on the right side of me. After that, several cars collided with each other. My car spun around and hit the tunnel wall, damaging the back.
Agent	Thank you for explaining that, Joshua. Just to confirm, were there any traffic cameras at the scene that may have captured the accident?
User	No, I don't believe there were any traffic cameras.
Agent	Thank you for providing all of that information, Joshua. Can I ask you to confirm some information about yourself?
User	Sure, go ahead.
Agent	Can I have your full name?
User	Joshua Walters. That's spelled W-A-L-T-E-R-S.
Agent	And can I have you provide your phone number and email address please?
User	Yes, my phone number is 555-1234, and my address is 123 Main Street.
Agent	Great, thank you for providing that information. Now it looks like your insurance coverage includes liability insurance, collision coverage, rental car coverage, and gap insurance.
User	Okay.
Agent	Do you understand how each of those coverages works?
User	Not really, can you explain it to me?
Agent	Sure, liability insurance covers any damages or injuries that you may cause to another person or their property. Collision coverage pays for damages to your own vehicle if you're in an accident. Rental car coverage pays for a rental car while your vehicle is being repaired. And gap insurance covers the difference between the actual cash value of your car and the balance remaining on your car loan if your car is totaled.
User	Okay, I think I understand now. Thank you for explaining that to me.
Agent	Do you have any other questions for me, Joshua?
User	No, I don't. Thank you for your help.
Agent	Before we end this call, do you happen to have any other details about the accident that you would like to add?
User	No, I think that covers everything.
Agent	Okay, thank you for providing all the information, Joshua. We will take care of everything from here.
User	Thank you so much for your help.
Agent	Of course! And please don't hesitate to reach out if you have any questions or concerns.
User	I won't. Thanks again.

Table 5.7: Sample DIALGEN-AIC dialogue 2.

Agent	Thank you for calling DialGen Insurance! This is William. How may I help you today?
User	Hello. This is Roberta. I am calling for a car accident.
Agent	I am sorry to hear that, Roberta. Before we get started, can I get your permission to record this call for quality and training purposes?
User	Yes, you can record the call.
Agent	Thank you. May I have your full name and policy number, please?
User	My name is Roberta Oliver, and my policy number is XXXXXXXX.
Agent	Got it. Can you please describe the accident to me?
User	Sure, I was driving my gold rideshare car on the highway after having dinner at a restaurant. I was changing lanes when Shane Scott's gray Chevrolet Camaro cut me off, and I had to slam on my brakes to avoid collision. Unfortunately, it was too late, and Shane's car crashed into mine.
Agent	Okay, thank you for the detailed account. Were there any witnesses who saw this happen? Any traffic controls?
User	No, I didn't see any traffic controls around. I'm not sure about any witnesses. Oh, I guess there were the passengers in Shane's car, but they were too shaken up to give their statements to the police.
Agent	Alright. How many passengers were in each car?
User	Shane had three passengers in her car. I was alone in mine.
Agent	Thank you for that information, Roberta. Can you provide me with the location details of the accident as well as the date and time it occurred?
User	It was May 15th at around 4 in the afternoon. The accident happened on the highway near exit 45B.
Agent	Thank you for sharing that information, Roberta. I forgot to ask earlier, what year is your car?
User	My car is a 2012 model.
Agent	Great, thanks for letting me know. Can you describe the traffic conditions at the time of the accident?
User	It was a beautiful day, and the traffic on the highway was moving at a steady pace. There were four lanes, and we were both in the second lane from the left.
Agent	Alright, I see. Before we proceed further, I want to let you know that I understand how stressful this situation can be. I want you to know that I am here to guide you through the process and make everything as clear and easy as possible. How are you feeling?
User	Honestly, I'm feeling pretty overwhelmed right now. My head has been hurting since the accident, and I'm worried about how much this is all going to cost.
Agent	That's perfectly understandable, Roberta. Just take a deep breath and try to relax. It's good that you're taking steps towards resolving this by calling us today. Let's move forward together, okay?
User	Okay, thank you.
Agent	Now you mentioned your head has been hurting since the accident. Did you injure your head during the crash?
User	Yeah, I hit my head on the steering wheel. Since then, I've been having constant headaches. It's been really difficult to focus on everyday tasks.
Agent	I'm sorry to hear that. Have you seen a doctor yet?
User	Yes, I went to the hospital after the accident. They gave me a CT scan which revealed that I had a minor concussion.
Agent	I'm sorry to hear that. Did they prescribe any treatment or medication?
User	Not really, other than rest and avoiding physical activities. They okayed me to go back home immediately, but I needed to have my husband check on me every few hours to make sure everything was fine that first night.
Agent	Have you been back to the hospital since to follow up on the headaches?
User	No, but I did call my doctor to ask her about it. She said that headaches are normal for the first couple of months after a concussion, but to go back if they get worse.
Agent	I see. Thank you for telling me that, Roberta, and I hope the headaches get better soon. Just a few more questions if you'll bear with me. Can you tell me which part of your car was damaged in the accident?
User	The front left side of my car was damaged. The back right side of Shane's car as well.
Agent	Thank you for that information. Now I understand that it can be frustrating when there are no witnesses to corroborate your story. However, do you have any evidence of the accident? Perhaps photos of the damage or the police report?
User	Yes, the police came to file a report. I have a copy of it at home. I also took some photos of the damage to my car and Shane's car.
Agent	Great, that will certainly help. Can you please send those photos over to our team? I can provide you with an email address where you can send them.
User	Sure, that would be helpful. What's the email address?
Agent	The email is claims@DialGen Insurance.com. Please put your full name and policy number in the subject line and attach the photos in the email body.
User	Okay, thanks. I will send them over as soon as possible.
Agent	Perfect. Is there anything else I can assist you with today, Roberta?
User	Yes, I was wondering about the insurance claim process. How long does it usually take to get a resolution?
Agent	It depends on a few factors, such as the complexity of the case and how much evidence we have. Our team will carefully review your claim and reach out to you within a few business days with a resolution.
User	Okay, that's good to know. And what about rental cars or any other expenses related to the accident?
Agent	We can certainly help you out with that if you need it. Our team can set up rental cars if necessary, and we will do everything we can to make sure you're not paying out of pocket for any expenses related to the accident. Will you be needing a rental car?
User	No, I don't think so.
Agent	Alright, no problem. If you do end up needing a rental car, feel free to let us know. We're here to help in any way we can.
User	Thanks, I appreciate it.
Agent	Of course, Roberta. Is there anything else I can assist you with today?
User	No, that's all for now. Thanks for your help, William.
Agent	It was my pleasure, Roberta. Take care and have a great day!
User	You too.

Table 5.8: Sample DIALGEN-AIC dialogue 3.

being deleted. Each dialogue involves 9 ± 10 times of partial or full subdialogue regeneration.

Data collection occurred over 2 months with multiple iterations as documentation and task instructions evolved to become more comprehensive and consistent. The human reviewers were recruited from the university list. They were compensated at a rate of $18.69 per hour following our institution's practices. A dialogue, including reviewing synthesizing and annotation processes, required 45-60 minutes, for a final cost per dialogue of $14-19. The final version of the task instructions further encouraged workers to update slot values in multiple ways and include multiple values in a slot (as described in §2.1). We follow the methodology in SQuAD [86] for calculating IAA. We select 3 trained workers who participated in data generation as our annotators. They annotated 15% of DIALGEN-AIC (32 dialogues), with a resulting IAA of 78.5% F_1. The average time to label a dialogue was 18 minutes. For every dialogue, one annotator is randomly assigned as the reference. We calculate max-F_1 of every predicted tuple for every turn and average over all turns, then average across all dialogues.

We investigate the importance of including human reviewers in DIALGEN by comparing fully automatically created dialogues with DIALGEN-AIC. These dialogues are generated by prompting ChatGPT using the same training scenarios. We refer to this LM-only framework as AUTOGEN and use it to generate the AUTOGEN-AIC dialogues. AUTOGEN prompt details are shown in Table 5.9. In comparing AIC, DIALGEN-AIC, and AUTOGEN-AIC, we find that synthetic data has fewer turns, longer turns, and less variance in length, with fully automatic data being the most extreme. Adding a human in the loop results in much longer and more varied dialogues, but they are still far from the complexity of human-human dialogues (see Table 5.4). Compared to MultiWOZ [13], DIALGEN-AIC is more complex. MultiWOZ dialogues average 14 turns and 8 active slots per dialogue, compared to 46 turns and 38 slots on average for DIALGEN-AIC, and 198 turns and 48 slots for AIC. We split DIALGEN-AIC into train/val./test sets with a ratio of 80/10/10 dialogues, selecting val./test sets by randomly sampling from the final iteration of data collection.

```
# Scenario
story
Nina Christensen had just left the pharmacy, her car slowly traveling forward in the four-lane road, headed towards church. It was
      a Sunday morning, and she was on her way to attend the weekly service. The traffic flow was moderate, and the green light
      signaled that she could proceed.
Suddenly, out of nowhere, a yellow Maybach 57 started backing up into her lane. Nina honked her horn loudly and tried to swerve to
      avoid the collision, but it was too late. The two cars collided on Nina's left side, causing significant damage.
Nina was shaken by the sudden impact and pulled over to the shoulder to assess the damage. Her shoulder was hurting, and she
      noticed that it was bleeding. She didn't know if anyone else was hurt or if there were any passengers in the other car. It
      all happened so fast.
Dorothy King stepped out of her car, clearly agitated by what had happened. She had been backing up to turn around and head back
      home when she accidentally backed into Nina's car. There were no witnesses around, but Nina quickly took pictures of the
      accident scene as evidence.
A few minutes later, a police officer arrived on the scene after someone called 911. He took down the details of what happened
      from both drivers and filled out an accident report. With no objective evidence confirmed it from both sides, he was unable
      to declare who was at fault.
However, Dorothy received a citation for careless driving since she was the one backing up into the road. She also revealed that
      she did have insurance coverage: liability insurance, collision insurance, and uninsured/underinsured motorist coverage.
Nina's car suffered significant damage on its left side while Dorothy's Maybach 57 suffered minor damage on its right rear side.
      Though Nina had no idea how long it would take for her car to get fixed or if she would need a new one altogether, she was
      grateful that the damage on Dorothy's car was minor. At least she wouldn't have to deal with any more unnecessary expenses.
The ambulance arrived on the scene a few minutes later, and paramedics tended to the drivers. Nina was told that she needed
      surgery for her shoulder, and Dorothy was diagnosed with a bruise after an X-ray. Neither of them knew if they would be able
      to drive again anytime soon.
Nina was still in shock over what had happened. She had never been in an accident before, and the ordeal was overwhelming. She
      wasn't sure how she would afford the medical bills or how she would get
--------
information
Accident details: accident location, damage part, num of passengers, (witnesses, unsure), date of accident, time of accident,
      children involved.
Evidences of the car accident: (police report, no), (pictures, at scene).
Traffic condition: (traffic signal, green light), description of lanes, num of lanes, traffic flow.
Caller's driver action: car motion, speed.
Caller's car information: make/model, make year, color, car mileage, (rideshare (uber/lyft), unsure).
Caller's injury details: injury type, (ambulance, unsure), medical treatment, (body part injured, shoulder).
--------
steps
Have role play car accident claim call. One person is an agent Joan from a car insurance company and the other is the caller Nina
      who wants to file a claim.
At beginning of the call, have Joan ask for Nina's permission to record the call and proceeds with the conversation.
Within some <p> </p>, have simulate poor phone connection. Have Joan and Nina can not hear each other and need to repeat what they
      said.
Have Joan verify Nina personal information to access account information at the beginning of the call.
Have Nina describe the car accident by using story and tuples above to describe the accident.
Have Joan confirm new information with Nina during the call to ensure consistency.
Have Joan and Nina engage in small talk with each other.
Have Joan explain the insurance coverages to Nina.
--------
personality
Nina is confused, unsure about what happened during the accident or what to do next, may ask a lot of questions.
Joan is conversational, personable, patient, empathetic, sympathetic and professional.
--------
instructions
Use the story, information, and personality to create a role play script and follow the steps.
------------------
# Demo dialogue
Agent: Thank you for calling DialGen Insurance! This is Joan. How may I help you today?
User: Hello. This is Kristina. I am calling for a car accident.
Agent: I'm sorry to hear that. Can you please provide me with the location and date of the accident, as well as the damage on your
      vehicle?
User: Yes, the accident happened on Sunday morning on the four-lane road. The other car backed up into my lane, and we collided.
      My car got significantly damaged on the left side.
Agent: Okay, thank you. Were there any passengers involved, and were there any witnesses around or at the scene?
User: No, both of us had no passengers, and I'm not sure if there were any witnesses around. But I took pictures of the accident
      as evidence.
Agent: Good thinking, Kristina. Can you please describe the traffic conditions at the time of the accident?
User: The traffic flow was moderate, and the traffic signal was green, so I thought it was safe to proceed.
========== truncated due to the page size =============
Agent: Kristina, since you need me to set up a rental car can I assume you had your car towed away?
User: Yes, my car was towed away.
Agent: One more question, Kristina. Can you tell me if you remember the time of the accident?
User: I think it was around 9:00 am.
Agent: Kristina, one more question. Did the airbags in your car deploy during the accident?
User: No, my airbags didn't deploy.
Agent: Last question Kristina, can you describe the color of the other driver's Maybach 57?
User: Yes, it was yellow.
Agent: Thank you for all of your help, Kristina. Is there anything else you would like to add before we end this call?
User: No, that's everything. Thank you for your assistance, Joan.
[End of the conversation]
------------------
Generate a new dialogue between an agent and a user for a car accident claim based on the information mention in the scenario.
Follow the format and style of the demo dialogue.
Agent:
```

Table 5.9: Example prompt used to generate the dialogues in AUTOGEN-AIC.

5.5 Experiments

5.5.1 In-context Learning

Hu et al. [43] propose IC-DST and use schema prompts and a specialized retriever to enable few-shot in-context learning to predict state change with an LM. Given longer dialogues, a more complex ontology, and more slots to track than the datasets discussed in Hu et al. [43], the representation of dialogue history becomes a crucial concern. The SQL tables of the ontology is 1696 tokens[4], and our chosen LM, ChatGPT-0301, has a token limit of 4096 tokens. To accommodate the token constraints, we truncate the in-context examples when given a longer dialogue state. We extract the TLB at the turn t and accumulate the TLBs as CB.

Furthermore, our task requires the model to identify the corresponding entity (referent) for the predicted slot-value pair. We redesign the prompt (Table 5.10) to instruct the LM to generate the referent, slot, and value simultaneously. The retriever, SBERT [89], is finetuned on the full DIALGEN-AIC training set, which is also used as the example selection pool. Due to privacy concerns, we only evaluate IC-DST on the DIALGEN-AIC test set.

5.5.2 Finetuned Transformers

We follow idea of the previous work [54, 69] to independently extracted the information and finetune T5 [85] and Long-T5 [35] with schema information embedded in the prompt. The models predict only active slots (together with referent and value) for a domain with a separate prompt for each domain. The domain-level TLB and CB predictions are aggregated over all domains to get the TLB and CB, respectively.

In addition, we explore four different configurations of prompt and model outputs:

Long-T5†: Use $\{(A, U)_\tau\}_{\tau=1}^{t-1}$ to predict CB.

Long-T5: Use $\{(A, U)_\tau\}_{\tau=1}^{t-1}$ to predict TLB; add to CB.

T5: Use $(A, U)_{t-1}$ to predict TLB; add to CB.

[4]We use the OpenAI API to count the number of tokens `https://platform.openai.com/tokenizer`.

```
CREATE TABLE AccidentDetails(
    'Damage Part' TEXT CHECK ('Damage Part' IN 'Front', 'Right', 'Back', 'Left', 'Front Right', 'Front Left', 'Back Left', 'Back
         Right', 'Other', 'Unsure'),
    'Accident Location' TEXT CHECK ('Accident Location' IN 'Parking Lot', 'Driveway', 'Highway', 'Roadway', 'Intersection', 'Other
         '),
    'Num of Passengers' TEXT CHECK ('Num of Passengers' IN '0', '1', '2+', 'Unsure'),
    'Witnesses' TEXT CHECK ('Witnesses' IN 'Yes', 'No', 'Unsure'),
    'Num of Involved Cars' TEXT CHECK ('Num of Involved Cars' IN '1', '2', '3', '4+', 'Unsure'),
    'Children Involved' TEXT CHECK ('Children Involved' IN 'Yes', 'No', 'Unsure'),
    'Airbag Deployed' TEXT CHECK ('Airbag Deployed' IN 'Yes', 'No', 'Unsure'),
    'Towed' TEXT CHECK ('Towed' IN 'Yes', 'No', 'Unsure'),
    'Pedestrians Involved' TEXT CHECK ('Pedestrians Involved' IN 'Yes', 'No', 'Unsure'),
    'Date of Accident' TEXT,
    'Time of Accident' TEXT,
    'Subjective Fault' TEXT CHECK ('Subjective Fault' IN 'Caller', 'Other Driver'),
)

CREATE TABLE Adjuster(
    'Explain Coverages' TEXT,
    'Permission to Record' TEXT CHECK ('Permission to Record' IN 'Yes', 'No'),
    'Set up Inspection' TEXT CHECK ('Set up Inspection' IN 'Quick Photo Claim', 'Field Assignment'),
    'Set up Rental' TEXT CHECK ('Set up Rental' IN 'Yes', 'No'),
)

CREATE TABLE CarInfo(
    'Make/Model' TEXT,
    'Make Year' TEXT,
    'Color' TEXT,
    'Car Mileage' TEXT,
    'Rideshare (Uber/Lyft)' TEXT CHECK ('Rideshare (Uber/Lyft)' IN 'Yes', 'No', 'Unsure'),
)

CREATE TABLE ContactInfo(
    'First Name' TEXT,
    'Last Name' TEXT,
    'Home Address' TEXT,
    'Phone Number' TEXT,
    'Email Address' TEXT,
    'Policy Number' TEXT,
    'Date of Birth' TEXT,
)

========== truncaation due to the page size ============

CREATE TABLE Trip(
    'Destination of Trip' TEXT,
    'Purpose of Trip' TEXT,
    'Origin of Trip' TEXT,
)

-- Using valid SQLite, answer the following multi-turn conversational questions for the tables provided above.

Example #1
[context]
[system] I see. Thank you for letting me know. Can you also provide me with the make, model, and year of your car, as well as its
     color?
Q: [user] Of course. It's a white Lexus sedan, 2018 model.
SQL: SELECT * FROM CarInfo WHERE Caller-Make_Year = 2018 AND Caller-Color = white AND Caller-Make/Model = Lexus sedan,;


========== truncaation due to the page size ============


Example #6
[context]
[system] Thank you for all the details, Richard. Can you please provide me with your car's make and model?
Q: [user] Yes, it's a white sedan, a 2007 make.
SQL: SELECT * FROM
CarInfo WHERE Caller-Color = white sedan AND Caller-Make_Year = 2007
 * FROM CarInfo WHERE Caller-Color = white sedan AND Caller-Make_Year = 2007
```

Table 5.10: We follow previous work [43] to represent the ontology via a SQL table format. In Appendix D3, we show an example of the prompt. For example, the domain "Adjuster" (as shown in the following text block) contains the domain slots Explain Coverages, Permission to Record, Set up Inspection, Set up Rental. The domain slots with CHECK values indicate they are categorical slots. Otherwise, the slots are non-categorical. We truncate part of the content due to the page size.

T5-SC: Use $(A, U)_{t-1}$ and previous domain-level CB to predict the domain-level state change ΔCB; update CB.

Because the input length can be longer than 1k tokens, we choose Long-T5 to cover all turns with the prompt, while the T5-based models make predictions based on the current turn only. T5-SC further considers the state change ΔCB, which is similar to the TLB but augmented with the four state-change commands.

5.5.3 Details of the Prompts for Each Configuration

```
Input:
[USER] My name is Bob Lee, and my policy number is 123456789. [SYSTEM] Thank you. Could you please provide me with your
      name and policy number so I can access your account information? [USER] Yes, that's fine. [SYSTEM] I am so sorry that
      happened. Before we begin, may I please have your permission to record this call for quality and training purposes? [
      USER] Hello. This is Bob. I am calling for a car accident. [SYSTEM] Thank you for calling AllState! This is Alice. How
       may I help you today? [domain] ContactInfo [possible slots] First Name (the First Name of the ContactInfo) [s] Last
      Name (the Last Name of the ContactInfo) [s] Home Address (the Home Address of the ContactInfo) [s] Phone Number (the
      Phone Number of the ContactInfo) [s] Email Address (the Email Address of the ContactInfo) [s] Policy Number (the
      Policy Number of the ContactInfo) [s] Date of Birth (the Date of Birth of the ContactInfo)

Output:
First Name [srv] Bob [rv] Caller [s] Last Name [srv] Lee [rv] Caller [s] Policy Number [srv] 123456789. [rv] Caller
```

Table 5.11: The prompt of Long-T5 for CB prediction.

Long-T5 for CB prediction

We present a training example for the "ContactInfo" domain with a complete dialogue history at time t. The example contains separators `[s]`, `[rv]`, and `[srv]` that label prior information as a slot, referent-value pair, or slot-referent-value triplet, respectively.

```
Input:
[USER] Hi, my name is Bob Lee. I was recently in a car accident and wanted to file a claim. [SYSTEM] Thank you for calling!
       This is Alice. How may I help you today? [domain] ContactInfo [possible slots] First Name (the First Name of the
      ContactInfo) [s] Last Name (the Last Name of the ContactInfo) [s] Home Address (the Home Address of the ContactInfo) [
      s] Phone Number (the Phone Number of the ContactInfo) [s] Email Address (the Email Address of the ContactInfo) [s]
      Policy Number (the Policy Number of the ContactInfo) [s] Date of Birth (the Date of Birth of the ContactInfo)

Output:
First Name [srv] Bob [rv] Caller [s] Last Name [srv] Lee [rv] Caller
```

Table 5.12: The prompt of Long-T5 and T5 models for TLB prediction.

Long-T5 and T5 models for TLB prediction

We present an example for the "ContactInfo" domain with the most recent two turns $(A, U)_t$ at time t. In the example, the caller (USER) mentions the first and the last name that are under the domain ContactInfo. The model is require to generate the active slots "First Name" and "Last Name" with the corresponding values "Bob" and "Lee", and referent "Caller."

```
Input:
[USER] Oh, sorry about that. You're right, it actually occurred on a Wednesday at 11 am. [SYSTEM] Also, I just wanted to
      clarify some information. In our previous conversation, you stated that the accident occurred on a Monday at 9 am.
      However, our records show that it actually occurred on a Wednesday at 11 am. Can you confirm which day and time the
      accident actually occurred? [state] Damage Part [srv] Front Left [rv] Caller [cv] Right [rv] Global [s] Accident
      Location [srv] Highway [rv] Global [s] Num of Passengers [srv] 0 [rv] Global [s] Witnesses [srv] Yes [rv] Global [s]
      Date of Accident [srv] this Monday [rv] Global [s] Time of Accident [srv] 9:00 am. [rv] Global [s] Subjective Fault [
      srv] Caller [rv] Caller [domain] AccidentDetails [possible slots] Damage Part [s] Accident Location [s] Num of
      Passengers [s] Witnesses [s] Num of Involved Cars [s] Children Involved [s] Airbag Deployed [s] Towed [s] Pedestrians
      Involved [s] Date of Accident [s] Time of Accident [s] Subjective Fault

Output:
Date of Accident [srv] Wednesday [v] this Monday [vo] [delete] [rv] Global [s] Time of Accident [srv] 11 am. [v] 9:00 am. [
      vo] [delete] [rv] Global
```

Table 5.13: The prompt of T5 with State Change (T5-SC).

T5 with State Change (T5-SC)

For T5-SC, the models need to predict entity-slot-value triplets and edit operations associated with the triplets. The final output of a state at time t will be calculated by applying the edit operations on the associated triplets given the previous state at time $t-1$. We consider four edit operations: `[new]`, `[keep]`, `[delete]`, and `[concat]`. We describe the four edit operations in the following paragraph.

If a triplet has not been observed in the previous state, the model is expected to predict `[new]`. Conversely, if the triplet has already been mentioned in the previous state, the model must predict `[keep]`. The `[delete]` operation is employed when a triplet mentioned in the previous state should be removed. If the value of a referent-slot is updated, then the model predicts both `[delete]` for the previous value and `[new]` for the updated value. On the other hand, the `[concat]` operation is used when the value of a triplet needs refinement, such as combining two values, 7 and AM, into a single value of 7 AM.

Due to the input length limit of the T5 model, we use the most recent k turns to create the previous state and omit the slot descriptions in order to cover more entity-slot-value triplets in the previous state. We get the best results when $k = 18$ for DIALGEN-AIC and $k = 20$ for AIC. We present a training example for

the "AccidentDetails" domain as follows.

In the example, the agent (SYSTEM) clarifies the date and time with the caller (USER) because the date and time the caller provides are different from the record in the agent's system. The caller admits the provided time and date are wrong. Therefore, the time and date must be updated. The date previously provided "this Monday" needs to be deleted, so we append an operation `[delete]` after the value. Similarly, we append the operation after the time "9:00 am."

5.5.4 Finetuning Details

When conducting experiments involving AIC, the model selection criterion is the highest TLB F_1 score on the AIC validation set. For experiments solely on DIALGEN-AIC or AUTOGEN-AIC, models were chosen based on the TLB F_1 score on the DIALGEN-AIC validation set. The additional hyperparameters can be found in Table 5.14.

Hyperparameter	T5 or T5-SC	Long-T5
Training batch size	16	16
Learning rate	5×10^{-4}	5×10^{-4}
Max generation length	256	256
Max input length	512	2592

Table 5.14: Hyperparameters for training T5, T5-SC and Long-T5. The other parameters are default values in Huggingface trainer.

All reported values represent the medians of five different random seeds. All experiments are conducted using either T5-base or Long-T5-base with the Huggingface implementation [113]. The training time for the full DIALGEN-AIC and AIC settings averages 3 hours on 2 NVIDIA V100 GPUs. For the experiments conducted solely on DIALGEN-AIC, 2 NVIDIA A40 GPUs are used. The total number of GPU training hours amounts to 110 hours.

5.5.5 Hyperparameters of OpenAI API Calls

Hyperparameter	DIALGEN	AUTOGEN	LM annotators	IC-DST
Language Model	ChatGPT	ChatGPT	ChatGPT or GPT-4	ChatGPT
Temperature	0.85 - 0.9	0.0	0.9	0.0
Max tokens	512	256	2048	512
Stop strings	["<\div>"]	–	–	["–", "\n", ";", "#"]
Presence penalty	0.2	0.0	0.2	0
Frequency penalty	0.2	0.0	0.2	0

Table 5.15: Hyperparameters for API calls.

We use `gpt-3.5-turbo-0301` and `gpt-4-0613` for ChatGPT and GPT-4, respectively. For AUTOGEN, the stop token is the default value of the API call. We parse the output of the response from ChatGPT and truncate tokens after the string `[End of conversation]`.

5.6 Results

We report the main results on both cumulative and turn update scores. The cumulative scores are presented in two ways: CB_{avg} as an average of CB across every user turn, and CB_Q as the CB at user turn t, where $t = \lceil QT/4 \rceil, Q \in \{1, 2, 3, 4\}$ and T is the total length of a dialogue. Thus, t will be a specific turn, at either a quarter, a half, three-quarters, or the end of the dialogue. The score of the last cumulative belief state CB_4 is the full F_1 score and can be regarded as evaluating a conversation summary. Model development was done only on the synthetic data to minimize use of real data.

Method	CB_{avg}	CB_1	CB_2	CB_3	CB_4	TLB
IC-DST	71.3	71.9	68.5	68.4	68.2	68.1
Long-T5†	71.8	72.5	71.7	71.0	70.4	–
Long-T5	66.3	64.3	64.8	64.3	63.9	68.5
T5	76.8	78.4	74.9	73.7	74.1	73.9
T5-SC	**78.2**	**79.3**	**76.4**	**76.6**	**76.9**	**74.2**
T5-SC§	78.5	78.7	76.2	76.0	76.2	75.0

Table 5.16: F_1 scores on the DIALGEN-AIC test set. § denotes test set results with name substitution.

Results on DIALGEN-AIC Test Set. The results on DIALGEN-AIC with different learning strategies and T5 configurations are presented in Table 5.16. The performance of IC-DST is lower than all T5 variants,

Method	Data	CB_{avg}	CB_1	CB_2	CB_3	CB_4	TLB
T5	AIC	38.3	39.6	37.1	36.2	35.1	34.8
T5	DIALGEN-AIC	40.4	41.7	42.6	39.9	37.7	40.9
T5	AIC + DIALGEN-AIC	43.7	42.9	42.2	43.0	41.9	43.7
T5-SC	AIC	39.2	40.0	38.1	37.1	36.1	33.9
T5-SC	DIALGEN-AIC	41.0	43.6	42.1	41.3	40.5	38.9
T5-SC	AIC + DIALGEN-AIC	**46.2**	**47.8**	**47.2**	**45.9**	**45.3**	**44.6**

Table 5.17: F_1 scores on the AIC test set for different training data.

although this may be due to the difference in use of domain-specific prompts. Note that our IC-DST implementation is based on the same ChatGPT model used for generating the DIALGEN-AIC, so the low results suggest that human collaboration creates data sufficiently different from ChatGPT text such that ChatGPT cannot easily address this task. Predicting CB directly requires the full history, which is only possible with Long-T5. With Long-T5, there is a benefit to predicting CB directly over TLB. However, optimizations needed to handle a longer history have tradeoffs that result in performance that is worse than the standard T5 model with TLB prediction for this task. T5-SC achieves the best result, which updates values rather than adding them as new values in a list.

To mitigate the potential risk of LMs generating personal information linked to randomly generated names in shared data, we replace them with other randomly generated names. As shown in Table 5.16, T5-SC exhibits comparable performance on both the original and renamed dialogues, indicating that the renaming process does not impact the model's effectiveness.

Results on AIC Test Set. The two best models (T5 and T5-SC) are used in experiments on the real data (AIC). The F_1 results for different training sources are given in Table 5.17. We measure the utility of synthetic data on model performance by varying amounts of DIALGEN-AIC. The performance for the model trained on the synthetic data alone is better than with the small amount of the real data, but the best results are obtained by model trained on the combined data. Because of the higher frequency of state changes in the human-human dialogues, there is a greater benefit from the T5-SC model for the real data, with an 8% improvement in the CB_4 score compared to 4% for the synthetic data when using all training data.

To provide more insight into performance, we present the precision/recall results for CB in Figure 5.3.

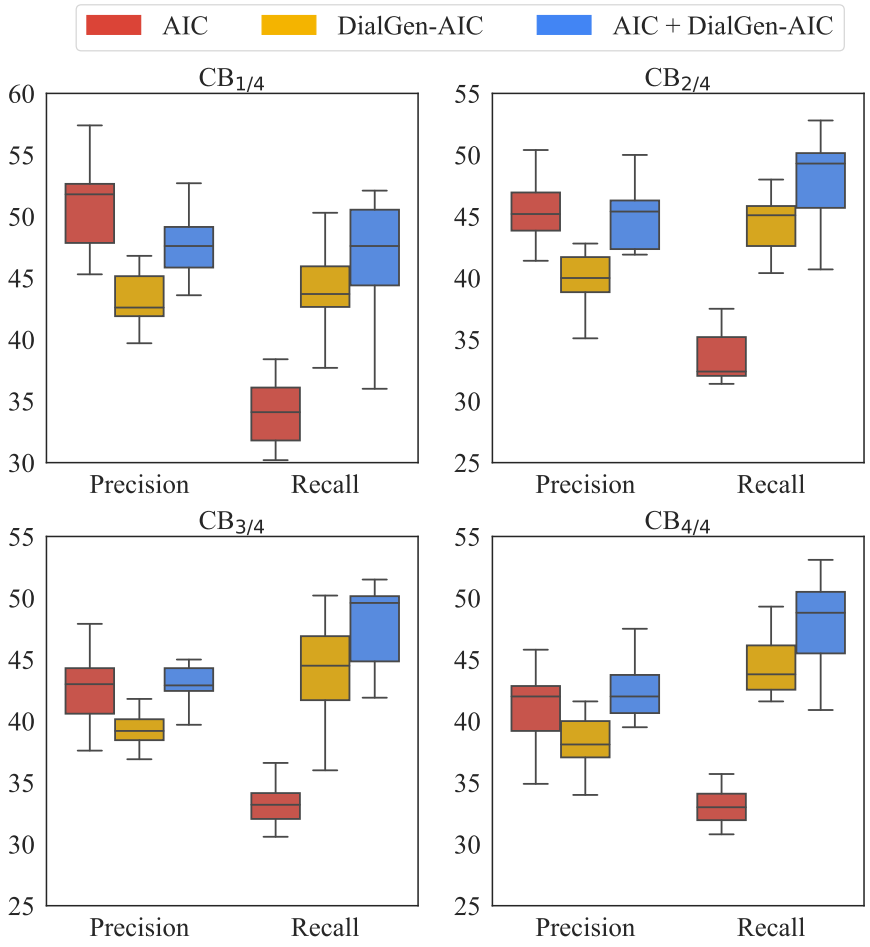


Figure 5.3: CB precision and recall scores on the AIC test set. All scores are based on T5-SC models.

Incorporating synthetic data yields higher recall and outperforms using real data alone in terms of F_1. The increased recall can be attributed to the inclusion of a wider range of values in the synthetic data, which are not covered by the AIC training set. However, this improvement comes at the expense of lower precision. By combining both data sets, the model achieves better alignment with real-world data while retaining the advantage of high recall scores from the synthetic data.

Figure 5.4 provides the TLB precision and recall results for the full state updates and different diagnostic scores (referent only, referent-slot, and slot-value). Consistent with the CB results, the biggest benefit of incorporating DIALGEN-AIC is improved recall. While referent, slot, and value all improve, the greatest improvement is in slot values.

We also experimented with varying the amount of synthetic data used in training the model in order to

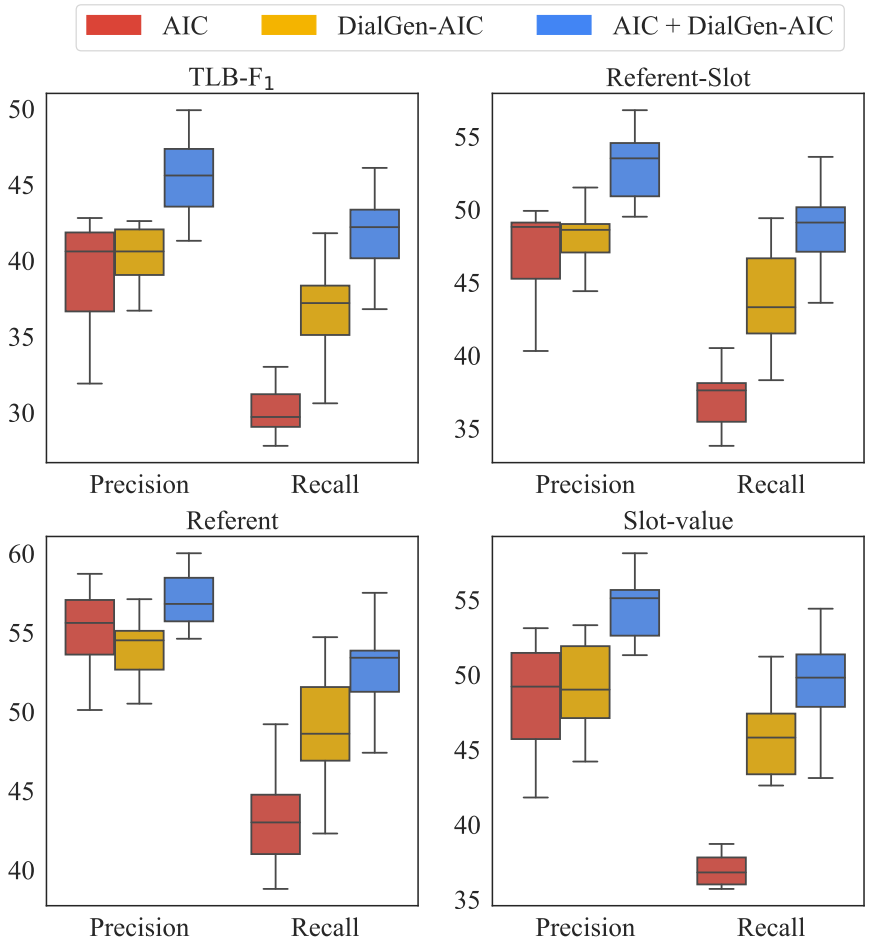


Figure 5.4: TLB and three diagnostic scores for precision and recall (m_{R}, m_{RS}, and m_{SV}) for the T5-SC model on AIC test set.

determine the relative value of synthetic versus real data. Figure 5.5 shows that using 59 synthetic dialogues (approximately 2.7K turns) yields results similar to those obtained from the AIC training set, which consists of 1.3K turns in 7 dialogues. These results suggest that roughly 2.1 times as many turns of synthetic data is needed to match the performance of the real data, or 8.4 times as many synthetic dialogues since the synthetic dialogues are shorter. However, the synthetic data is more valuable in combination with real data, for which the benefit beyond 97 dialogues (50%) is minimal. This suggests an opportunity for further improvement through strategic scenario sampling.

Comparison of DIALGEN with AUTOGEN. To understand the importance of the role of humans in data synthesis, we investigate whether an LM annotator is sufficient and compare the value of dialogues

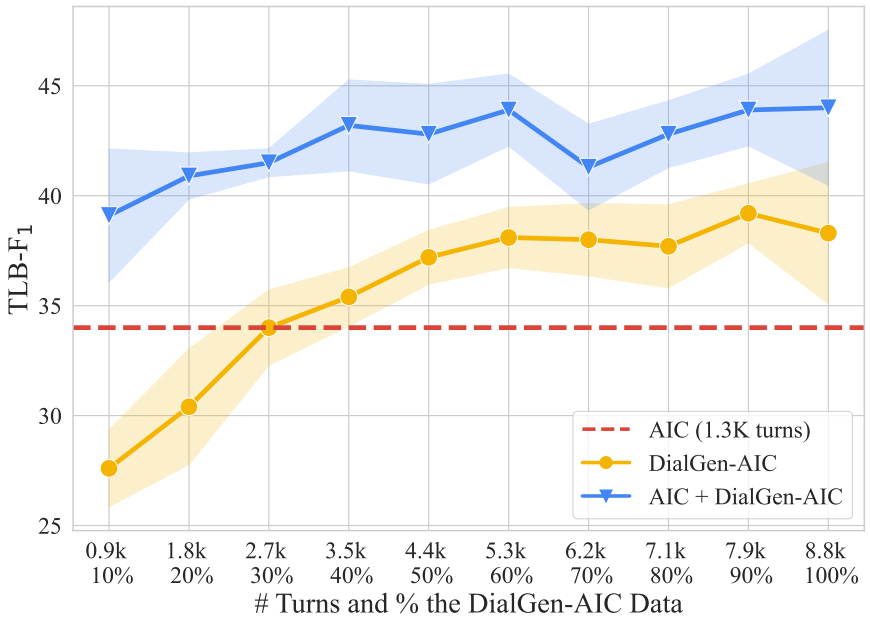


Figure 5.5: TLB-F_1 scores for T5-SC on AIC test set by varying the amount of DIALGEN-AIC training data.

Framework	Dialogue Generator	Annotator	TLB
AUTOGEN	ChatGPT	ChatGPT	49.5
	ChatGPT	GPT-4	60.7
DIALGEN	ChatGPT w/ Humans	GPT-4	63.2
	ChatGPT w/ Humans	Humans	73.9

Table 5.18: Comparison between different data synthesis frameworks. The TLB F_1 scores are obtained by finetuning a T5 models on generated data and testing on the DIALGEN-AIC test set.

generated by AUTOGEN and DIALGEN. First, we use ChatGPT and GPT-4 [74] to annotate DIALGEN-AIC training dialogues and compare the annotations with labels, obtaining 52.3 and 65.5 TLB accuracy respectively. This suggests GPT-4 is a better annotator than ChatGPT, which aligns with the findings of Gray et al. [33] and Tekumalla and Banda [100]. Then, we finetune T5 models on each setting and report TLB score to evaluate the quality of synthesized data (Table 5.18). T5 trained on dialogues generated by the DIALGEN framework and annotated by humans yielded the best result, demonstrating how a human-in-the-loop data synthesis framework can provide higher quality training data than a fully automated system.

5.7 Error Analysis

Out of the 56 slots in the AIC test set, we noticed an improvement in 45 slots, while 4 slots were tied, and the remaining 7 slots have slightly worse performance. Our error analysis reveals two main categories for the performance loss: data mismatch between AIC and DIALGEN-AIC and over-reliance on surface-level features.

5.7.1 Data Mismatch

We lose performance for the slot *Car Mileage* because of a difference in language used when describing the mileage of a car. In AIC, agents ask a binary confirmation for whether the mileage on the vehicle is above a certain threshold, whereas callers in DIALGEN-AIC describe car mileage with an exact number. For the slot *Traffic Controls Obeyed*, AIC callers indirectly indicate that traffic controls are not obeyed, e.g. stating that the other driver ran a red light. In DIALGEN-AIC, the agent asks the caller to confirm directly whether traffic controls were obeyed.

5.7.2 Surface Level Text

The model both over- and under-predicts slots due to surface-level features such as predicting *Number of Involved Cars* when the text discusses counting vehicles, despite many such instances in AIC simply describing the traffic environment to contextualize the accident, e.g., there was a vehicle in front of the caller, but it was not involved in the accident. The model also predicted this slot when there was language about the number of passengers with a driver. Similarly, *Color* would be predicted whenever colors were mentioned, e.g., a purple bruise. *Traffic Flow* was severely under-predicted when it would have been beneficial for the model to predict the slot whenever it saw information describing lane direction.

5.8 Summary

In this chapter, we propose DIALGEN, in which humans and LMs collaborate to generate long, complex dialogues. We demonstrate its effectiveness by synthesizing auto insurance calls and conducting information extraction (IE) experiments. While we build on the DST framework, our IE experiments target an ontology

and data that are more complex than the DST task was originally designed for. To serve the IE task, we introduce an entity-centric scoring methodology more suitable for our IE task than the conventional joint goal accuracy metrics used in DST. The extracted information can be considered as a conversation-level, tabular-based summary, providing a concise and informative overview for auto insurance adjusters to read.

In our controlled experiments, we contrast the outcomes of training with data synthesized through a human-in-the-loop method against data generated and annotated entirely by ChatGPT or GPT-4. Our findings indicate that human-LM collaboration enhances the process at both the data generation and annotation phases, validating the importance of including humans in the data synthesis process to generate more realistic dialogues. Our experiments demonstrate that the data generated by DIALGEN, despite dissimilarities with the data it is designed to emulate, can significantly improve model performance for information extraction on real-world human dialogues.

Chapter 6

Conclusion

To conclude, we first provide a summary of the work carried out in this thesis in section 6.1. After that, we discuss future directions for work with dialogue summarization in section 6.2

6.1 Summary and Contributions

6.1.1 Dialogue Summarization for Call Center Dialogues

In this thesis, we address the challenges in underexplored dialogue domains, specifically focusing on telecommunications service call center dialogues and auto insurance claim call center dialogues. Our goal is to enhance the efficiency of summarizing these conversations for human use through both corpus-level and conversation-level summarization techniques. We introduced the novel task of corpus-level dialogue summarization, which uses a graph to create an interpretable summary of the entire corpus. This approach significantly reduces the burden on humans by allowing them to analyze large volumes of conversations more efficiently. The graph-based summary facilitates a quicker understanding of the corpus and aids in extracting valuable insights, such as effective negotiation strategies employed by successful salespeople or efficient techniques used by experienced agents. At the conversation level, inspired by dialogue state tracking, we proposed using the dialogue state at the final turn as a conversation-level summary for auto insurance claim calls. This structured summary encapsulates the essential details required to determine fault and compensation in car accidents, eliminating the need to read through entire dialogues. Automatic generation of

structured conversation summaries has the potential to greatly increase the efficiency of claim processing, providing comprehensive and concise information for decision-making.

6.1.2 Enhancing Transformer Models with Structured Information

To improve task performance and efficiency, we integrate additional structured information into transformer models, rather than relying solely on raw text input. Our approach involves two types of structures: the latent structure of input dialogues and the expert-designed structure of output summaries.

Latent Structure of Input Dialogues

This structure is derived from a corpus-level summary graph, where each dialogue is represented as a path on this graph, creating a sequence of subdialogue states. By incorporating this latent structure as an additional feature, we enhance the capabilities of encoder-only hierarchical transformers.

Expert-Designed Output Structure

This structure includes elements such as section titles in medical notes or domain-slot names in task-oriented dialogues, which divide the main task into smaller, simpler subtasks. This subtasking approach allows encoder-decoder transformers to share encoded input and process each subtask output in parallel, resulting in more efficient transformer decoding.

6.1.3 Small Transformer Models for Specialized NLP Tasks

In this thesis, we demonstrate that larger decoder-only transformer models (e.g., LLaMA, LLaMA2, ChatGPT, GPT-4, etc.) are not always the optimal solution for all NLP tasks, particularly when some annotated data is available for fine-tuning a transformer model for complex and specialized domains. Our research reveals that smaller encoder-decoder models can often outperform these large decoder-only models, offering better or comparable performance and significantly faster inference speeds in specific domains such as dialogue state tracking and summarization.

6.1.4 Human-LM Data Synthesis Framework

Our observations indicate that solely relying on decoder-only transformer models for data synthesis fails to achieve human-like data quality, both in text generation and annotation. The highest performance is attained when human correction is integrated into the data synthesis process to refine the models' annotations.

6.2 Future Directions

We discuss a few important future directions to further improve models for dialogue summarization.

6.2.1 Direct Evaluation on Graph Quality

In chapter 3, we indirectly evaluate the quality of corpus-level summarization by using cluster and state sequences as additional features to enhance model performance on predictive tasks. Although the downstream performance is enhanced by the additional features that does not imply that the resulting graphs are the best representation of the corpus or the best representation for human to analyze. A more direct evaluation of the graph might provide a better reflection of the graph summary quality. Future work could involve developing metrics that directly assess the informativeness of the graph and its summary. Additionally, research is needed to develop a framework for user studies that could be conducted to gather feedback from domain experts on interpretability of graphs and utility for human analysts. Exploring advanced summarization techniques could further enhance the usability and interpretability of these graph representations.

6.2.2 Enhancement of Structure-aware Decoding

In chapter 4, our prompt-in-decoder (PID) method is currently limited to decomposable tasks that use a shared input document. The subtasking strategies in our datasets were crafted by humans, typically based on structured output sections. A promising direction for future research involves developing a model that can autonomously learn how to subtask. This advancement could significantly expand the range of tasks our method can handle. Second, although our current experiments on subtasking utilize only encoder-decoder models, we propose that our strategy of shared embeddings and task decomposition should be equally effective with decoder-only models. Experimental validation of this hypothesis is needed.

6.2.3 Advanced Synthetic Dialogue Generation

While DIALGEN results in dialogues that are more similar to real human-human interactions than AUTOGEN, our analysis (section 5.7) shows that there are still substantial differences. Some of these gaps will be filled as LMs improve. However, it is likely that additional work will still be needed to further minimize the differences between synthetic and natural data, enhancing the realism of generated dialogues.

Finally, our controlled experiments comparing AUTOGEN and DIALGEN reveal that the dialogue generation and annotation process in AUTOGEN is managed by a single LM agent using various associated prompts. There is substantial potential for enhancing AUTOGEN, including exploring different prompt designs and incorporating multiple language model agents (e.g., integrating quality control agents) to improve the overall quality and reliability of the generated dialogues.

Bibliography

[1] Joshua Ainslie, Santiago Ontanon, Chris Alberti, Vaclav Cvicek, Zachary Fisher, Philip Pham, Anirudh Ravula, Sumit Sanghai, Qifan Wang, and Li Yang. ETC: Encoding long and structured inputs in transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 268–284, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.19. URL https://aclanthology.org/2020.emnlp-main.19.

[2] Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebron, and Sumit Sanghai. GQA: Training generalized multi-query transformer models from multi-head checkpoints. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 4895–4901, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.298. URL https://aclanthology.org/2023.emnlp-main.298.

[3] Tobias Albrecht, Theresa Maria Rausch, and Nicholas Daniel Derra. Call me maybe: Methods and practical implementation of artificial intelligence in call center arrivals' forecasting. *Journal of Business Research*, 123:267–278, 2021. ISSN 0148-2963. doi: https://doi.org/10.1016/j.jbusres.2020.09.033. URL https://www.sciencedirect.com/science/article/pii/S0148296320306160.

[4] Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. Self-RAG: Learning to retrieve, generate, and critique through self-reflection. In *Proceedings of International Conference*

on Learning Representations (ICLR), 2024. URL https://openreview.net/forum?id=hSyW5go0v8.

[5] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *Proceedings of International Conference on Learning Representations*, 2015.

[6] Jianzhu Bao, Rui Wang, Yasheng Wang, Aixin Sun, Yitong Li, Fei Mi, and Ruifeng Xu. A synthetic data generation framework for grounded dialogues. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10866–10882, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.608. URL https://aclanthology.org/2023.acl-long.608.

[7] Iz Beltagy, Matthew E Peters, and Arman Cohan. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020.

[8] Taylor Berg-Kirkpatrick, David Burkett, and Dan Klein. An empirical investigation of statistical significance in NLP. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 995–1005, Jeju Island, Korea, July 2012. Association for Computational Linguistics. URL https://aclanthology.org/D12-1091.

[9] Helena Bonaldi, Sara Dellantonio, Serra Sinem Tekiroğlu, and Marco Guerini. Human-machine collaboration approaches to build a dialogue dataset for hate speech countering. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 8031–8049, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. URL https://aclanthology.org/2022.emnlp-main.549.

[10] Florian Boudin and Emmanuel Morin. Keyphrase extraction for N-best reranking in multi-sentence compression. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 298–305, Atlanta, Geor-

gia, June 2013. Association for Computational Linguistics. URL https://aclanthology.org/N13-1030.

[11] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. In *Proceedings of Advances in neural information processing systems*, volume 33, pages 1877–1901, 2020.

[12] Tomáš Brychcín and Pavel Král. Unsupervised dialogue act induction using Gaussian mixtures. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 485–490, Valencia, Spain, April 2017. Association for Computational Linguistics. URL https://aclanthology.org/E17-2078.

[13] Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gašić. MultiWOZ - a large-scale multi-domain Wizard-of-Oz dataset for task-oriented dialogue modelling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5016–5026, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1547. URL https://aclanthology.org/D18-1547.

[14] Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. Accelerating large language model decoding with speculative sampling. *arXiv preprint arXiv:2302.01318*, 2023. URL https://arxiv.org/pdf/2302.01318.

[15] Derek Chen, Howard Chen, Yi Yang, Alexander Lin, and Zhou Yu. Action-based conversations dataset: A corpus for building more in-depth task-oriented dialogue systems. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3002–3017, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.239. URL https://aclanthology.org/2021.naacl-main.239.

[16] Jack Choquette, Wishwesh Gandhi, Olivier Giroux, Nick Stam, and Ronny Krashinsky. Nvidia a100

tensor core GPU: Performance and innovation. *IEEE Micro*, 41(2):29–35, 2021. doi: 10.1109/MM.2021.3061394.

[17] Krzysztof Marcin Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Quincy Davis, Afroz Mohiuddin, Lukasz Kaiser, David Benjamin Belanger, Lucy J Colwell, and Adrian Weller. Rethinking attention with performers. In *Proceedings of International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=Ua6zuk0WRH.

[18] Peng Chu, Xiao Bian, Shaopeng Liu, and Haibin Ling. Feature space augmentation for long-tailed data. In *Proceedings of Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXIX 16*, pages 694–710. Springer, 2020.

[19] Elizabeth Clark, Tal August, Sofia Serrano, Nikita Haduong, Suchin Gururangan, and Noah A. Smith. All that's 'human' is not gold: Evaluating human evaluation of generated text. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli, editors, *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)*, pages 7282–7296, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.565. URL https://aclanthology.org/2021.acl-long.565.

[20] Mark G Core and James F Allen. Coding dialogs with the DAMSL annotation scheme. In *Proceedings of Working Notes AAAI Fall Symp. Commun. Action in Humans*, 1997.

[21] Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. FlashAttention: Fast and memory-efficient exact attention with IO-awareness. In *Proceedings of Advances in Neural Information Processing Systems*, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/67d57c32e20fd0a7a302cb81d36e40d5-Paper-Conference.pdf.

[22] Michiel de Jong, Yury Zemlyanskiy, Joshua Ainslie, Nicholas FitzGerald, Sumit Sanghai, Fei Sha, and William Cohen. FiDO: Fusion-in-decoder optimized for stronger performance and faster in-

ference. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, editors, *Findings of the Association for Computational Linguistics: ACL 2023*, pages 11534–11547, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-acl.732. URL https://aclanthology.org/2023.findings-acl.732.

[23] Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. GPT3.int8(): 8-bit matrix multiplication for transformers at scale. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Proceedings of Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=dXiGWqBoxaD.

[24] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. QLoRA: Efficient finetuning of quantized LLMs. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=OUIFPHEgJU.

[25] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina 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)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://aclanthology.org/N19-1423.

[26] Yao Dou, Maxwell Forbes, Rik Koncel-Kedziorski, Noah A. Smith, and Yejin Choi. Is GPT-3 text indistinguishable from human text? Scarecrow: A framework for scrutinizing machine text. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio, editors, *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7250–7274, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.501. URL https://aclanthology.org/2022.acl-long.501.

[27] Bradley Efron and Robert Tibshirani. An introduction to the bootstrap. In *Chapman & Hall/CRC Monographs on Statistics & Applied Probability*. Taylor & Francis., 1993.

[28] G David Forney. The Viterbi algorithm. *Proceedings of the IEEE*, 61(3):268–278, 1973.

[29] Zihao Fu, Wai Lam, Qian Yu, Anthony Man-Cho So, Shengding Hu, Zhiyuan Liu, and Nigel Collier. Decoder-only or encoder-decoder? interpreting language model as a regularized encoder-decoder. *arXiv preprint arXiv:2304.04052*, 2023. URL https://arxiv.org/abs/2304.04052.

[30] Mor Geva, Yoav Goldberg, and Jonathan Berant. Are we modeling the task or the annotator? an investigation of annotator bias in natural language understanding datasets. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan, editors, *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1161–1166, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1107. URL https://aclanthology.org/D19-1107.

[31] Alexios Gidiotis and Grigorios Tsoumakas. A divide-and-conquer approach to the summarization of long documents. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 28:3029–3040, 2020. URL https://ieeexplore.ieee.org/document/9257174.

[32] Jianping Gou, Baosheng Yu, Stephen J Maybank, and Dacheng Tao. Knowledge distillation: A survey. *International Journal of Computer Vision*, 129:1789–1819, 2021. URL https://arxiv.org/abs/2006.05525.

[33] Morgan Gray, Jaromir Savelka, Wesley Oliver, and Kevin Ashley. Can GPT alleviate the burden of annotation? In *Legal Knowledge and Information Systems*, pages 157–166. IOS Press, 2023.

[34] Barbara J. Grosz and Candace L. Sidner. Attention, intentions, and the structure of discourse. *Computational Linguistics*, 12(3):175–204, 1986. URL https://aclanthology.org/J86-3001.

[35] Mandy Guo, Joshua Ainslie, David Uthus, Santiago Ontanon, Jianmo Ni, Yun-Hsuan Sung, and Yinfei Yang. LongT5: Efficient text-to-text transformer for long sequences. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 724–736, Seattle, United States, July 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.findings-naacl.55. URL https://aclanthology.org/2022.findings-naacl.55.

[36] Som Gupta and S. K Gupta. Abstractive summarization: An overview of the state of the art. *Expert Systems with Applications*, 121:49–65, 2019. ISSN 0957-4174. doi: https://doi.org/10.1016/j.eswa.2018.12.011. URL https://www.sciencedirect.com/science/article/pii/S0957417418307735.

[37] Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel Bowman, and Noah A. Smith. Annotation artifacts in natural language inference data. In Marilyn Walker, Heng Ji, and Amanda Stent, editors, *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 107–112, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-2017. URL https://aclanthology.org/N18-2017.

[38] Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah 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*, pages 8342–8360, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.740. URL https://aclanthology.org/2020.acl-main.740.

[39] He He, Derek Chen, Anusha Balakrishnan, and Percy Liang. Decoupling strategy and generation in negotiation dialogues. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2333–2343, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1256. URL https://aclanthology.org/D18-1256.

[40] Sophie Henning, William Beluch, Alexander Fraser, and Annemarie Friedrich. A survey of methods for addressing class imbalance in deep-learning based natural language processing. In Andreas Vlachos and Isabelle Augenstein, editors, *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 523–540, Dubrovnik, Croatia, May 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.eacl-main.38. URL https://aclanthology.org/2023.eacl-main.38.

[41] Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. In

Proceedings of Advances in Neural Information Processing Systems Deep Learning and Representation Learning Workshop, 2015. URL http://arxiv.org/abs/1503.02531.

[42] Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *Proceedings of International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

[43] Yushi Hu, Chia-Hsuan Lee, Tianbao Xie, Tao Yu, Noah A. Smith, and Mari Ostendorf. In-context learning for few-shot dialogue state tracking. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang, editors, *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 2627–2643, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.findings-emnlp.193. URL https://aclanthology.org/2022.findings-emnlp.193.

[44] Cheng-Zhi Anna Huang, Ashish Vaswani, Jakob Uszkoreit, Ian Simon, Curtis Hawthorne, Noam Shazeer, Andrew M. Dai, Matthew D. Hoffman, Monica Dinculescu, and Douglas Eck. Music transformer. In *Proceedings of International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=rJe4ShAcF7.

[45] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3):535–547, 2019.

[46] Rishabh Joshi, Vidhisha Balachandran, Shikhar Vashishth, Alan Black, and Yulia Tsvetkov. DialoGraph: Incorporating interpretable strategy-graph networks into negotiation dialogues. In *Proceedings of International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=kDnal_bbb-E.

[47] Dan Jurafsky, Elizabeth Shriberg, , and Debra Biasca. Switchboard SWBD-DAMSL shallow-discourse-function annotation coders manual, draft 13. Technical report, University of Colorado, Boulder, 1997.

[48] Jordan Juravsky, Bradley Brown, Ryan Ehrlich, Daniel Y Fu, Christopher Ré, and Azalia Mirhoseini. Hydragen: High-throughput LLM inference with shared prefixes. *arXiv preprint arXiv:2402.05099*, 2024. URL https://arxiv.org/abs/2402.05099.

[49] J. F. Kelley. An iterative design methodology for user-friendly natural language office information applications. *ACM Trans. Inf. Syst.*, 2(1):26–41, jan 1984. ISSN 1046-8188. doi: 10.1145/357417.357420. URL https://doi.org/10.1145/357417.357420.

[50] Hyunwoo Kim, Jack Hessel, Liwei Jiang, Peter West, Ximing Lu, Youngjae Yu, Pei Zhou, Ronan Bras, Malihe Alikhani, Gunhee Kim, Maarten Sap, and Yejin Choi. SODA: Million-scale dialogue distillation with social commonsense contextualization. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 12930–12949, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.799. URL https://aclanthology.org/2023.emnlp-main.799.

[51] Sneha Kudugunta, Yanping Huang, Ankur Bapna, Maxim Krikun, Dmitry Lepikhin, Minh-Thang Luong, and Orhan Firat. Beyond distillation: Task-level mixture-of-experts for efficient inference. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih, editors, *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 3577–3599, Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.findings-emnlp.304. URL https://aclanthology.org/2021.findings-emnlp.304.

[52] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, pages 611–626, 2023. URL https://dl.acm.org/doi/10.1145/3600006.3613165.

[53] Sophia Lam, Charles Chen, Kristi Kim, George Wilson, J Holt Crews, and Matthew S Gerber. Optimizing customer-agent interactions with natural language processing and machine learning. In *Pro-*

ceedings of Systems and Information Engineering Design Symposium (SIEDS), pages 1–6. IEEE, 2019.

[54] Chia-Hsuan Lee, Hao Cheng, and Mari Ostendorf. Dialogue state tracking with a language model using schema-driven prompting. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 4937–4949, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.404. URL https://aclanthology.org/2021.emnlp-main.404.

[55] Eric Lehman, Evan Hernandez, Diwakar Mahajan, Jonas Wulff, Micah J Smith, Zachary Ziegler, Daniel Nadler, Peter Szolovits, Alistair Johnson, and Emily Alsentzer. Do we still need clinical language models? In Bobak J. Mortazavi, Tasmie Sarker, Andrew Beam, and Joyce C. Ho, editors, *Proceedings of the Conference on Health, Inference, and Learning*, volume 209 of *Proceedings of Machine Learning Research*, pages 578–597. PMLR, 22 Jun–24 Jun 2023. URL https://proceedings.mlr.press/v209/eric23a.html.

[56] Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *Proceedings of International Conference on Machine Learning*, pages 19274–19286. PMLR, 2023. URL https://proceedings.mlr.press/v202/leviathan23a/leviathan23a.pdf.

[57] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.703. URL https://aclanthology.org/2020.acl-main.703.

[58] Haonan Li, Martin Tomko, Maria Vasardani, and Timothy Baldwin. MultiSpanQA: A dataset for multi-span question answering. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1250–1260, Seattle, United States, July 2022. Association for Computational Linguistics.

doi: 10.18653/v1/2022.naacl-main.90. URL https://aclanthology.org/2022.naacl-main.90.

[59] Zhuoyan Li, Hangxiao Zhu, Zhuoran Lu, and Ming Yin. Synthetic data generation with large language models for text classification: Potential and limitations. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 10443–10461, Singapore, December 2023. Association for Computational Linguistics. URL https://aclanthology.org/2023.emnlp-main.647.

[60] J. C. R. Licklider. Man-computer symbiosis. *IRE Transactions on Human Factors in Electronics*, HFE-1(1):4–11, 1960. doi: 10.1109/THFE2.1960.4503259. URL https://ieeexplore.ieee.org/document/4503259.

[61] Chin-Yew Lin. ROUGE: A package for automatic evaluation of summaries. In *Proceedings of Text Summarization Branches Out*, pages 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics. URL https://aclanthology.org/W04-1013.

[62] Hui Lin and Vincent Ng. Abstractive summarization: A survey of the state of the art. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 9815–9822, 2019. URL https://ojs.aaai.org/index.php/AAAI/article/view/5056.

[63] Alisa Liu, Swabha Swayamdipta, Noah A. Smith, and Yejin Choi. WANLI: Worker and AI collaboration for natural language inference dataset creation. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 6826–6847, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. URL https://aclanthology.org/2022.findings-emnlp.508.

[64] Liu Liu, Zheng Qu, Zhaodong Chen, Fengbin Tu, Yufei Ding, and Yuan Xie. Dynamic sparse attention for scalable transformer acceleration. *IEEE Transactions on Computers*, 71:3165–3178, 2022. URL https://api.semanticscholar.org/CorpusID:252421345.

[65] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining

approach. *arXiv preprint arXiv:1907.11692*, 2019. URL https://arxiv.org/abs/1907.11692.

[66] Bo-Ru Lu, Frank Shyu, Yun-Nung Chen, Hung-Yi Lee, and Lin-Shan Lee. Order-preserving abstractive summarization for spoken content based on connectionist temporal classification. In *Proceedings of Interspeech*, pages 2899–2903, 2017. doi: 10.21437/Interspeech.2017-862. URL http://dx.doi.org/10.21437/Interspeech.2017-862.

[67] Bo-Ru Lu, Yushi Hu, Hao Cheng, Noah A. Smith, and Mari Ostendorf. Unsupervised learning of hierarchical conversation structure. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang, editors, *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 5657–5670, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.findings-emnlp.415. URL https://aclanthology.org/2022.findings-emnlp.415.

[68] Bo-Ru Lu, Nikita Haduong, Chia-Hsuan Lee, Zeqiu Wu, Hao Cheng, Paul Koester, Jean Utke, Tao Yu, Noah A. Smith, and Mari Ostendorf. Does collaborative human-lm dialogue generation help information extraction from human-human dialogues? In *Proceedings of Conference on Language Model (CoLM)*, 2024. URL https://arxiv.org/abs/2307.07047.

[69] Bo-Ru Lu, Nikita Haduong, Chien-Yu Lin, Hao Cheng, Noah A Smith, and Mari Ostendorf. Encode once and decode in parallel: Efficient transformer decoding. *arXiv preprint arXiv:2403.13112*, 2024. URL https://arxiv.org/abs/2403.13112.

[70] Rui Meng, Khushboo Thaker, Lei Zhang, Yue Dong, Xingdi Yuan, Tong Wang, and Daqing He. Bringing structure into summaries: A faceted summarization dataset for long scientific documents. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli, editors, *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 1080–1089, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-short.137. URL https://aclanthology.org/2021.acl-short.137.

[71] N. Moratanch and S. Chitrakala. A survey on extractive text summarization. In *Proceedings of International Conference on Computer, Communication and Signal Processing (ICCCSP)*, pages 1–6, 2017. doi: 10.1109/ICCCSP.2017.7944061.

[72] Kevin Murphy. *Machine Learning: A Probabilistic Perspective*. MIT Press, 2012.

[73] Xuefei Ning, Zinan Lin, Zixuan Zhou, Zifu Wang, Huazhong Yang, and Yu Wang. Skeleton-of-thought: Prompting LLMs for efficient parallel generation. In *Proceedings of International Conference on Learning Representations*, 2024. URL `https://openreview.net/forum?id=mqVgBbNCm9`.

[74] OpenAI. Gpt-4 technical report, 2024.

[75] Mari Ostendorf and Harald Singer. HMM topology design using maximum likelihood successive state splitting. *Computer Speech & Language*, 11(1):17–41, 1997.

[76] Raghavendra Pappagari, Piotr Zelasko, Jesús Villalba, Yishay Carmiel, and Najim Dehak. Hierarchical transformers for long document classification. In *Proceedings of IEEE automatic speech recognition and understanding workshop (ASRU)*, pages 838–844. IEEE, 2019.

[77] Joon Sung Park, Lindsay Popowski, Carrie Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. Social simulacra: Creating populated prototypes for social computing systems. In *Proceedings of the 35th Annual ACM Symposium on User Interface Software and Technology*, pages 1–18, 2022.

[78] Noseong Park, Mahmoud Mohammadi, Kshitij Gorde, Sushil Jajodia, Hongkyu Park, and Youngmin Kim. Data synthesis based on generative adversarial networks. *Proc. VLDB Endow.*, 11 (10):1071–1083, jun 2018. ISSN 2150-8097. doi: 10.14778/3231751.3231757. URL `https://doi.org/10.14778/3231751.3231757`.

[79] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An

imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Proceedings of Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[80] Piotr Pęzik, Gosia Krawentek, Sylwia Karasińska, Paweł Wilk, Paulina Rybińska, Anna Cichosz, Angelika Peljak-Łapińska, Mikołaj Deckert, and Michał Adamczyk. DiaBiz – an annotated corpus of Polish call center dialogs. In Nicoletta Calzolari, Frédéric Béchet, Philippe Blache, Khalid Choukri, Christopher Cieri, Thierry Declerck, Sara Goggi, Hitoshi Isahara, Bente Maegaard, Joseph Mariani, Hélène Mazo, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pages 723–726, Marseille, France, June 2022. European Language Resources Association. URL https://aclanthology.org/2022.lrec-1.76.

[81] Kun Qian, Ahmad Beirami, Zhouhan Lin, Ankita De, Alborz Geramifard, Zhou Yu, and Chinnadhurai Sankar. Annotation inconsistency and entity bias in MultiWOZ. In Haizhou Li, Gina-Anne Levow, Zhou Yu, Chitralekha Gupta, Berrak Sisman, Siqi Cai, David Vandyke, Nina Dethlefs, Yan Wu, and Junyi Jessy Li, editors, *Proceedings of the 22nd Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 326–337, Singapore and Online, July 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.sigdial-1.35. URL https://aclanthology.org/2021.sigdial-1.35.

[82] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training, 2018.

[83] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[84] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020. URL http://jmlr.org/papers/v21/20-074.html.

[85] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551, 2020.

[86] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas, November 2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1264. URL https://aclanthology.org/D16-1264.

[87] Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 3505–3506, 2020.

[88] Abhinav Rastogi, Xiaoxue Zang, Srinivas Sunkara, Raghav Gupta, and Pranav Khaitan. Towards scalable multi-domain conversational agents: The schema-guided dialogue dataset. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8689–8696, 2020.

[89] Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan, editors, *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1410. URL https://aclanthology.org/D19-1410.

[90] Aurko Roy, Mohammad Saffar, Ashish Vaswani, and David Grangier. Efficient Content-Based Sparse Attention with Routing Transformers. *Transactions of the Association for Computational Linguistics*, 9:53–68, 02 2021. ISSN 2307-387X. doi: 10.1162/tacl_a_00353. URL https://doi.org/10.1162/tacl_a_00353.

[91] Andrea Santilli, Silvio Severino, Emilian Postolache, Valentino Maiorca, Michele Mancusi, Ric-

cardo Marin, and Emanuele Rodola. Accelerating transformer inference for translation via parallel decoding. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12336–12355, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.689. URL https://aclanthology.org/2023.acl-long.689.

[92] Bishal Santra, Potnuru Anusha, and Pawan Goyal. Hierarchical transformer for task oriented dialog systems. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5649–5658, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.449. URL https://aclanthology.org/2021.naacl-main.449.

[93] Jacob Schreiber. Pomegranate: fast and flexible probabilistic modeling in python. *Journal of Machine Learning Research*, 18(164):1–6, 2018.

[94] Guokan Shang, Wensi Ding, Zekun Zhang, Antoine Tixier, Polykarpos Meladianos, Michalis Vazirgiannis, and Jean-Pierre Lorré. Unsupervised abstractive meeting summarization with multi-sentence compression and budgeted submodular maximization. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 664–674, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-1062. URL https://aclanthology.org/P18-1062.

[95] Ashish Sharma, Inna Wanyin Lin, Adam S. Miner, David C. Atkins, and Tim Althoff. Human–ai collaboration enables more empathic conversations in text-based peer-to-peer mental health support. *Nature Machine Intelligence*, 5:46–57, 2022. URL https://api.semanticscholar.org/CorpusID:247778407.

[96] Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. Self-attention with relative position representations. In Marilyn Walker, Heng Ji, and Amanda Stent, editors, *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 464–468, New Orleans, Louisiana, June

2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-2074. URL https://aclanthology.org/N18-2074.

[97] Noam Shazeer. Fast transformer decoding: One write-head is all you need. *arXiv preprint arXiv:1911.02150*, 2019. URL https://arxiv.org/abs/1911.02150.

[98] Sarvesh Soni, Meghana Gudala, Atieh Pajouhi, and Kirk Roberts. RadQA: A question answering dataset to improve comprehension of radiology reports. In Nicoletta Calzolari, Frédéric Béchet, Philippe Blache, Khalid Choukri, Christopher Cieri, Thierry Declerck, Sara Goggi, Hitoshi Isahara, Bente Maegaard, Joseph Mariani, Hélène Mazo, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pages 6250–6259, Marseille, France, June 2022. European Language Resources Association. URL https://aclanthology.org/2022.lrec-1.672.

[99] Felix Stahlberg and Shankar Kumar. Synthetic data generation for grammatical error correction with tagged corruption models. In *Proceedings of the 16th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 37–47, Online, April 2021. Association for Computational Linguistics. URL https://aclanthology.org/2021.bea-1.4.

[100] Ramya Tekumalla and Juan M. Banda. Leveraging large language models and weak supervision for social media data annotation: An evaluation using covid-19 self-reported vaccination tweets. In Hirohiko Mori, Yumi Asahi, Adela Coman, Simona Vasilache, and Matthias Rauterberg, editors, *HCI International 2023 – Late Breaking Papers*, pages 356–366, Cham, 2023. Springer Nature Switzerland. ISBN 978-3-031-48044-7.

[101] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[102] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and

fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023. URL https://arxiv.org/abs/2307.09288.

[103] Adaku Uchendu, Jooyoung Lee, Hua Shen, Thai Le, Dongwon Lee, et al. Does human collaboration enhance the accuracy of identifying LLM-generated deepfake texts? In *Proceedings of the AAAI Conference on Human Computation and Crowdsourcing*, volume 11, pages 163–174, 2023.

[104] Takuma Udagawa, Aashka Trivedi, Michele Merler, and Bishwaranjan Bhattacharjee. A comparative analysis of task-agnostic distillation methods for compressing transformer language models. In Mingxuan Wang and Imed Zitouni, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 20–31, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-industry.3. URL https://aclanthology.org/2023.emnlp-industry.3.

[105] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Proceedings of Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[106] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *Proceedings of International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=rJXMpikCZ.

[107] Sinong Wang, Belinda Z Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*, 2020. URL https://arxiv.org/abs/2006.04768.

[108] Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. Self-instruct: Aligning language models with self-generated instructions. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, editors, *Proceedings of the 61st Annual*

Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 13484–13508, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.754. URL https://aclanthology.org/2023.acl-long.754.

[109] Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. Emergent abilities of large language models. *Transactions on Machine Learning Research*, 2022. ISSN 2835-8856. URL https://openreview.net/forum?id=yzkSU5zdwD. Survey Certification.

[110] Tsung-Hsien Wen, David Vandyke, Nikola Mrkšić, Milica Gašić, Lina M. Rojas-Barahona, Pei-Hao Su, Stefan Ultes, and Steve Young. A network-based end-to-end trainable task-oriented dialogue system. In Mirella Lapata, Phil Blunsom, and Alexander Koller, editors, *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 438–449, Valencia, Spain, April 2017. Association for Computational Linguistics. URL https://aclanthology.org/E17-1042.

[111] Jason Williams, Antoine Raux, Deepak Ramachandran, and Alan Black. The dialog state tracking challenge. In *Proceedings of the SIGDIAL 2013 Conference*, pages 404–413, Metz, France, August 2013. Association for Computational Linguistics. URL https://aclanthology.org/W13-4065.

[112] Samuel Williams, Andrew Waterman, and David Patterson. Roofline: an insightful visual performance model for multicore architectures. *Commun. ACM*, 52(4):65–76, apr 2009. ISSN 0001-0782. doi: 10.1145/1498765.1498785. URL https://doi.org/10.1145/1498765.1498785.

[113] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, October 2020. Association for Computa-

tional Linguistics. doi: 10.18653/v1/2020.emnlp-demos.6. URL https://aclanthology.org/2020.emnlp-demos.6.

[114] Zeqiu Wu, Bo-Ru Lu, Hannaneh Hajishirzi, and Mari Ostendorf. DIALKI: Knowledge identification in conversational systems through dialogue-document contextualization. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 1852–1863. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.emnlp-main.140. URL https://aclanthology.org/2021.emnlp-main.140.

[115] Zhewei Yao, Reza Yazdani Aminabadi, Minjia Zhang, Xiaoxia Wu, Conglong Li, and Yuxiong He. Zeroquant: Efficient and affordable post-training quantization for large-scale transformers. *Proceedings of Advances in Neural Information Processing Systems*, 35:27168–27183, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/adf7fa39d65e2983d724ff7da57f00ac-Paper-Conference.pdf.

[116] Fanghua Ye, Yue Feng, and Emine Yilmaz. ASSIST: Towards label noise-robust dialogue state tracking. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio, editors, *Findings of the Association for Computational Linguistics: ACL 2022*, pages 2719–2731, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.findings-acl.214. URL https://aclanthology.org/2022.findings-acl.214.

[117] Fanghua Ye, Jarana Manotumruksa, and Emine Yilmaz. MultiWOZ 2.4: A multi-domain task-oriented dialogue dataset with essential annotation corrections to improve state tracking evaluation. In Oliver Lemon, Dilek Hakkani-Tur, Junyi Jessy Li, Arash Ashrafzadeh, Daniel Hernández Garcia, Malihe Alikhani, David Vandyke, and Ondřej Dušek, editors, *Proceedings of the 23rd Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 351–360, Edinburgh, UK, September 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.sigdial-1.34. URL https://aclanthology.org/2022.sigdial-1.34.

[118] Xiaoju Ye. calflops: a FLOPs and params calculate tool for neural networks in pytorch framework. 2023. URL https://github.com/MrYxJ/calculate-flops.pytorch.

[119] Zihao Ye, Lequn Chen, Ruihang Lai, Yilong Zhao, Size Zheng, Junru Shao, Bohan Hou, Hongyi Jin, Yifei Zuo, Liangsheng Yin, Tianqi Chen, and Luis Ceze. Accelerating self-attentions for LLM serving with flashinfer, February 2024. URL https://flashinfer.ai/2024/02/02/introduce-flashinfer.html.

[120] Zihao Ye, Ruihang Lai, Bo-Ru Lu, Chien-Yu Lin, Size Zheng, Lequn Chen, Tianqi Chen, and Luis Ceze. Cascade inference: Memory bandwidth efficient shared prefix batch decoding, February 2024. URL https://flashinfer.ai/2024/02/02/cascade-inference.html.

[121] Wen-wai Yim, Yujuan Fu, Asma Ben Abacha, Neal Snider, Thomas Lin, and Meliha Yetisgen. Aci-bench: a novel ambient clinical intelligence dataset for benchmarking automatic visit note generation. *Scientific Data*, 10(1):586, 2023. URL https://www.nature.com/articles/s41597-023-02487-3.pdf.

[122] Ali Hadi Zadeh, Isak Edo, Omar Mohamed Awad, and Andreas Moshovos. Gobo: Quantizing attention-based nlp models for low latency and energy efficient inference. In *2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 811–824. IEEE, 2020. URL https://microarch.org/micro53/papers/738300a811.pdf.

[123] Yifan Zhang, Bingyi Kang, Bryan Hooi, Shuicheng Yan, and Jiashi Feng. Deep long-tailed learning: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(9):10795–10816, 2023. doi: 10.1109/TPAMI.2023.3268118. URL https://ieeexplore.ieee.org/document/10105457.

[124] Yusen Zhang, Ansong Ni, Ziming Mao, Chen Henry Wu, Chenguang Zhu, Budhaditya Deb, Ahmed Awadallah, Dragomir Radev, and Rui Zhang. Summ^n: A multi-stage summarization framework for long input dialogues and documents. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio, editors, *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1592–1604, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.112. URL https://aclanthology.org/2022.acl-long.112.

[125] Jeffrey Zhao, Raghav Gupta, Yuan Cao, Dian Yu, Mingqiu Wang, Harrison Lee, Abhinav Rastogi, Izhak Shafran, and Yonghui Wu. Description-driven task-oriented dialog modeling. *arXiv preprint arXiv:2201.08904*, 2022. URL https://arxiv.org/abs/2201.08904.

[126] Yilong Zhao, Chien-Yu Lin, Kan Zhu, Zihao Ye, Lequn Chen, Size Zheng, Luis Ceze, Arvind Krishnamurthy, Tianqi Chen, and Baris Kasikci. Atom: Low-bit quantization for efficient and accurate LLM serving. *arXiv preprint arXiv:2310.19102*, 2023. URL https://arxiv.org/abs/2310.19102.

[127] Zhipeng Zhao, Kun Zhou, Xiaolei Wang, Wayne Xin Zhao, Fan Pan, Zhao Cao, and Ji-Rong Wen. Alleviating the long-tail problem in conversational recommender systems. In *Proceedings of the 17th ACM Conference on Recommender Systems*, RecSys '23, page 374–385, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9798400702419. doi: 10.1145/3604915.3608812. URL https://doi.org/10.1145/3604915.3608812.

[128] Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Jeff Huang, Chuyue Sun, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E Gonzalez, et al. Efficiently programming large language models using SGLang. *arXiv preprint arXiv:2312.07104*, 2023. URL https://arxiv.org/abs/2312.07104.

[129] Ming Zhong, Pengfei Liu, Yiran Chen, Danqing Wang, Xipeng Qiu, and Xuanjing Huang. Extractive summarization as text matching. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault, editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6197–6208, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.552. URL https://aclanthology.org/2020.acl-main.552.

[130] Sitong Zhou, Meliha Yetisgen, and Mari Ostendorf. Building blocks for complex tasks: Robust generative event extraction for radiology reports under domain shifts. In Tristan Naumann, Asma Ben Abacha, Steven Bethard, Kirk Roberts, and Anna Rumshisky, editors, *Proceedings of the 5th Clinical Natural Language Processing Workshop*, pages 344–357, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.clinicalnlp-1.38. URL https://aclanthology.org/2023.clinicalnlp-1.38.

[131] Yiheng Zhou, Yulia Tsvetkov, Alan W Black, and Zhou Yu. Augmenting non-collaborative dialog systems with explicit semantic and strategic dialog history. In *Proceedings of International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=ryxQuANKPB.

Chapter A

Appendix

A.1 User Interface for Data Collection for DIALGEN

We list two main pages of our interface for dialogue generation. They are editing and labeling steps.

First, the editing step (Figure A.1) page provides dialogue scenarios (slot value pairs), dialogue history, extracted tuples (annotated entity-slot-value triplets), instruction for regeneration, and current subdialogue for editing. A human reviewer can provide an instruction to guide the LM to generate a desired subdialogue to replace the current subdialogue. If the current subdialogue is satisfied with the reviewer, they can edit turns to fix the minor errors in the subdialogue.

Second, the labeling step page (Figure A.2) is an optional page for the DIALGEN framework. This page is designed for the dialogue state tracking task where the human reviewer can annotate and edit the subdialogue in the previous editing step. Note that the labeling step can be fully decoupled from the framework.

The human reviewer will iteratively collaborate with the LM to generate and revise the subdialogues and to annotate the subdialogues until the end of the dialogue.

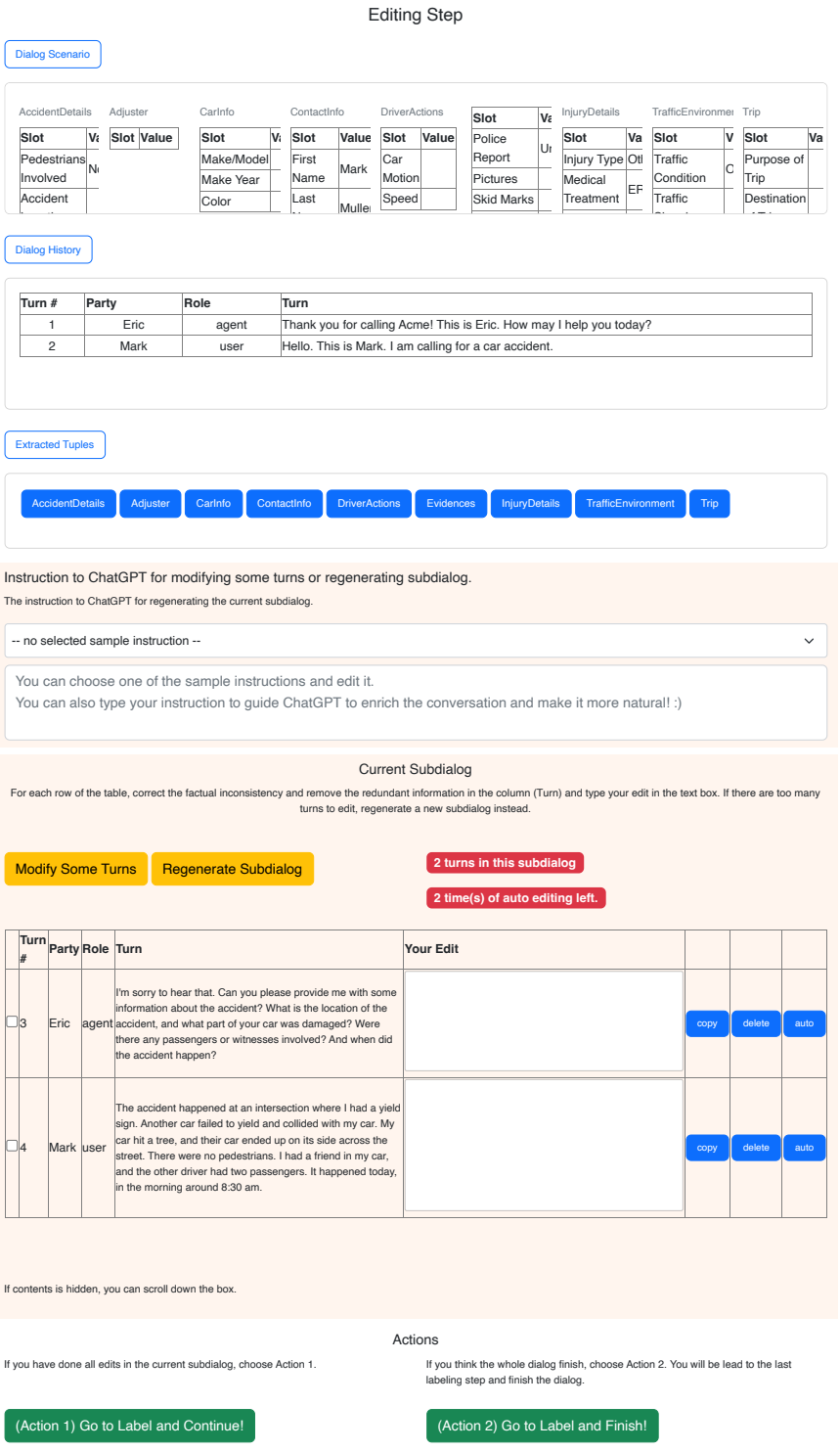


Figure A.1: The first step in DIALGEN is to create the subdialogue. A dialogue scenario table is provided to indicate slots expected to appear in the conversation. A human reviewer selects LM-generated text and edit it as needed. They can also ask the LM to regenerate selected turns or the full subdialogue and optionally provide extra instructions to guide the LM's generation process.

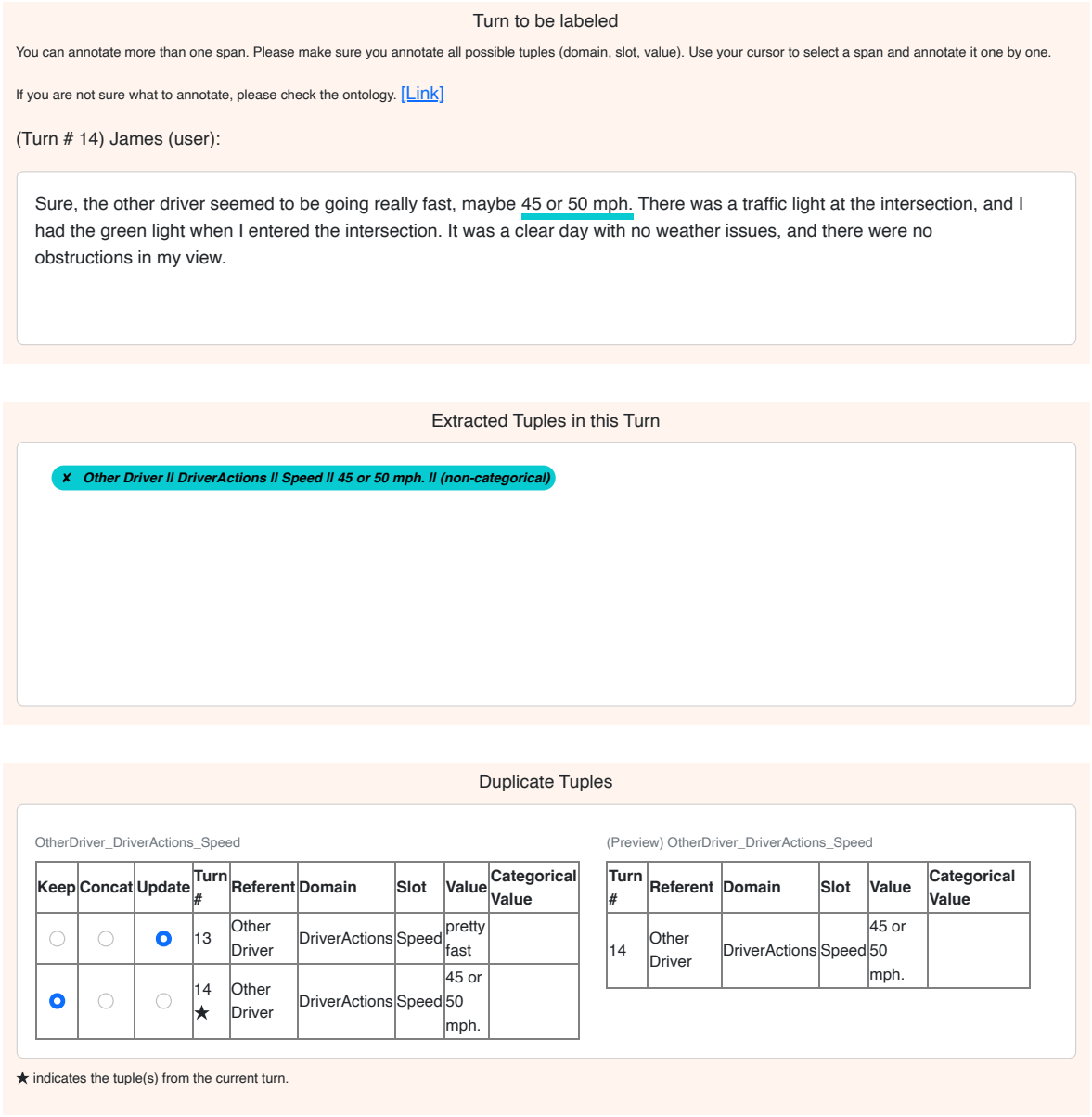


Figure A.2: A human reviewer selects a span and label it. If there exists a duplicate label, they are prompted to resolve the conflict by selecting to update (as shown), concat, or keep multiple labels.