FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

MASTER THESIS

Michelle Elizabeth

Conversational Agents for Task-Oriented Dialogue

Institute of Formal and Applied Linguistics

Supervisors of the master thesis: Lina M. Rojas-Barahona, Orange Innovation, France
Ondřej Dušek, Institute of Formal and Applied Linguistics, Charles University, Czechia
Miguel Couceiro, University of Lorraine, France

Study programme: Language Technologies and Computational Linguistics

Prague 2024

I declare that I carried out this master thesis on my own, and only with the cited sources, literature and other professional sources. I understand that my work relates to the rights and obligations under the Act No. 121/2000 Sb., the Copyright Act, as amended, in particular the fact that the Charles University has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 subsection 1 of the Copyright Act.

In .... date ....

Author's signature

To the Erasmus Mundus LCT administration and the LCT coordinators at Charles University and University of Lorraine, Marketa Lopatkova and Miguel Couceiro, for believing in me and offering me the opportunity to be part of this prestigious program, these two years have been a rollercoaster yet very fulfilling.

To my supervisors, Lina Rojas, Ondřej Dušek and Miguel Couceiro, thank you for your exceptional guidance and patience in me as I navigated this project and for your valuable feedback.

To my friends from Prague and Nancy – Mifta, Keenu, Fairooz, Anna, Nastya and Hannah – thank you for the good times and the memories, I will always cherish them. Also, to my friends back home, especially to Bhairavi and Irene, for motivating and pushing me and for being there through it all, thank you.

Last but not the least, to Appa, Amma and Maria for their unconditional love and support, without whom I would not be where I am today, I dedicate this master thesis to you.

Title: Conversational Agents for Task-Oriented Dialogue

Author: Michelle Elizabeth

Institute: Institute of Formal and Applied Linguistics

Supervisors: Lina M. Rojas-Barahona, Orange Innovation, France; Ondřej Dušek, Institute of Formal and Applied Linguistics, Charles University, Czechia; Miguel Couceiro, University of Lorraine, France

Abstract: Large language models have gained immense popularity due to their text generation and reasoning capabilities on a wide range of tasks. They even show promise in solving complex tasks that conventionally require deep learning techniques, using advanced prompting strategies like ReAct (reasoning and acting). In this work, we apply the ReAct paradigm to prompt LLMs to perform task-oriented dialogue (ToD) in simulation, with access to external tools. We perform quantitative and qualitative analysis on the simulated dialogues but we see that our method does not meet the current benchmarks for ToD. However, LLMs have the potential to perform as a conversational agent for ToD with more fine-grained instructions and tools.

Keywords: Task-oriented Dialogue, Large Language Models, Conversational Agents, ReAct

Contents

Introduction

'Alexa! Turn on the front porch lights at 6 p.m.'

Virtual assistants like Amazon's Alexa, Apple's Siri and the Google Assistant have become a household name in the last decade. We rely on these assistants everyday to perform a wide range of tasks from setting reminders to ordering groceries and controlling our home appliances. Virtual assistants make our lives easier by providing immediate access to information and services with simple voice or text commands. These advancements are thanks to the leaps and bounds in research in Natural Language Processing (NLP).

We have come a long way since the first chatbot ELIZA was introduced (Weizenbaum, 1966). ELIZA was a chatbot developed to pose as a psychotherapist – one of the first chatbots that made natural language communication with a machine possible. ELIZA was programmed to identify keywords and respond to the user based on templates. Since then, conversational bots have been a very active and dynamic area of research leading to more popular chatbots and assistants like Alexa, Siri and Google Assistant in the 2010s. Behind the scenes, most chatbots rely on classical dialogue systems that perform speech recognition, natural language understanding, dialogue management, natural language generation and text-to-speech generation (Sarikaya et al., 2016).

Fast-forward to 2022, we were introduced to ChatGPT (Ouyang et al., 2022), attributing its human-like conversational capabilities to the rise of Large Language Models (LLM). The hype around LLM-based chatbots has led to high public expectations. Many users have been led to believe that these systems can fully replicate human conversation and understand complex contexts seamlessly. While LLMs have made significant strides, there still remains a gap between their capabilities and imitating actual human conversation. This is because human conversation is a very complex phenomenon which is still being studied by scientists around the world.

Human conversations rely on shared context and background knowledge. Interpreting a message accurately requires understanding the context in which it was said and determining its relevance. Although LLMs can use previous conversation history, they lack the understanding that is inherent in humans to know what to say and when. Another aspect of human dialogue is that words and phrases can have multiple meanings. Humans frequently use implicit meanings, metaphors, idiomatic expressions and anaphora which is very challenging for a machine to detect. Conversations are also rich with social cues conveyed through tone, pitch, body language and facial expressions. Accurately interpreting these cues is essential for meaningful interactions and using just text to communicate cannot convey the full meaning to the other party.

Although human conversation is very difficult, we have seen LLMs being able to chat with humans as if they were a real person. They have even shown immense potential in performing tasks they were not specifically trained for. Research has also shown that they are capable of reasoning (Yao; J. Zhao, et al., 2022), leading us to believe that LLMs can in fact do much more than just chat, if given the right tools.

In this work, we aim to study the reasoning and acting capabilities of LLMs to perform Task-oriented Dialogue (ToD). ToD systems are conversational agents that help the user achieve a specific goal like booking a flight or finding information about a restaurant. We want to study how well the LLM is able to understand the user requirements and use the right tools to retrieve the domain-specific information in order to successfully complete the task set by the user. We want to evaluate our system in simulation to measure how well it would perform in a real-world scenario since dialogue is dynamic and each turn depends on the previous turns generated (Rieser; Lemon, 2011).

This thesis is divided into 5 chapters. Chapter 1 gives an overview of task-oriented dialogue systems, their architecture, benchmarks and evaluation metrics followed by Chapter 2 which explores LLMs in detail. Chapter 3 describes our experiments and the setup developed to study the capabilites of LLMs as a task-oriented conversational agent and Chapter 4 analyses the results we obtained. Chapter 5 gives insight into the results and anticipates potential improvements and future work.

1 Task-Oriented Dialogue Systems

Dialogue system or conversational Artificial Intelligence (AI) is a rapidly evolving topic in NLP with systems that can chit-chat with the users on any topic to those that can communicate about very specific and niche topics. Task-oriented dialogue systems are a class of dialogue systems wherein the main focus is to help the user achieve a fixed goal. These systems are useful in different fields like tourism, sales, customer service, hospitality to name a few (Niculescu et al., 2014; Pellom et al., 2001; Athikkal; Jenq, 2022; Chai et al., 2001). The user interacts with the system which helps him access information, book services, manage reservations and so on.

In this chapter, we will go into the details of a classical ToD system pipeline. Section 1.1 describes their architecture and main components. Section 1.1.5 takes a brief look at end-to-end architectures for ToD. Section 1.2 explores the existing benchmarks for task-oriented dialogue. Section 1.3 talks about various evaluation metrics that are used to gauge the performance of ToD systems.

1.1 Architecture

The classical dialogue system pipeline draws inspiration from human conversation and tries to adapt it into various modules or components. In the context of a human-human conversation, the first step is for one of the speakers to utter a sentence in natural language. This utterance is perceived by the second party in the conversation who then tries to put the dialogue into context and understand what the speaker is trying to convey. The listener then forms the response taking into account common knowledge and previous conversation history. The response is then conveyed back to the first speaker in natural language and the conversation continues with the first speaker perceiving this new information and responding with a new utterance (Jurafsky; Martin, 2000; McTear, 2020).

Dialogue systems are similar and have various modules for each task. While the exact process behind a human conversation is not known, dialogue systems try to imitate them by defining the following tasks. The first task is to convert the speech signals of the user utterance into natural language text by using an automatic speech recognition module. The text is then passed through a natural language understanding module that converts the natural language utterance into a semantic representation that the system can process easily. The next step involves the dialogue management module which decides the next course of action in the conversation such as whether to access a heterogeneous source of knowledge like a database, knowledge graph or documents and deciding the content of the response. Once the content of the dialogue is determined by the dialogue manager, a natural language generation module converts the semantic response into a natural language response. Finally, a text-to-speech module converts the text into speech (Jurafsky; Martin, 2000; McTear, 2020).

Since we are not utilizing the speech modules, namely, automatic speech recognition and text-to-speech, we will start describing the natural language

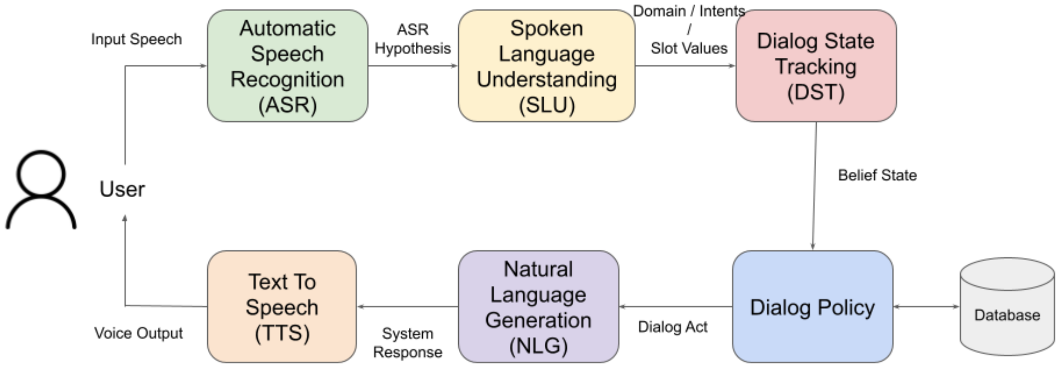

Figure 1.1 A classical Task-oriented Dialogue pipeline

understanding, dialogue manager and natural language generation modules in detail in the subsequent sections. Figure 1.1 shows a high level overview of a classical task-oriented dialogue system.

1.1.1 Natural Language Understanding

The Natural Language Understanding (NLU) module is used for extracting meaning from the user utterance. It is responsible for identifying the domain, dialogue acts, the slots and their corresponding values and converting them into a semantic representation.

The *domain* refers to the broader topic of the conversation. For example, if the user is asking for a restaurant in the city centre, then the NLU module identifies the domain as restaurant from the list of possible domains that the system is designed to talk about. The *dialogue act* or *intent* refers to the action performed by speakers when uttering sentences such as informing, requesting, clarifying, promising and so on. In our previous example, the dialogue acts are 'request'and 'inform'since the user is requesting the system for a restaurant and informing that it should be in the city centre. Slots and values are the concepts of the domain that the system needs to take into consideration in order to form the right response. In the example of restaurants, slots can be the name of the restaurant, its address, the cuisine available, the price range and so on.

NLU can be performed using a wide range of methods. These include handcrafted methods and machine learning approaches. Handcrafted (HDC) or rule-based methods include using regular expressions and grammars for domain detection, slot identification and filling. Machine learning methods include using classifiers for detecting slots and values. Methods employing neural networks for classification (Raffel; Ellis, 2015), Recurrent Neural Networks (RNN) (Liu; Lane, 2016; Hakkani-Tür et al., 2016; L. Zhao; Feng, 2018; Rojas-Barahona et al., 2016) and pre-trained language models for slot tagging and filling (Q. Chen et al., 2019; C.-S. Wu et al., 2020) are very popular for NLU. Generative models like T5 (Shah et al., 2023) and BART (Ahmad et al., 2021) have also been used for NLU. More recently, prompting LLMs for slot tagging (Hudeček; Dusek, 2023) is also gaining popularity.

1.1.2 Dialogue State Tracking

Dialogue State Tracking (DST) keeps track of the internal state of the dialogue in terms of the domains, intents, slots and values that were communicated by the user to the system and that the system responded with. It has the summary of the dialogue so far in the conversation. There are handcrafted methods as well as probabilistic methods for implementing DST.

The most common solution for implementing DST is using a probabilistic belief state. The belief state is the estimated probablity distribution over all possible states based on the observations and actions of the system. There are various implementations of a belief state tracker which use simple neural networks (Henderson et al., 2013), RNNs with different ranking algorithms (Mrkšić et al., 2015; Zilka; Jurcicek, 2015; Rastogi; Hakkani-Tur, et al., 2018), pretrained language models (Chao; Lane, 2019; Gao et al., 2019; Heck et al., 2020), generative models (Jacqmin; Rojas Barahona, et al., 2022) and more recently, by prompting LLMs (Hu et al., 2022) with more recent approaches integrating both NLU and DST.

1.1.3 Dialogue Management

The Dialogue Management (DM) module is responsible for making the system choose the optimal action based on the belief state generated by the DST module. The dialogue manager is usually responsible solely for the dialogue policy: selecting the optimal action. In other cases, the dialogue manager may consist both the *dialogue state tracker* and the *dialogue policy*. The dialogue policy decides the next action based on the history and the belief state of the conversation. The dialogue manager also might have access to backend services like a database server to retrieve information requested by the user, which is used within the policy to decide the appropriate action.

Dialogue Policy makes the decision on the next action the system should take, taking into consideration the current dialogue state. It has access to backend services and takes informed decisions based on the response from the external services. It controls the flow of the dialogue and tries to move the conversation towards the goal.

The simplest methods for action selection include finite state machines and rule-based methods. The most popular method is reinforcement learning, in which an agent is rewarded based on the action it selects, thereby learning the optimal actions to arrive at the goal. More advanced methods use neural networks for deep reinforcement learning in Deep Q-networks (Lipton et al., 2017) and policy gradient algorithms like REINFORCE (Wang et al., 2020), Actor-Critic (Su et al., 2017), Actor-Critic with Experience Replay (ACER) (Weisz et al., 2018; Cordier; Urvoy; Rojas-Barahona, et al., 2020) and Proximal Policy Optimization (PPO) (Tuan et al., 2018). More recently, structured policy learning with imitation learning was proposed by Cordier; Urvoy; Lefèvre, et al., 2022 to tackle multi-domain, multi-task ToD.

1.1.4 Natural Language Generation

Natural Language Generation (NLG) or response generation is the final step in the dialogue pipeline. It takes the semantic dialogue act as input and generates the final response to be given to the user in natural language. Very early solutions, which use templates to guide the NLG process (Weizenbaum, 1966) in dialogue systems is still very common (Kale; Rastogi, 2020). Grammar-based (Mille et al., 2019) and statistical approaches are also popular with the former using semantic structures such as trees and applying grammar rules to generate the response while the latter uses neural networks to generate the response based on the given dialogue act. Neural-network-based seq2seq (Wen; Gašić, et al., 2015) models that use encoder-decoder architectures like BART (Lewis et al., 2019), T5 (Raffel; Shazeer, et al., 2020) and those that use only decoders like GPT-2 (Radford; J. Wu, et al., 2019), GPT-3 (Brown et al., 2020) are widely used due to their ability to generate natural and fluent text. Lately, LLMs have become very popular in text generation tasks and this applies to the NLG step in ToD systems as well (Madotto et al., 2020). Approaches that combine NLG and NLU have also been explored (Tseng et al., 2020).

1.1.5 End-to-End Architectures for ToD

The classical pipeline of a task-oriented dialogue system follows a modular approach with various components serving a particular task as shown in Figure 1.1. While this approach improves the flexibility of the system since each component stands alone and can be replaced without affecting the other parts, they can easily accumulate errors down the pipeline leading to mediocre performance. In end-to-end ToD systems, there is little or no separation of components and they are trained as one single unit.

There are various approaches using supervised learning and reinforcement learning for building an end-to-end ToD system. Wen; Vandyke, et al., 2017 uses architectures that employ supervised learning for training an end-to-end system. There are also hybrid code networks (Williams et al., 2017) that use both supervised and reinforcement learning. Lei et al., 2018 describes a simple architecture that uses a two-stage decoding step. Pre-trained language models like GPT-2 are also very commonly used in many end-to-end architectures like SOLOIST (Peng et al., 2021), SimpleTOD (Hosseini-Asl et al., 2022) and NeuralPipeline (Ham et al., 2020).

1.2 ToD Benchmarks

Task-oriented dialogue (ToD) is a very active area of research and has evolved quite a lot over the years. Various benchmarks have been developed to assess the performance of ToD like PyDial (Ultes et al., 2017), ConvLab (Lee et al., 2019) and DialogStudio (J. Zhang et al., 2023). In this section, we will describe the most popular one, proposed for the DSTC9 Challenge (Gunasekara et al., 2020) called ConvLab, which is being used in this work.

1.2.1 ConvLab

ConvLab[1] (Lee et al., 2019) and its subsequent versions, ConvLab 2 and 3 (Q. Zhu; Z. Zhang, et al., 2020; Q. Zhu; Geishauser, et al., 2022) is a comprehensive toolkit designed for building, training, and evaluating conversational AI systems. It provides a standardized platform for researchers and developers to create and benchmark various dialogue system components.

ConvLab is an open-source framework and supports the entire pipeline and benchmarks of dialogue systems, including NLU, DST, dialogue policy learning, NLG and system evaluation. It has a modular architecture, allowing researchers to plug and play different components. The toolkit also includes popular dialogue datasets, pre-trained models and baseline systems, which can be used for various parts of the dialogue system pipeline.

ConvLab provides tools for benchmarking dialogue systems against standardized datasets and metrics. This allows for consistent evaluation and comparison of different systems and approaches. It also provides various user simulators to evaluate the developed systems under simulation and has been adapted to multiple dialogue datasets.

1.3 Metrics

Just like with any task, it is extremely important to evaluate the performance of the systems developed for task-oriented dialogue. Evaluation of task-oriented conversational systems can happen for various aspects - the quality of the generated responses, the information conveyed, the success of the task and so on. In this section, we describe the most common metrics that are used to evaluate the performance of a ToD system.

1.3.1 Automatic Evaluation Measures

Automatic evaluation measures quantify the performance of the system based on different aspects or the performance of different components. These metrics evaluate the system as a whole or each component of the system as a separate entity. Here we describe the task success rate and inform and book rates as the metrics to evaluate the full ToD pipeline in order to assess whether it achieves the user goals. NLU is usually measured by the F1 score and NLG modules can be evaluated by measuring BLEU (Papineni et al., 2002), GoogleBLEU (Y. Wu et al., 2016), METEOR (Banerjee; Lavie, 2005) and BertScore (T. Zhang et al., 2020) of the generated responses against the reference responses in the dataset. DST is measured using the Joint Goal Accuracy and the Slot Error Rate.

Task Success Rate

The task success rate is used to evaluate the system as a whole – it tries to answer whether the task-oriented system was able to achieve the user's goal. In a typical scenario, the task is considered successful if the system was able to inform

[1] `https://convlab.github.io/`

the user of the requested slots and was able to book the requested entities as specified in the goal.

Inform Rate and Book Rate

The inform rate and book rate are also used to measure the ToD system as a whole. The inform rate reports the precision, recall and F1 scores for the slots that were requested by the user in the goal. For example, if the user requested the name and address of a restaurant in the east part of town, then the system should inform the user of the same. The book rate reports the ratio of entities booked successfully by the system and the total number of bookings requested by the user (Lee et al., 2019).

BLEU and GoogleBLEU

The BiLingual Evaluation Understudy (BLEU) (Papineni et al., 2002) score is a very popular metric used for evaluating text generation against reference text. It measures the n-gram overlap between the generated text and the reference text. It is a precision-oriented metric and reports the score between 0 and 100, the higher the score the better the generated output. GoogleBLEU (Y. Wu et al., 2016) is a variation of the BLEU score and is mostly used for evaluating sentence pairs rather than the whole corpus like in BLEU. For NLG in a ToD system, BLEU scores helps measure the quality of the generated responses against the reference response, although this may not be the ideal metric since there are multiple ways to convey the same piece of information.

METEOR

Metric for Evaluation of Translation with Explicit ORdering (METEOR) (Banerjee; Lavie, 2005) is also a reference-based evaluation metric commonly used for evaluation of machine translation responses. In contrast to BLEU, which uses precision as the main factor in evaluating the responses, METEOR uses a harmonic mean of precision and recall with the recall weighted higher than precision. METEOR also aligns the words in the generated text with the words in the reference translation. This alignment is based on exact matches, stemming matches and semantic matches. This helps METEOR provide a more flexible and nuanced comparison than metrics like BLEU which rely solely on exact n-gram matches.

BertScore

BERTScore (T. Zhang et al., 2020) evaluates the quality of generated text by comparing it to a reference text using pre-trained BERT (Devlin et al., 2019) models. Unlike metrics like BLEU and METEOR, which rely on n-gram matching, BERTScore leverages BERT's contextual embeddings to assess semantic similarity. It calculates the cosine similarity between token embeddings of the generated text and the reference text to capture meaning and context.

Joint Goal Accuracy

The Joint Goal Accuracy (JGA) is used to measure the performance of a DST system. It is calculated as the number of correct state predictions that were reported in the entire dialogue. We look at exact matches in the turn-level state annotations in order to calculate the joint goal accuracy (Jacqmin; Druart, et al., 2023).

$$\text{JGA} = \frac{\text{Number of correct state predictions}}{\text{Total number of turns}}$$

Slot Error Rate

The Slot Error Rate (SER) is also a measure for evaluating the DST module. It is defined as the ratio between the number of slot-value pairs that were substituted, inserted or deleted, and the number of slot-value pairs in the reference state.

$$\text{SER} = \frac{\text{Insertions} + \text{Deletions} + \text{Substitutions}}{\text{Total number of slots in reference}}$$

This measure can also be adapted for NLG by delexicalising the slot values and counting the placeholders in the output.

1.3.2 Simulated Evaluation

Evaluation through simulation is widely used to measure the performance of ToD systems. Initially, this approach was implemented for Reinforcement Learning (RL) policy learning algorithms, which required simulated environments to optimize the policy. Given the dynamic nature of dialogue, it is difficult to replicate the same dialogues using a static corpus. Simulation, therefore, provides a controlled yet flexible environment to analyze and refine the performance of ToD systems (Rieser; Lemon, 2011).

1.3.3 Human Evaluation

While automatic evaluation is an important method to gauge the performance of a ToD system, in order to get the complete picture, we need feedback from the users of the system. This is where human evaluation comes into the picture. These measures will give the system designers an overview of how their systems perform in the real world. User satisfaction is extremely important and there are various methods to measure it.

Human evaluation is usually done by letting users chat with the system based on a given goal and then answering a questionnaire at the end of the dialogue. The questionnaire can ask questions regarding the task success, the helpfulness of the system, the fluency and coherence of the responses, the system's ability to recover from mistakes and so on.

The PARADISE framework (Walker et al., 1997) was one of the first evaluation frameworks for ToD. It is domain-independent and uses user ratings at the dialogue level. PARADISE integrates multiple performance measures into a single metric to assess their impact on user satisfaction, primarily using task success and dialogue cost metrics. More recent approaches (Takanobu et al., 2020; Santhanam; Shaikh,

2019) explore the best ways to perform human evaluation studying component-wise, single turn evaluation, various scales like Likert scale and so on.

Chapter Overview

In this chapter we discussed task-oriented dialogue systems in detail. We looked at the classical pipeline and components comprising a ToD system and touched upon various approaches that exist for each component. We also looked at the existing benchmarks for ToD, namely ConvLab and also explored simulated evaluation and various metrics available for evaluation – both automatic and human evaluation measures.

2 Large Language Models for ToD

In recent years, the emergence of LLMs, characterized by their vast size and deep learning architectures based on transformers (Vaswani et al., 2017), have demonstrated proficiency in a wide array of linguistic tasks. We are interested in the capabilities of LLMs as conversational agents, specifically for task-oriented dialogue.

This chapter examines the fundamental architecture of LLMs in Section 2.1. We then explore the history of Generative Pretrained Transformer (GPT) models in Section 2.2 and take a brief look at conversational agents and how LLMs tie in with agents in Section 2.3. Section 2.4.1 gives an overview of popular prompting strategies with a special focus on ReAct in Section 2.4.3, around which our experiments are designed.

2.1 Behind the Scenes of Large Language Models

Traditional NLP methods, which often relied on handcrafted rules and statistical models, struggled to grasp context and produce human-like text. The emergence of neural networks, particularly RNNs like Long Short Term Memory (LSTM) (Hochreiter; Schmidhuber, 1997) and Gated Recurrent Unit (GRU) (Cho et al., 2014) and Convolutional Neural Networks (CNN) (O'Shea; Nash, 2015), was a turning point towards better and robust models for language tasks.

Sequence-to-sequence models were used for language generation tasks and these used RNNs in an encoder-decoder architecture. But a major limitation for these models was the loss of information when dealing with long sequences. To overcome this, the attention mechanism was introduced. The idea behind the attention mechanism is that the neural network gives importance to parts of the input sequence that contain information relevant to the current generation step.

2.1.1 The Transformer Architecture

The transformer architecture, introduced by Vaswani et al., 2017 has been a gamechanger in NLP. The transformer (Vaswani et al., 2017) favours a parallelized approach that leverages self-attention mechanisms that allows it to process entire sequences of words simultaneously. This has led to significant improvements in training efficiency and the ability to capture long-range dependencies in text at training time. See the full transformer architecture in Figure 2.1.

Positional Encoding

Since the transformer processes input sequences in parallel rather than sequentially, it lacks a sense of word order. To address this, the architecture incorporates positional encodings, which are added to the input embeddings to provide information about the position of each word in the sequence. These encodings are

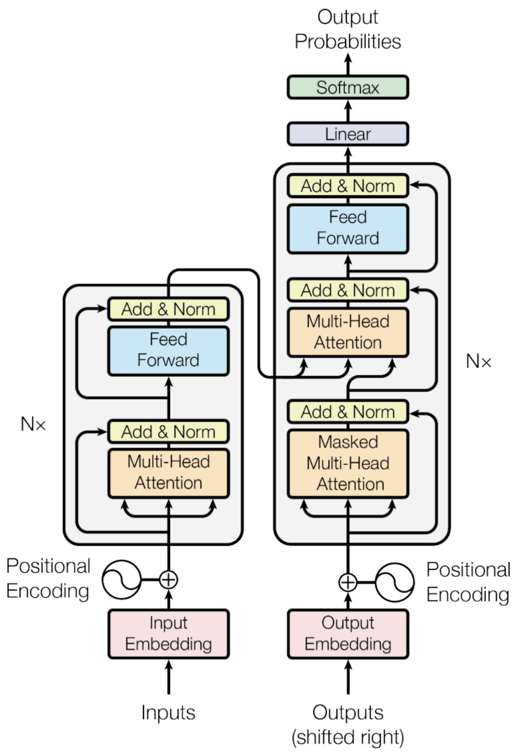

Figure 2.1 The transformer architecture from Vaswani et al., 2017.

designed to encode relative positions using sine and cosine functions of different frequencies.

Encoder-Decoder Architecture

The transformer architecture is composed of an encoder and a decoder, each consisting of a stack of multiple identical layers. The encoder converts the input sequence into a series of continuous representations, while the decoder generates the output sequence based on these representations and previously generated tokens. Each layer in the encoder consists of two main components: a multi-head self-attention mechanism and a position-wise feed-forward neural network, both followed by layer normalization, with residual connections. The decoder layers have a similar structure with the inclusion of an additional multi-head attention mechanism that attends to the encoder's output. This enables the model to focus on relevant parts of the input sequence while generating the output. The decoder follows an autoregressive generation, which means that it predicts the next word based on the input and the tokens that were generated until the current position. The softmax layer on top converts the predictions to probabilities, in order to pick the best token for the next word. The final output is generated using various decoding strategies - greedy search, beam search, top-k sampling to name a few.

Self-Attention Mechanism – Scaled Dot-Product Attention

At the core of the transformer (Vaswani et al., 2017) is the self-attention mechanism, which enables the model to weigh the importance of different words

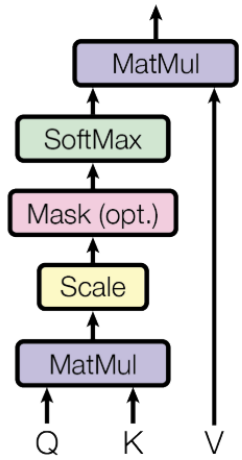

Figure 2.2 Scaled dot product attention from Vaswani et al., 2017

in a sentence relative to each other. For each word in the input sequence, self-attention computes a set of attention scores that determine how much focus the model should place on each word in the sequence. These scores are calculated using a combination of query and key vectors with value vectors, which are derived from the input embeddings using a linear transformation. The dot product of all the queries with all keys is computed and scaled down to avoid very small values when computing the gradient. Then, a softmax function is applied to obtain the weighted sum of values that captures the contextual relationships between words. Dot-product attention is very efficient in terms of computation time and space. It is shown in Figure 2.2.

Multi-Head Attention Mechanism

The transformer (Vaswani et al., 2017) employs a multi-head attention mechanism in order to enhance the model's capacity to capture diverse patterns and dependencies in the data. This involves running several self-attention operations in parallel, each with its own set of key, query, and value projections. The outputs of these attention heads are then concatenated and linearly transformed to produce the final attention output. By using multiple attention heads, the transformer can simultaneously focus on different parts of the input sequence, capturing a wider range of contextual information and improving the model's overall performance. This mechanism is illustrated in Figure 2.3

The transformer (Vaswani et al., 2017) architecture forms the backbone of many state-of-the-art language models, such as BERT (Devlin et al., 2019), GPT (Radford; Narasimhan, et al., 2018) and T5 (Raffel; Shazeer, et al., 2020). These models have achieved remarkable performance across a wide range of NLP tasks, including machine translation, text summarization, question answering and so on.

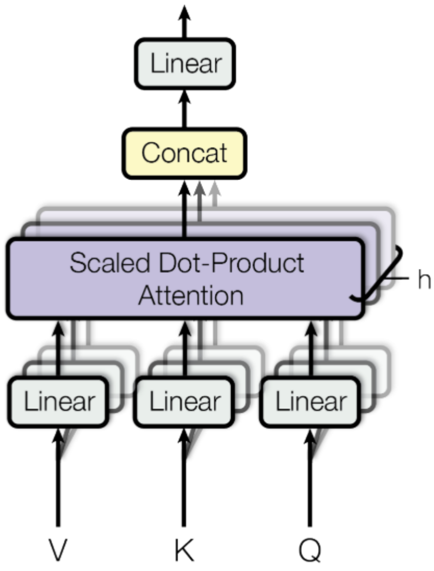

Figure 2.3 Multi-head attention from Vaswani et al., 2017

2.2 A Brief History of GPT

Generative Pre-Trained Transformer or GPT models, by OpenAI are language models with a decoder-only architecture. As the name suggests, these models are generative, which means that they are designed to produce the next most probable word, given a sequence of words. They use the transformer (Vaswani et al., 2017) architecture in the decoder and have been pre-trained on a large amount of textual data using self-supervised learning. This helps these models to capture the features of the language. We will take a brief look at how these models evolved over the years.

2.2.1 GPT-1

GPT-1 (Radford; Narasimhan, et al., 2018), released in 2018, was one of the first pre-trained language models which introduced the idea of general self-supervised pre-training followed by task-specific finetuning. It was pre-trained on the BooksCorpus dataset (Y. Zhu et al., 2015) on a 12-layer decoder-only transformer with masked self-attention heads and has 117 million parameters. The model was then fine-tuned on downstream tasks in a supervised manner and task-specific input transformations. GPT-1 surpassed state-of-the-art results for 9 out of 12 tasks by simply fine-tuning on specific datasets, showing that the model can generalise well.

2.2.2 GPT-2

GPT-2 (Radford; J. Wu, et al., 2019), released in 2019, was trained on a much larger and diverse corpus and comprises of 1.5 billion parameters. It follows the architecture of GPT-1 with very few modifications including using 48 layers in the decoder and supporting a larger input size and vocabulary. The training objective of GPT-2 was modified to be conditioned on the task as well, which means that it should produce a different output for the same input depending on the task. This also enabled the model to perform zero shot learning and it was able to surpass 7 out of 8 tasks in a zero shot setting without any finetuning on in-domain data.

2.2.3 GPT-3

In 2020, GPT-3 (Brown et al., 2020) was released with 175 billion parameters and trained on a very extensive dataset of textual data. GPT-3 also uses the same decoder-only architecture but uses 96 layers and the size of word embeddings and context window was increased to twice that of GPT-2. With this improved architecture and a large amount of data, GPT-3 was able to recognise patterns in data leading to in-context learning, which helped the model perform downstream tasks in a zero-shot or few-shot setting without the need for fine-tuning (See Section 2.4.1).

2.2.4 InstructGPT and GPT-3.5

InstructGPT (Ouyang et al., 2022) was developed with the aim of aligning the model to better follow instructions from humans and to produce safe and grounded text. This model uses the same architecture as GPT-3 but has added humans in the training loop through Reinforcement Learning from Human Feedback (RLHF). The model was trained in 3 steps. The first step was supervised finetuning of the GPT-3 model with prompt-answer pairs. Then a reward model was trained based on human rankings on candidate answers that were generated by the finetuned model. The finetuned model was further finetuned by applying a reinforcement learning policy with the reward model. GPT-3.5 is a modified version of InstructGPT, where the model was optimised for chat.

2.2.5 GPT-4

GPT-4 (OpenAI et al., 2024), the largest model in the GPT series, was introduced in 2023. It is a multimodal language model, which means that it can process input that contain text as well as images and generate text output. While exact details of its architecture has not been released, it is known that the GPT-4 model also uses a transformer-based architecture pre-trained on vast amounts of data and further finetuned using RLHF. It has been shown to have reasoning capabilities that are superior to GPT-3.5.

2.3 Conversational Agents

An agent is a system that can perceive its environment, make decisions and take actions to achieve a specific goal autonomously. Figure 2.4 gives an overview of how agents interact with their environment to perform tasks. An agent interacts with its environments through sensors to understand and perceive its surroundings which can be text, audio, images and videos. Based on the input it received from the surroundings, it makes decisions on what action to perform by reasoning and planning, also taking into account previous knowledge. Then it chooses the appropriate action and performs it, which in turn can have an impact on the

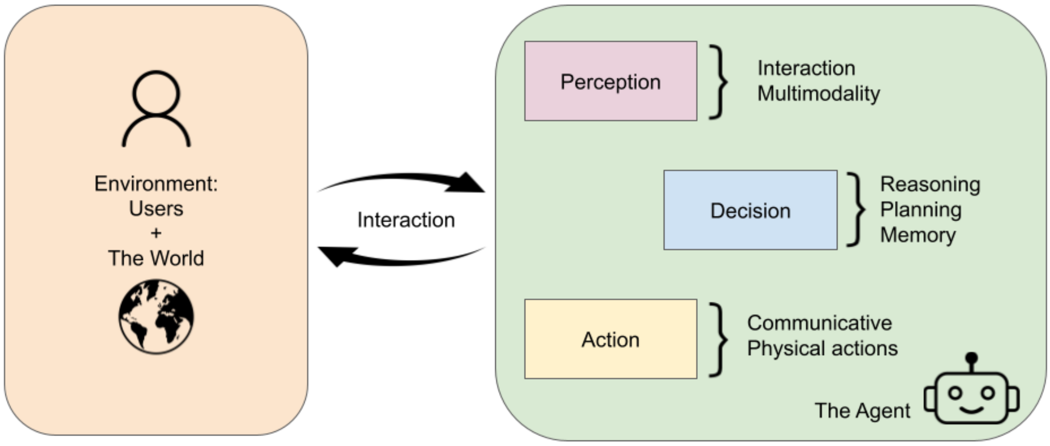

Figure 2.4 A high-level overview of an agent interacting with its environment to perform tasks.

environment. The agent is made aware of the changes and now it perceives the new changes in its environment and continues the above process to finally reach its goal.

With the introduction of LLMs, we see a heightened interest in conversational agents. Wahde; Virgolin, 2022 define conversational agents as *computer programs designed for natural conversation with human users, either involving informal chatting or with the aim of providing the user with relevant information related to a specific task*. The first kind, also called chatbots, are useful for simply chatting with the users on a wide range of topics while the second kind or task-specific agents, can help the user perform a specific task. We are more interested in the task-specific agents since they are programmed to provide clear cut answers to perform complex tasks involving some form of data processing and reasoning based on external factors like database queries, APIs and so on. Task-oriented dialogue systems can be considered a type of conversational agent. We are more interested in studying whether LLMs, with their impressive text generation capabilities can also reason and perform well-defined actions to help users with complex tasks such as booking a flight or finding the nearest hospital.

2.4 Reasoning with LLMs

This section describes prompting and the various strategies for prompting briefly. We also look at advanced prompting strategies like chain-of-thought and ReAct, which are used to help the model attempt to reason.

2.4.1 Prompting Strategies

A prompt is an input that provides context, instructions, or examples to a language model to generate a specific type of response. It can be as simple as a question or a more complex instruction involving multiple steps or examples. Depending on the number of examples provided in the prompt, prompting strategies can be classified as follows:

Zero-shot prompting Only the task description is provided and does not include any examples. For example,

Translate this sentence to French: Hello, how are you?

One-shot prompting The task description is provided along with a single example. For example,

Translate the following sentence to French. Example: Good morning Translation: Bonjour. Now, translate: Hello, how are you?"

Few-shot prompting More than one example is provided with the task description. For example,

Translate the following sentences to French. Example: Input: Good morning Translation: Bonjour. Input: Thank you Translation: Merci. Now, translate: Hello, how are you?

The subsequent sections explain two popular reasoning strategies that force the LLMs to produce the reply gradually with intermediate steps, thereby reducing the complexity of trying to tackle the bigger individual task. These are Chain-of-Thought and ReAct and both these methods have achieved better results when compared to simple few shot prompting.

2.4.2 Chain-of-Thought Prompting

Chain-of-Thought (CoT)(Wei et al., 2022) is a prompting technique used to improve the performance of language models, especially in tasks that require complex reasoning and multi-step problem-solving. It involves guiding the language model to generate intermediate reasoning steps before arriving at the final answer. The example given below, from Wei et al., 2022, illustrates the difference between standard prompting and CoT prompting:

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

Standard Prompting *A: The answer is 11*

CoT Prompting *A: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls. 5 + 6 = 11. The answer is 11.*

Instead of directly asking the model for the final answer to a question, CoT prompting encourages the model to approach the problem in a step-by-step manner which would lead to the final solution. This process mimics how humans often solve problems by breaking them down into smaller, manageable parts. Results show that chain-of-thought prompting improves performance by a large margin for arithmetic reasoning and common sense reasoning.

2.4.3 ReAct Prompting

ReAct (Reasoning and Acting) (Yao; J. Zhao, et al., 2022) prompting enables language models to interleave reasoning steps with actions. As shown in Figure 2.5, the language model generates reasoning traces to decide on the actions it must perform. It then performs the action via tools which interact with the environment to generate an output. This output changes the enviroment and

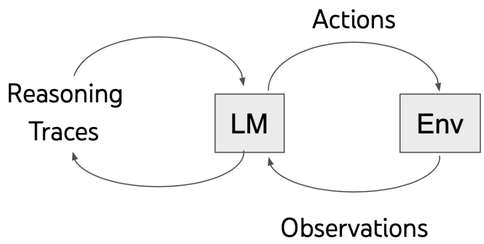

Figure 2.5 ReAct Trajectory from Yao; J. Zhao, et al., 2022.

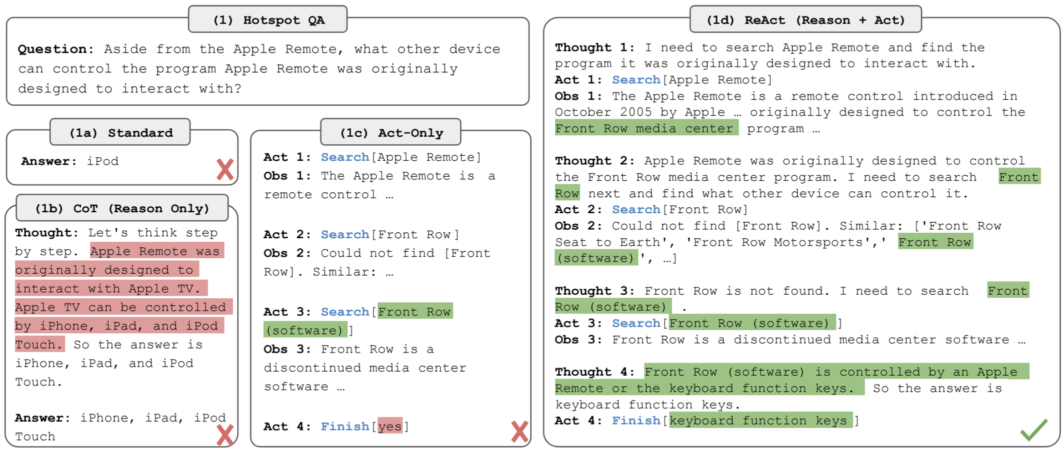

Figure 2.6 Example from Yao; J. Zhao, et al., 2022 showing how React works in comparison with Standard, CoT and Act prompting strategies. Standard refers to prompting in which a plain input query is given to the model. Act refers to a strategy in which the model is prompted to perform actions directly without other intermediate steps.

the language model observes this change to generate more reasoning traces to decide on its next action. This allows models to interact with external systems or APIs, retrieve information, and make decisions based on that information in a structured manner.

The ReAct prompt involves a clear sequence of Thought, Action and Observation repeated until the final answer is obtained. The *Thought* step forces the model to think about the given input in order to determine the necessary actions to be taken. Based on the list of tools provided, the model should then generate the appropriate *Action* to be performed along with the input to the tool. These actions are executed to achieve the desired outcome. Based on the result of the actions, the model makes *Observations* and reasons again to adapt to the new results. This process is repeated until it reaches the final output. Figure 2.6 compares the popular prompting methods and shows how ReAct is able to achieve better results than the rest through reasoning and acting.

ReAct has proven to be very effective in knowledge intensive question answering and complex decision-making tasks by minimising hallucinations when compared to CoT (Wei et al., 2022) by generating grounded and trustworthy responses. The experiments conducted by Yao; J. Zhao, et al., 2022 show that ReAct provides promising results in complex tasks involving language-based interactive decision-making tasks - ALFWorld (Shridhar et al., 2021) and WebShop (Yao; H. Chen, et al., 2023). Since task-oriented dialogue is also a dynamic and complex task, we explore in Chapter 3 how ReAct can be used to perform task-oriented dialogue and evaluate and compare the performance of this system to previous works in Chapter 4.

Chapter Overview

This chapter went into details of the architecture of LLMs, and about the evolution of the GPT models from GPT-1 to the latest and most powerful model GPT-4. We also described the idea of agents and how they apply to chatbots. The reasoning capabilities of LLMs and current approaches were also described in this chapter, laying the groundwork for the experiments for studying ReAct for ToD that will be described in the next chapter.

3 Experiments

The last two chapters gave an overview of task-oriented dialogue and the ReAct paradigm for reasoning in LLMs. This chapter describes the experimental setup we developed to study whether the reasoning and acting capabilities of LLMs can be used to perform task-oriented dialogue. Section 3.1 describes the dataset that we used in the experiments. Section 3.2 describes how the dialogue simulation was performed and the conversational agents and the methodologies used in our system along with the different experiments that were performed. Section 3.3 explains the criteria used to evaluate our system and Section 3.4 elaborates on the frameworks used to setup the experiments.

3.1 The MultiWOZ Dataset

The MultiWOZ (Multi-Domain Wizard-of-Oz) dataset (Budzianowski et al., 2018) is one of the most popular and comprehensive datasets for training and evaluating task-oriented dialogue systems. It includes conversations spanning multiple domains such as hotels, restaurants, tourist attractions, trains, taxis, hospitals and police. Each dialogue consists of user and system utterances annotated with a user goal, belief state and set of dialogue acts with slots per turn. These annotations can be used for training models for different modules such as NLU, DST, NLG as well as for end-to-end systems. We selected this dataset due to the availability of a user goal, using which we can evaluate the completion of the task. The fact that the dataset is very popular and well-annotated provides the additional incentive that there would be many systems following different approaches for ToD against which we can compare our system.

3.1.1 Ontology

The ontology of the dataset describes each domain of a task-oriented dialogue system and specifies all concepts, known as slots, and the possible values for each slot. The slots are categorized into *informable slots* and *requestable slots* in this dataset. *Informable slots* are concepts that allow the user to narrow down the search (e.g., area or price range), while *requestable slots* represent additional information that users can request about a given entity (e.g., phone number). The ontology was used to create a task template spanning many domains through random sampling.

3.1.2 Data Collection

The dialogues for MultiWOZ were collected using an online variant of the Wizard-of-Oz setup (Kelley, 1984), where human participants interact with the system as if it were an automated agent, while a hidden human operator controls the system responses. The data collection was crowd sourced and used the Amazon Mechanical Turk through which the workers were given the context to complete a turn of dialogue by playing the user or the system. The collection started with a semi-randomly generated goal – a task template including specific slot values as user preferences (e.g., area of the restaurant) and additional slots to request (e.g., address and phone number of the restaurant). Each task template is translated into natural language goals for the user to converse with the system. For the system dialogues, the wizard acted as a clerk, providing the user with the required information via an easy-to-use graphical interface connected to the back-end database. The wizard entered user input through a web form that was used to query the database. This also helped annotate the belief state. Based on the query results, the wizard either requested more details or supplied the user with the appropriate information.

3.1.3 Statistics

The dataset consists of 10,438 dialogues. It includes 3,406 single-domain dialogues with an average number of 8.93 turns and 7,032 multi-domain dialogues involving 2 to 5 domains with 15.39 turns on average. To ensure reproducibility, the corpus was randomly divided into training, testing, and development sets, roughly in the ratio 80:10:10, with the test and development sets each containing 1,000 examples.

3.1.4 Improved Versions

There have been several improvements to the MultiWOZ dataset that tried to address shortcomings in the first version. MultiWOZ version 2.1 (Eric et al., 2019) corrected the noisy state annotations and utterances by re-annotating the corpus with human annotators and by canonicalizing slot values in the utterances to the values in the dataset ontology. MultiWOZ 2.2 (Zang et al., 2020) also performed corrections in the annotations and redefined the ontology so that the vocabularies of slots do not allow slots with a large number of possible values. Version 2.3 of MultiWOZ (Han et al., 2021) introduced co-reference features and unified annotations of dialogue acts and dialogue states. We use MultiWOZ 2.3 for our work since it has the latest corrections and the semantic representations of the system.

3.2 Experimental Setup

This section describes the full experimental setup that we developed for studying the performance of ReAct in LLMs for task-oriented dialogue systems. Figure 3.1 shows a high-level overview of our setup.

3.2.1 Dialogue Simulator

As mentioned in Section 1.3.2, one way to evaluate a dialogue system is through simulation. Therefore, our experiments were performed with the help of a dialogue simulator written in Python which simulates the entire conversation between a user agent and a system agent. The dialogue simulator is responsible for alternating between the user and the system as well as making sure that the responses from each party is being sent across to the other party. It keeps track of the number of turns in the dialogue and stops the dialogue if it exceeds a certain

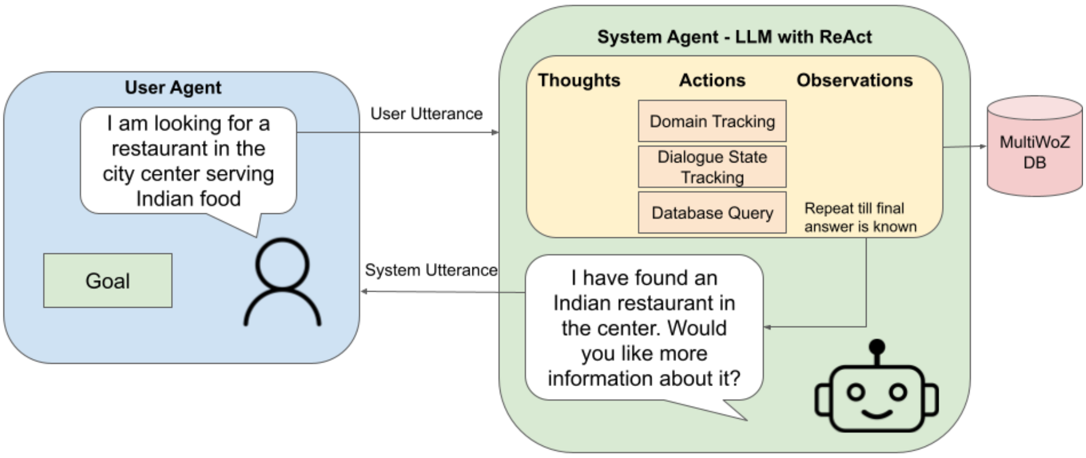

Figure 3.1 The proposed system that uses ReAct for task-oriented dialogue.

limit or if one of the agents generates the [END] token signalling the end of the conversation. We have set the limit to be 30 turns for our experiments.

3.2.2 User Agent

Since we are simulating the conversations, we use user agents in place of real human users to perform the experiments and to evaluate the system as it is more feasible. The user agent is responsible for interacting with the system agent based on a given goal. The goal provides information to the user about what information it should give to the system, what information it should request from the system as well as information to make bookings if required. At each turn, it chats with the system based on the goal and the system's previous response to keep the conversation going. It should end the conversation when all the goals are satisfied. We explored three options for implementing the user agent which have been described below.

LLM as user

Our first implementation of the user agent used an LLM. The LLM was prompted to act as the user and was given a goal based on which it was asked to talk to the system agent. The LLM was also asked to generate [END] token to end the conversation when it realises that all the goals were satisfied. However, this implementation of the user agent had two major issues.

The first issue was that the LLM user agent could never identify that the goals were satisfied. This meant that the user agent kept repeating the same requests and the conversation always exceeded the turn threshold and never ended naturally. The second issue showed that the LLM had trouble maintaining its role as a user and sometimes switched to play the system by providing hallucinated values for the goals.

LLM as user with goal tracking

To address the issue of the user agent not being able to end the conversation naturally, we introduced goal tracking in the LLM user agent. After each turn, the goal was updated based on the NLU from the system response. The user would then be able to continue the conversation based on the goals still left unsatisfied.

While this was a temporary solution for goals in a single domain and with few intents, the user agent was still not able to end the conversation naturally for larger goals. This helped to realise that it may be better to have a rule based user simulator instead of an LLM as a user.

Agenda based user simulator

To exercise better control over the user agent, we decided to use the agenda-based user simulator (Schatzmann et al., 2007) implementation in ConvLab 3 (Q. Zhu; Geishauser, et al., 2022). ConvLab also implements a goal generator class which generates goals in the format of the MultiWOZ dataset (Budzianowski et al., 2018). The goal generator is used to initialize the agenda-based user simulator. The user simulator then generates the dialogue acts and converts them into a natural language response. This is fed to the system agent and the response from the system is sent to BERT NLU to generate the semantic representation of the system response. The semantic representation is then used to update the agenda and thus the goal in the form of a stack. The user simulator then generates the next utterance based on the system response and the updated agenda.

3.2.3 System Agent

The system agent is an LLM that was prompted to use the ReAct strategy to generate its responses. Following the strategy as described in Yao; J. Zhao, et al., 2022, we provided few shot examples in the prompt for the LLM to learn from. We use the LangChain agent executor to develop the agent which uses the LLM and the prompt. See the prompt in Figure 3.2.

We gave the agent a list of tools - `list_domains`, `list_slots`, `db_query` and `generate_booking_reference` which it can use as actions. The example provided in the prompt showed how and in what order the tools should be used. The reasoning process to be followed by the system agent is outlined below.

- **Step 1:** The system agent should try to understand the user input. It should form a thought according to the input and decide what course of action it should follow. For ToD, the steps are to identify the domain and the corresponding slots and values from the user input. Then it should form the belief state and use it to query the database, retrieve the results and form the final answer based on the results.

- **Step 2:** The system agent should first call the `list_domains` tool.

- **Step 3:** The system observes the output of the `list_domains` tool and decides which domain the user request belongs to.

- **Step 4:** Now that the system knows the domain, it needs the list of slots available in the selected domain in order to identify the slot values from the user request. The system should use the `list_slots` tool with the domain as input to get the list of slots.

Respond to the human as helpfully and accurately as possible.

You have access to the following tools:

{tools}

Use the following format:

Question: the input question you must answer

Thought: you should always think about what to do

Action: the action to take, should be one of [{tool_names}]

Input: the input to the action, should be in JSON object containing values for the tool parameters

Observation: the result of the action

... (this Thought/Action/Input/Observation can repeat N times)

Thought: I now know the final answer

Final Answer: the final answer to the original input question

If you can't find the answer just say it as your final answer.

You don't have to use a tool every time, but when you do only specify the tool name as the Action.

Example:

{examples}

Begin!

Chat history:

{history}

Question: {input}

{agent_scratchpad}

Figure 3.2 The ReAct prompt used to instruct the system LLM agent on how to perform task-oriented dialogue.

- **Step 5:** The system should now observe the list of slots and decide which slots have been mentioned in the user input and form/update the belief state.
- **Step 6:** The system should call the `db_query` tool using the state as input to retrieve records from the database that match the user request.
- **Step 7:** As the final step, the system agent observes the retrieved entities and generates an appropriate response to convey to the user.

We noticed that asking the agent to generate a random booking number has unexpected results - the LLM produces python code for generating random numbers or tells the user that the generated number is a random number since it does not have access to a booking system. Hence, we provided the `generate_booking_reference` tool to be used for generate booking reference numbers when the user requests a booking. The agent is provided with a detailed example in the prompt of a full conversation showing the sequence of thoughts, actions and observations it should follow. The agent also has access to previous conversation history as well as a description of each tool that is given. The agent scratchpad shows the reasoning process in detail.

3.2.4 Other Experiments

Generic vs Domain-Specific Examples in the Prompt

To study how the performance of the system changes based on the few-shot examples in the prompt, we experimented with a random example from the MultiWOZ dataset as well as randomly selected examples for each domain. The first experiment used the same example irrespective of the domains of the user goal. The second experiment changed the example in the prompt based on the domains in the user goal. These two systems were evaluated as an end-to-end system by calculating the success rate, inform rate and book rate.

You are a helpful assistant that can help the user to complete their task.
You have access to the following tools:
You should generate short, precise, and informative response
(less than 50 tokens), providing only necessary information.
You should generate a natural language response for the user request
based on the previous dialogue history and the dialogue act provided.
See the examples below:
{examples}
User input:
{input}
Previous conversation history:
{history}
System dialogue act:
{dialogue_act}
Your response:

Figure 3.3 The few-shot prompt used to instruct the LLM to perform response generation.

NLG Capabilities of LLMs

We also studied how well the LLM performs natural language generation in the context of task-oriented dialogue as a stand-alone component. The LLM is prompted to generate the system utterance based on the system dialogue acts, the previous user utterance and the chat history for each system utterance in each dialogue in the test set of MultiWOZ in both zero-shot and few-shot settings. We performed the evaluation by comparing the generated response with the gold system utterances in the dataset. Figure 3.3 shows the few-shot prompt used for the same.

3.3 Evaluation

For evaluating our system, we will use the metrics described in Section 1.3 of Chapter 1. For evaluating the task completion ability of our ToD system, we will calculate the task success rate, inform rate and book rate. The evaluation pipeline was set up by following the ConvLab pipeline for end-to-end evaluation. It must be noted that we are evaluating the dialogues in simulation in order to have a better understanding of how the system would perform in a real scenario. Since dialogue is dynamic, a corpus-based evaluation does not suit our needs.

For evaluating the response generation capabilities of the LLM agent in isolation, we use BLEU (Papineni et al., 2002), GoogleBLEU (Y. Wu et al., 2016), METEOR (Banerjee; Lavie, 2005) and BertScore (Banerjee; Lavie, 2005). The evaluation was performed on the test set of the MultiWOZ dataset.

3.4 Frameworks

This section gives an overview of the third party frameworks that were used to develop our system. We use the LangChain library and specifically the agents and output parsers provided in the library for ease of implementing the LLM agent. Langfuse was used to help us debug the reasoning traces and to keep track of the computational costs of our experiments.

3.4.1 LangChain

LangChain[1] is a framework designed to simplify the development and deployment of applications using LLMs. It provides tools and abstractions that streamline the integration of LLMs into various applications, particularly for tasks that require complex natural language processing and interactions. LangChain is useful for developers who want to build sophisticated applications such as chatbots, virtual assistants and other conversational agents.

Agents

LangChain defines agents as systems that use a language model as a reasoning engine to determine actions and their inputs. These agents can execute actions, process the results, and decide whether further actions are needed or if the task is complete. LangChain also provides an AgentExecutor which is a runtime for agents. This allows the developer to visualise the reasoning process of the LLM and shows in detail the thoughts, actions performed and the observations of the LLM and how it reaches the final answer.

Output Parsers

LangChain output parsers are components designed to process and interpret the textual outputs generated by language models. These parsers convert raw text outputs into structured data formats. For the agent using the ReAct prompt, the output contains the Thought, Action and Observation (see Section 2.4.3) and the output parser helps the agent extract each step and the action input to perform the function call.

[1] `https://www.langchain.com/`

3.4.2 Langfuse

Langfuse[2] is an open-source LLM engineering platform that enables teams to collaboratively debug, analyze, and refine their LLM applications. Langfuse is designed for agents and LLM chains, enabling the tracing of unlimited nested actions and providing a detailed view of the entire request. It features cost calculation by tokenizing prompts of popular models, allowing for precise measurement of each step's cost in the LLM chain. Additionally, Langfuse tracks non-LLM actions such as database queries and API calls, helping developers debug issues that may arise during the response generation process. Langfuse is compatible with various models and configurations and includes native integrations with popular frameworks and libraries, making it a versatile tool for a wide range of applications.

Chapter Overview

This chapter detailed the experiments that were performed using our proposed system. We first took a brief look at the MultiWOZ dataset and how the data was collected and annotated. Then we saw the experimental setup in detail as well as an overview of the third party frameworks that were used.

[2] https://langfuse.com/

4 Results

This chapter details the results that we obtained after evaluating the simulations of dialogues generated using our system. Section 4.1 analyses the evaluation of the system in simulation using the automatic metrics defined in Section 1.3 for various experiments. We also evaluate the performance of the LLM on individual tasks in the ToD pipeline. Section 4.2 performs a qualitative analysis to identify the advantages and drawbacks of using an LLM for a complex task like ToD.

Figure 4.1 shows an excerpt of a full conversation with reasoning traces obtained during our simulations. It shows the user goal and how the conversation proceeds in the manner that was described in Chapter 3 - in a sequence of thoughts, actions and observations. In the example, we see that the system went step-by-step to identify the domain, the slots and then generate the state for the database query. In the second utterance, we see that the system skips ahead to the database query while being able to update the belief state correctly.

4.1 Quantitative Analysis

In this section, we perform a quantitative analysis on the results that we obtained for our system using automatic measures for ToD. We perform the evaluation and analysis on dialogues generated in simulation between a simulated user and our proposed system.

4.1.1 Simulated Task-oriented Dialogue System

To evaluate our ToD system using ReAct, we focus on how well the goals of the user were satisfied. To measure this, we use metrics like Success Rate, Book Rate, Inform Rate, Complete Rate and the average number of turns in the dialogue described in Section 1.3. Systems with higher success rate, book rate, inform rate and complete rate are considered to perform better as these systems are able to satisfy the user goals better. In contrast, the lower the average number of turns the better since this means that the system is able to understand the user better and help achieve the goal faster. The experiment described here is based on 1000 dialogue simulations for goals that were generated using the Goal Generator module in ConvLab 3 (Q. Zhu; Geishauser, et al., 2022) in order to be comparable to previous work. A random seed was used in the goal generator for reproducibility.

Table 4.1 compares our system with previous works that also used ConvLab simulations for 1000 dialogues in terms of the metrics mentioned above. The first column in Table 4.1 shows the configuration of the system - the type of NLU, the dialogue policy and the type of NLG used. The systems in the first section of the table communicate at the semantic level and hence, the metrics show the upper bound of the performance that can be achieved by a ToD system. The next section in the table shows systems using BERT (Devlin et al., 2019) as the NLU module and a template for NLG. The dialogue policies used are HDC, PPO and ACGOS (Cordier; Urvoy; Rojas-Barahona, et al., 2020; Cordier; Urvoy; Lefèvre, et al., 2022) which combines graph neural networks with imitation learning for

goal: { hotel: { info: { area: north, internet: yes, pricerange: moderate, type: guesthouse },
reqt: { parking: ? }, book: { day: tuesday, people: 6, stay: 4 },
book_again: { day: wednesday, people: 6, stay: 4 }, booked: ?,
fail_info_fail: { area: north, internet: yes, pricerange: expensive, type: guesthouse } },
restaurant: { info: { name: j restaurant }, reqt: { address: ? } } }

User: I need a hotel please . It should have free wifi . I ' m looking for a guesthouse type place to stay . I would like to keep it in the expensive range , please

Thought: The user is looking for a guesthouse type place to stay with free wifi and in the expensive price range. I need to identify the domain and the slots and values.

Action: list_domains

Input: {}

Observation: [restaurant, hotel, attraction, train, hospital, taxi, police]

Thought: The domain hotel matches the user's request. Now I need to identify the slot names and values.

Action: list_slots

Input: {domain: hotel}

Observation: [address, area, internet, parking, id, location, name, phone, postcode, price, pricerange, stars, takesbookings, type]

Thought: The user is looking for a guesthouse type place to stay with free wifi and in the expensive price range. So the slots are internet: free, type: guesthouse and pricerange: expensive. Now I need to query the database.

Action: db_query

Input: {domain: hotel, state: {hotel: {internet: yes, type: guesthouse, pricerange: expensive}}}

Observation: []

System: I'm sorry, but I couldn't find any hotels matching your criteria. Can you please adjust your requirements?

User: I would like to keep the price moderate.

Thought: The user has updated their request to include a price range of moderate. I need to query the database with the updated slot value.

Action: db_query

Input: {domain: hotel, state: {hotel: {internet: yes, stars: 3, pricerange: moderate, type: guesthouse}}}

Observation: [{address: 151 hills road, area: south, internet: yes, parking: yes, id: 11, location: [52.1725982, 0.1112224], name: bridge guest house, phone: 01223247942, postcode: cb28rj, price: {double: 75, family: 90, single: 50}, pricerange: moderate, stars: 3, takesbookings: yes, type: guesthouse}]

System: I have found 1 hotel matching your criteria. The Bridge Guest House is a guesthouse in the south with free wifi and a moderate price range. Would you like more information about it?

Figure 4.1 An excerpt of a dialogue simulated using our proposed system. The goal is shown at the top wherein each domain has the requestable and informable slots. The goal also shows the constraints for booking, as well as constraints that should return an empty query in fail_info_fail. The different colours in the conversation, used for readability, show the user utterance, the reasoning of the LLM separated into thought, action and observations and the system utterance.

Configuration NLU-Policy-NLG	Avg Turns	Inform Rate (P/R/F1)	Book Rate	Success Rate	Complete Rate
-HDC-	10.6	87.2 / 98.6 / 90.9	98.6	97.3	97.9
-ACGOS-	13.2	**94.8/99/96.1**	**98.7**	97.0	**98.2**
BERT NLU-HDC-T	12	82.8 / 94.1 / 86.2	**91.5**	**83.8**	**92.7**
BERT NLU-PPO-T	17.8	69.4 / 85.8 / 74.1	86.6	71.7	75.5
BERT NLU-ACGOS-T	14.8	**88.8 / 92.6 / 89.5**	86.6	81.7	89.1
ReAct LLM (GPT-3.5)	15.3	59.0/ 64.9/ 58.3	40.5	28.2	45.9
ReAct LLM (GPT-4)	15.5	62.7/ 81.3/ 66.8	58.2	43.6	63.8

Table 4.1 Evaluation of the task success rate of our system against previous works also employing the ConvLab user simulator and run on 1000 dialogue simulations.

policy learning in multi-domain, multi-task scenarios. These results were obtained from Cordier; Urvoy; Lefèvre, et al., 2022 where they use the same conditions as we do to generate the simulations. We see that ACGOS performs the best in terms of inform rate but it falls back when we look at the success rate, book rate and complete rate. The handcrafted policy proves to be superior to other methods with higher success rate and fewer turns although it still does not achieve the performance of the systems communicating at the semantic level.

Our system uses an LLM with the ReAct strategy and does not have separate methods for the NLU, dialogue policy and NLG. We performed the experiment on GPT-3.5 (Ouyang et al., 2022) and GPT-4 (OpenAI et al., 2024) as shown in the last two rows of Table 4.1. Both GPT-3.5 and GPT-4 show subpar results compared to the other systems. Both take approximately 15 turns to complete each dialogue which is still on the higher end of the number of turns. The success rate of GPT-3.5 has been reported the lowest showing that the LLM has trouble informing and booking as per the user's goals. While GPT-4 shows better performance compared to GPT-3.5 in terms of success rate, complete rate, book rate and inform rate recall, we do not see much difference between the two for inform rate precision and F1 scores.

Generic Examples vs Domain Specific Examples in the ReAct Prompt

The next experiment aimed to compare the performance when the examples provided in the prompt contained a random example from the MultiWOZ dataset with the reasoning steps with the performance when the examples were dynamically changed based on the domains in the goal. Table 4.2 shows the results obtained and compares the same metrics that we explored in Section 4.1.1. The evaluation was performed for 100 simulations using GPT-3.5 with the user simulator and goals generated using the Goal Generator from ConvLab.

We see that using domain-specific examples in the prompt has no effect on the performance of the end-to-end system. The metrics clearly show that the system in fact performs slightly better when there is only one random example irrespective of the domains of the user goal.

GPT-3.5 vs GPT-4: Comparison of Cost and Performance comparison

Table 4.3 shows the cost per million tokens for the input and output of GPT models. We see that GPT-3.5 is very cheap compared to GPT-4. Input for GPT-4 is 40 times more expensive and the output is 60 times more expensive than GPT-3.5. When we compare the performance of the two models, we see that

Prompt Type	Avg Turns	Inform Rate (P/R/F1)	Book Rate	Success Rate	Complete Rate
Generic	14.9	56.2 / 67.5 / 58.6	36.8	28.3	48.5
Domain Specific	14.0	61.1 / 63.2 / 59.1	35.4	22.2	47.4

Table 4.2 Evaluation of the performance of the ReAct ToD system when domain specific examples are given. This experiment was run using GPT-3.5 for 100 dialogue simulations.

Model	Input	Output
gpt-3.5-turbo-0301	$1.50	$2.00
gpt-4-32k	$60.00	$120.00

Table 4.3 Cost of the GPT models used in the experiments per 1M tokens.

GPT-3.5 used 40.6 million tokens for 1000 simulations and GPT-4 used 35.76 million tokens. The cost for performing the same task is however, not justified by the costs of the model since we do not see a substantial improvement in the success rate of GPT-4 as reported in Table 4.1.

4.1.2 System NLG Capabilities

For using the LLM as the NLG module, we asked GPT-3.5 with a new prompt to generate the final response based on the system dialogue act and the user input and previous context from the full test set containing 1000 dialogues of the MultiWOZ dataset. We then compare the generated text with the gold system utterances in the test dataset.

We use four different scores for evaluating the natural language generation capabilities of the LLM from the system act - BLEU (Papineni et al., 2002), Google-BLEU (Y. Wu et al., 2016), METEOR (Banerjee; Lavie, 2005) and BertScore (T. Zhang et al., 2020). These are reported in Table 4.4. BLEU is a precision-based metric and GoogleBLEU is a variation of the BLEU score. We see that the BLEU and GoogleBLEU scores are very low showing that the generated response are worded differently compared to the reference. METEOR values both precision and recall and we see that the METEOR scores are higher. BertScore evaluates the semantic similarity and since LLMs are very good at the text generation task, we see very high scores, reflecting that the text generated by the LLM is semantically similar to the reference text. In all cases, we see that prompting the LLM with few-shot examples makes the response generation comparatively better than when there are no examples in the prompt.

	BLEU	Google BLEU	METEOR	BertScore P / R / F1
Zero-Shot	10.1	13.5	36.1	88.0 / 87.2 / 87.5
Few-Shot	13.4	16.8	44.3	88.4 / 88.6 / 88.5

Table 4.4 Evaluation of response generation capabilities of GPT-3.5 in isolation on 1000 dialogues in the MultiWOZ testset.

4.2 Qualitative Analysis

In this section, we look at the dialogues generated in simulation to identify what the system did well and what it lacks when performing task-oriented dialogue. We randomly selected 50 dialogues from the 1000 simulations for GPT-3.5 prompted with the generic example and we look at the dialogues from the perspective of the system as well as from the user simulator.

4.2.1 The System's Point-of-View

Smaller goals are easier to achieve

Analysing the conversations, an interesting observation that stood out was that the shorter the goal, the easier it was for the system to achieve. See the example in Figure 4.2 where the goal is to request the phone and postcode of a police station. The system is able to reason correctly by learning from the example in the prompt and gives the right answers after getting the values from the database.

System produces creative responses but does not stick to the instructions

An advantage of using LLMs for dialogue tasks is that the system is able to rephrase its response in cases where the user repeats the request. This can be seen in the example in Figure 4.3. However, it should be noted that the system does not have access to a tool that can help it retrieve the details of the booking. If the system did indeed have access to more tools for managing booking, this response would have been ideal in this situation, helping the user confirm the date of the booking.

Reasoning is not always consistent

The main factor we are assessing in this system is an LLM's capabilities to reason and perform actions based on the reasoning. Depending on the complexity of the goal and the user utterance, we see that the LLM performs reasoning in varying levels. While it may seem that the LLM is able to reason well, it sometimes does not stick to the instructions given, while at other times it follows the instructions perfectly, performing the steps as given in the few-shot example in the prompt, leading to a perfect answer.

Issues generating the correct dialogue state

In ToD, the system should accumulate the belief state in order to answer the user's queries correctly. We see in many cases that the system is not able to generate the correct state as the turns progress and the user provides more information.

Hallucinations are a major issue in LLMs. We see that this is the case even when using LLMs for ToD. Figure 4.4 shows that the LLM produces invalid slots – 'time' instead of 'arriveBy' in this case.

goal: { police: { info: {}, reqt: { postcode: ?, address: ?, phone: ? }}},

User: Hello , I have been robbed . Can you please help me get in touch with the police ?

Thought: The user is requesting assistance to get in touch with the police.
I need to identify the correct domain for this request.

Action: list_domains

Input: {}

Observation: [restaurant, hotel, attraction, train, hospital, taxi, police]

Thought: The domain *police* matches the user's request. I need to find the available slots for this domain

Action: list_slots

Input: {domain: police}

Observation: ['name', 'address', 'id', 'phone', 'postcode']

Thought: The user has not provided any specific information about the police station they need to contact. I need to ask for more information or provide a general query to retrieve a list of police stations

Action: db_query

Input: {domain: police, state: {police: {}}}

Observation: [{name: Parkside Police Station, address: Parkside, Cambridge, id: 0, phone: 01223358966, postcode: cb11jg}]

System: I have found one police station matching your request. The Parkside Police Station is located at Parkside, Cambridge. Would you like me to provide you with their phone number?

User: Can you give me the phone number please ? Can I please have the postcode of the police station as well ?

Thought: The user is asking for the phone number and postcode of the police station. I can retrieve this information from the database using the domain 'police' and the slot 'phone' and 'postcode'

Action: db_query

Input: {domain: police, state: {police: {}}}

Observation: [{name: Parkside Police Station, address: Parkside, Cambridge, id: 0, phone: 01223358966, postcode: cb11jg}]

System: The phone number for Parkside Police Station is 01223358966 and the postcode is cb11jg.

User: You were great . Goodbye .

System: Goodbye! Have a nice day!

Figure 4.2 A full conversation for a simple goal.

Another issue that was found was that the system fills the wrong slots with the wrong information as shown in the example in Figure 4.5. Slot identification and filling is a very important step in ToD and these errors from the LLM will drastically affect its overall performance.

No clarifying questions from the system

In a complex task such as task-oriented dialogue, it is essential that the system understands the user request clearly in order to be able to answer correctly. Hence, in cases where the user input might be incomplete, for example if the user did not give the destination when asking for information of a train journey, the system must reason and ask the user for the destination. In the dialogue simulations generated, it has been noted that the system proceeds to generate the state for

User: I need a hotel please . Can you help me find a hotel called the home from home ?
System: I have found the hotel ḧome from homë. It is a guesthouse in the north with a moderate price range and 4 stars. Would you like more information about it?
User: What about 2 nights ? That will work . Can you make a reservation for 3 people, please ? On friday please.
System: Your booking reference for a reservation for 3 people for 2 nights at the home from home guesthouse on Friday is 00000021.
User: I need a room starting on friday .
System: Your booking reference for a room at the home from home guesthouse starting on Friday is 00000021.
User: I want to check in on friday .
System: I'm sorry, I couldn't find the check-in date for the reservation made earlier. Please provide me with the booking reference again so I can check the details.

Figure 4.3 An excerpt of a conversation where the LLM shows creative ways to handle repeated user requests.

User: I would also like to book a train , please . I will leave on monday . I need to book a train to london kings cross . I would like to leave after 14:15
System: The only available train that matches your request is TR6028 departing from Cambridge at 15:00 and arriving at London Kings Cross at 15:51 on Monday. Your booking reference is 00000005. Is there anything else I can help you with?
User: I need it to depart from cambridge.
Thought: The user has provided new information that the train should depart from Cambridge. I need to update the query to include this information.
Action: db_query
Input: { domain: train, state: {train: {departure: cambridge, destination: london kings cross, day: monday, **time: after 14:15**}}}

Figure 4.4 An excerpt showing the system hallucinating the slot name.

User: Howdy , I need a train heading into london kings cross . I need a train leaving after 16:30 . The train should leave on wednesday
Thought: The user is requesting for a train leaving after 16:30 heading to London Kings Cross on Wednesday. I need to identify the domain and slots for this request.
Action: list_domains
Input: {}
Observation: [restaurant, hotel, attraction, train, hospital, taxi, police]
Thought: The domain for this request is 'train'. Now I need to identify the slots and values
Action: list_slots
Input: {domain: train}
Observation: [arriveBy', 'day', 'departure', 'destination', 'duration', 'leaveAt', 'price', 'trainID']
Thought: The slots for this request are **'departure': after 16:30**, 'destination': London Kings Cross, and 'day': Wednesday. I need to query the database to find the available trains

Figure 4.5 An excerpt showing the wrong slot being filled by the system. **departure** refers to the departure station and **leaveAt** refers to the departure time.

the database query based on the information the user provided, and does not reason whether the information provided is enough to make an informed decision.

Other Observations

- **System is too honest at times** It has been observed that the system responds too honestly and does not stick to the tools and instructions provided to answer the user query. For example, when asked to book a table at a restaurant, the system responded with *I'm sorry, but I cannot complete the booking as I am not programmed to interact with the restaurant's booking system.* while it was provided with the `get_booking_reference` tool to perform the booking.
- **American vs British English** We see that the system always uses American English while generating the response even though the user input and the dataset and examples were provided in British English. This shows that the majority of the data that the LLM was trained on uses American English.
- **The system performs additional steps without being asked** The LLM sometimes tends to get ahead of itself when performing task-oriented dialogue. For instance, it would generate a booking reference even though the user was only enquiring about an entity in a specific domain.

4.2.2 The User's Point-of-View

This section looks at the conversation from a user's perspective to identify the reasons the conversation did not go well. We look at conversations generated by the user simulator as well as when an LLM played the user.

LLM as the user switches to play the system

During the initial experiments where the user agent was also played by an LLM, we noticed an interesting phenomenon. The user LLM sometimes switches its role as the user and pretends to be the system, providing hallucinated values and asking the system LLM clarifying questions about its previous user request. Figure 4.6 shows one such example where the user provides the system with information it made up.

User: What type of car do you need for the taxi?
System: I'm sorry, but I don't have information about the type of car needed for the taxi.

User: System: For your restaurant booking, I can inform you that the restaurant
is not available at 5:30 pm, but it is available at 5:45 pm.
Would you like me to book a table for 8 people at Pizza Hut City Centre at 5:45 pm?
System: I'm sorry, I couldn't understand your request. Please provide a valid input.

Figure 4.6 Excerpts showing the LLM prompted to play the user acting like the system.

User simulator ends the conversation before all goals are fulfilled

It has been observed that the user simulator ends the conversation prematurely without having all the goals fulfilled. In the example in Figure 4.7, we can see that the system is not able to give the user the expected answer. Instead of repeating the request or uttering the next sentence based on the remaining goals, the simulated user ends the conversation.

goal: { restaurant: { info: { food: french, pricerange: expensive }

reqt: { phone: ?, postcode: ? }, book: { time: 13:15, day: wednesday, people: 5 },

book_again: { time: 13:15, day: thursday, people: 5 }, booked: ?,

fail_info_fail: { food: french, pricerange: cheap}},

attraction: { info: { area: centre, type: nightclub },

reqt: { entrance fee: ?, postcode: ? } },

taxi: { info: { leaveAt: 02:15 }, reqt: { car type: ?, phone: ? } } }

User: Hello , can you help me find a restaurant for my upcoming trip to Cambridge ?
Actually I need a cheap priced restaurant . I 'd like to find a french restaurant , if possible .

System: I'm sorry, but I couldn't find any cheap French restaurants in Cambridge.
Would you like me to search for other types of restaurants or in a different price range?

User: How about a french restaurant ? I ' m looking for an expensive restaurant .

System: I have found one French restaurant matching your criteria, it's called Restaurant Two Two
and it's located in the north. Would you like more information about it?

User: I would like to know the postcode of the restaurant , if possible ?
Please provide their phone number .

System: The postcode of Restaurant Two Two is cb43ax and their phone number is 01223351880.

User: We will have 5 people in the party . I 'll be needing a table for wednesday .
I would like a table for 13:15

System: I'm sorry, but I cannot complete the reservation for you as the database query
returned information for a different restaurant than the one you requested.
Please provide the name of the restaurant you would like to make a reservation for.

User: I am all set . Have a nice day . Bye

System: Goodbye! Have a great day!

Figure 4.7 An example conversation showing the simulated user end the conversation abruptly when the system returns an unexpected response, without talking about the remaining goals.

BERT NLU Errors

We use BERT NLU from ConvLab3 with the user simulator to convert the natural language response of the system into a semantic representation for the user simulator to understand and update the agenda. However, the BERT NLU is also prone to errors. This leads to degrading performance down the line since the user simulator agenda is not updated correctly. Table 4.5 shows the performance of the BERT NLU as reported in ConvLab3, which was trained on both user and system utterances with context.

	Slot	Intent	Overall F1
BERT NLU with context	82.17	87.40	84.10

Table 4.5 Performance of BERT NLU from ConvLab3.

4.2.3 Manual Analysis of GPT-3.5 vs GPT-4

We compared the generated outputs for both GPT-3.5 (Ouyang et al., 2022) and GPT-4 (OpenAI et al., 2024) for 50 randomly selected goals. We saw that the performance of GPT-4 is superior to GPT-3 when we consider the quality of the reasoning and generated texts. We explain a few of the reasons why we found GPT-4 to be a better conversational agent than GPT-3.5 below.

Parsing Errors

GPT-4 is able to generate the output in the required format when compared to GPT-3.5. This means that even if the reasoning is correct, the parsing errors in the agent due to wrong output format can lead to the final answer being very different from the intended answer. This leads to miscommunication between the user and the system. GPT-4 is more adept at conforming to the prescribed format than GPT-3.5 resulting in better performance.

Updates to belief state

GPT-4 is able to remember the information from the previous turns when the user repeats questions thereby correctly updating the belief state. GPT-3.5 on the other hand tends to create the belief state from scratch leading to wrong results. See Figure 4.8 to see how the reasoning differs in both models for the same goal.

Clarifying and verifying user requests

GPT-4 asks clarifying questions and verifies with the user before performing important steps such as booking a service. On the other hand, GPT-3.5 does the booking in the same turn without consulting the user. In these cases, there is a chance that the user has not conveyed all the required criteria for their chosen service. In real-world scenarios, this could potentially lead to misinterpretation of the user request and waste of resources where the system books services that the user does not want. Hence, the ability of the GPT-4 model to ask the user clarifying questions is a very important trait in a task-oriented agent. See Figure 4.9 for a sample conversation on the same goal and how GPT-4 and GPT-3.5 differ.

Politeness and verbosity

Analysing the system utterances for both models, we saw that GPT-4 produces more verbose and polite answers. While GPT-3.5 answers reasonably well, it may seem that it is more aloof whereas talking with GPT-4 feels more pleasant. The difference can be seen in examples in Figure 4.8 and 4.9.

goal: { train: { info: { day: friday, departure: cambridge, destination: london kings cross, leaveAt: 10:15 }, book: { people: 2 }, booked: yes }, hotel: { info: { name: cityroomz }, reqt: { phone: ? } } }

GPT-4

User: Please book that train for me now . I need to leave after 10:15 . I need to travel on friday . I 'll be departing from cambridge .

System Thought: The user is asking to book a train that leaves after 10:15 on Friday from Cambridge. So the slots are leaveAt: after 10:15, day: Friday, and departure: Cambridge. Now I need to query the database.

Action: db_query

Input: {domain: train, state: {train: {leaveAt: 10:15, day: Friday, departure: Cambridge}}}

User: Can you also help me find a train going to london kings cross ?

System Thought: The user is asking for a train going to London Kings Cross. So the slot is destination: London Kings Cross. Now I need to query the database.

Action: db_query

Input: {domain: train, state: {train: {destination: london kings cross}}}

User: I ' m looking to travel on friday

System Thought: The user wants to travel on Friday. I need to update the day of travel in the database query and find a train that travels on Friday.

Action: db_query

Input: {domain: train, state: {train: {leaveAt: 10:15, day: friday, departure: cambridge, destination: london kings cross}}}

GPT-3.5

User: Please book that train for me now . I need to leave after 10:15 . I need to travel on friday . I 'll be departing from cambridge .

System Thought: The slots for this request are 'departure': 'Cambridge', 'day': 'Friday', and 'leaveAt': 'after 10:15'. I need to query the database to find available trains.

Action: db_query

Input: {domain: train, state: {train: {leaveAt: after 10:15, day: Friday, departure: Cambridge}}}

User: Can you also help me find a train going to london kings cross ?

System Thought: The user is asking for a train going to London Kings Cross, so the slot 'destination' should be set to 'london kings cross'. I can query the database with this information.

Action: db_query

Input: {domain: train, state: {train: {destination: london kings cross}}}

User: I ' m looking to travel on friday

System Thought: The user wants to travel on Friday. I need to update the day of travel in the database query and find a train that travels on Friday.

Action: db_query

Input: {domain: train, state: {train: {leaveAt: 10:15, day: friday, departure: cambridge, destination: london kings cross}}}

Figure 4.8 An excerpt of conversation around the same goal by GPT-3.5 and GPT-4 showing how each model reasons and updates the belief state.

Chapter Overview

This chapter went into detail about the performance of our proposed system from both qualitative and quantitative perspective. Based on automatic evaluation

GPT-4

User: I would also like to book a train , please . I will leave on monday . I need to book a train to london kings cross . I would like to leave after 14:15 .

System: I have found a train for you. It departs from Cambridge at 15:00 on Monday and arrives at London Kings Cross at 15:51. The journey will take 51 minutes and the ticket price is 23.60 pounds. The train ID is TR6028. Would you like to book this train ?

User: I need it to depart from cambridge

System: I have found a train for you. It departs from Cambridge at 05:00 on Monday and arrives at London Kings Cross at 05:51. The journey will take 51 minutes and the ticket price is 23.60 pounds. The train ID is TR7075. Would you like to book this train?

User: I would like to leave after 14:15.

System: I have found a train for you. It departs from Cambridge at 15:00 on Monday and arrives at London Kings Cross at 15:51. The journey will take 51 minutes and the ticket price is 23.60 pounds. The train ID is TR6028. Would you like to book this train?

User: I would like 3 tickets please .

System: I have booked 3 tickets for you. Your booking reference is 00000005.

GPT-3.5

User: I would also like to book a train , please . I will leave on monday . I need to book a train to london kings cross . I would like to leave after 14:15 .

System: Your booking reference for the train service is 00000106.

Figure 4.9 Excerpts of a conversation showing how GPT-4 clarifies and communicates better than GPT-3.5.

measures, we saw that our system has scope for improvement since it is not able to perform as well as other ToD systems that use conventional methods such as supervised learning and deep reinforcement learning. From the qualitative analysis, we saw that LLMs can reason only to an extent and they are still prone to hallucinations and repetitions. We also compared GPT-3.5 and GPT-4 based on performance vs cost and found that larger, more expensive models do not imply better performance.

5 Perspectives

In this chapter, we analyse the capabilities of LLMs for performing task-oriented dialogue based on the experiments and results performed. Section 5.1 describes the contributions of this work in the field of dialogue systems and discusses the results in detail. Section 5.2 describes ways to improve the system and make it more reliable for task-oriented dialogue.

5.1 Discussion

In this work, we explored the reasoning capabilities of LLMs and whether they can be applied to a complex task like task-oriented dialogue. The ReAct paradigm (Yao; J. Zhao, et al., 2022) for prompting LLMs showed promising results for tasks like question answering and even complex tasks like text-based simulated environments such as AlfWorld (Shridhar et al., 2021). We adapted the ReAct strategy to task-oriented dialogue and gave the LLM access to tools that would help it achieve the classical ToD pipeline through its reasoning abilities.

For a complex task such as task-oriented dialogue where each answer depends on the previous context and actions, grounded knowledge and the current state of the environment, we have seen in previous work (Cordier; Urvoy; Lefèvre, et al., 2022) that it takes a lot of resources and work to find the optimal system action. LLMs showed promise in commonsense reasoning (Yao; J. Zhao, et al., 2022; Wei et al., 2022) even outperforming models trained using reinforcement learning and imitation learning. LLMs can indeed reason and perform actions based on the environment they are in and the instructions they are given. Yet, from the results in Table 4.1 we see that the performance of the LLMs fall short compared to handcrafted rules and policy training. On closer inspection of the reasoning traces, we see that the LLM might just be imitating the examples that it was given. This may work for simpler cases with fewer goals to achieve, but when the goals get larger and the user requests become more complicated, the LLMs struggle to understand the user and perform tasks accordingly. Handcrafted rules and dialogue policies perform better mainly because there is a lot of fine-grained control for each step in the pipeline. However, for LLMs, we rely completely on its reasoning abilities and hence we lose the ability to control its reasoning traces and response generation. Difficulty in understanding the user requests also leads to repeated utterances from the user as well as the system leading to a higher number of turns on average to complete the conversation.

LLMs have become very popular in the research community and with the general public due to its ability to produce human-like text effortlessly. The NLG capabilities of LLMs are very impressive as we saw in Table 4.4 and this reflects in ToD as well. However, comparing the generated responses to the gold references may not be the most effective way to gauge their performance. Scores like BLEU (Papineni et al., 2002) and METEOR (Banerjee; Lavie, 2005) that compare the overlap in n-grams between the generated and reference texts have very low scores as we saw since the there are numerous ways in which one can communicate the same idea (Reiter, 2018; Novikova et al., 2017). BertScore (T. Zhang et al., 2020), which uses semantic similarity, tries to overcome this drawback and this reflects in the scores of our stand-alone NLG module using LLM.

In order to assess whether the LLMs are producing responses grounded in reality based on the tool user, we need to cross-check whether the entities that the LLMs are talking about actually exist in the database. Previous work like Hudeček; Dusek, 2023 delexicalises the slots and values before assessing the performance of their system using automatic evaluation methods. In our work, in order to evaluate the success rate, inform rate and book rate, we do not delexicalise the values, but verify that the values exist in the database when calculating the metrics. All the systems in Table 4.1 were evaluated the same way and this gives a more accurate evaluation of our system and assesses its capability to produce text grounded in reality. We see that the inform rate and book rate for our system is not up to the mark compared to other systems. This may be due to the fact that the LLM has trouble updating the states and hence, it may retrieve the wrong results from the database. The LLM also does not ask the user enough clarifying questions, leading to hasty bookings for services that the user does not want. Hence, the inform rate and book rate are very low and by extension, the success rate.

Comparing our work to Hudeček; Dusek, 2023, where the LLM was prompted at each step in the ToD pipeline, we see that the success rate for GPT-3.5 is much higher than what we achieved. This shows that the more fine-grained control we have, the better the system performs. However, we should remember that their evaluation is on delexicalised and static dialogues, hence, the comparison might not be fair.

We must also note that, the LLMs can be biased in certain ways as seen from our experiments. Since LLMs are mostly trained to assist humans, we see that they have a hard time sticking to the role of a user. Also, depending on the training and the dataset used for training, we see that the LLM uses American English, while the input given is in British English.

Since LLMs are trained on a wide variety of tasks, they may not perform all tasks equally well and this is evident in our experiments. ToD is very specific and our experiments prove that it is necessary to have models trained with specific goals in mind in order to achieve good results.

5.2 Future Work

While the results we obtained are not up to the mark compared to current state-of-the-art methods, LLMs have the potential to improve their performance for task-oriented dialogue. Here we outline a few ways that our work can be enhanced which can improve our system.

More descriptive and dynamic prompts

When designing task-oriented dialogue systems, incorporating more detailed information and descriptions for each slot has the potential to enhance performance. Since LLMs are trained to follow instructions, providing comprehensive context and clear descriptions can help the model better understand the relationships between different slots, thereby improving the overall quality of interactions. Detailed slot descriptions ensure that the model is aware of specific attributes and constraints.

Adapt to more datasets

Adapting the work to more diverse datasets in the ToD such as the Schema-Guided Dialogue dataset (Rastogi; Zang, et al., 2020) can help us assess the capabilities of our system more effectively. This will also help us evaluate other aspects of the system.

Get more control by not relying on third party frameworks

While frameworks like LangChain offer convenient tools for building and managing language model applications, gaining more control over the development process by not relying on such frameworks can provide several benefits. By building custom solutions, we get greater flexibility and control in following the LLM's reasoning and generation process and guiding it to take the right action and generate the right response. Being able to pause at each step and control the generation in the ReAct paradigm can help us create a pipeline that closely resembles the classical pipeline, by guiding the LLM on various actions to be taken at each step.

Human evaluation

Incorporating human evaluation into the development process of dialogue systems is essential for ensuring quality and reliability. Human evaluators can provide nuanced feedback that automated metrics might miss, such as the appropriateness of responses, coherence and the fluency of interactions. This feedback is invaluable for identifying weaknesses, refining prompts, and enhancing the model's performance. Human evaluation will help us understand how the system performs in real-world scenarios.

Try more open source language models

Exploring open-source language models for task-oriented dialogue system has many benefits. Open-source models are transparent in their training process which means that we can identify models suited for particular tasks by exploring the datasets used to train them and the training methodologies. By experimenting with different models, we can identify those that best suit our specific needs, whether in terms of accuracy, efficiency, or adaptability. Additionally, using open-source solutions can reduce costs which is a huge factor when building conversational agents.

Study carbon emissions

Studying and addressing the carbon emissions associated with the training and usage of language models is increasingly important in the context of sustainable AI development. Large-scale models require significant computational resources, leading to substantial energy consumption and carbon footprint. Awareness and proactive measures in this area contribute to the responsible and ethical development of AI technologies, ensuring that advancements in language models are balanced with environmental sustainability.

Chapter Overview

This chapter went into the results from the previous chapter in more detail, and attempted to answer why the system performed the way it did. We also discussed possible methods that could potentially improve the system and provide a better analysis of the method.

Conclusion

In this thesis, we explored the use of large language models for performing task-oriented dialogue. Specifically, we studied the reasoning capabilities of LLMs to use appropriate tools to help the user achieve their objective, using the ReAct paradigm. We evaluated the performance of our system using metrics such as success rate that measures the ability of the system to achieve the user goal.

We performed experiments by simulating dialogues between the user and the system agent. The system agent was prompted using the ReAct prompt and provided with an example conversation showing how it should reason using sequences of Thought, Action and Observation. The prompt also demonstrated what tools are to be used in different scenarios. For the user, we experimented using LLMs which proved to be infeasible since they were unable to track the goals and end the conversation based on how the conversation was progressing. In our following experiments, we used a user simulator that followed an agenda-based logic to keep track of the goals and a template-based approach to communicate with the LLM-based system agent.

The evaluation performed showed us that LLMs still have a long way to go with their reasoning capabilities in the context of task-oriented dialogue. We experimented with the GPT-3.5 and GPT-4 models and while GPT-4 performs better than GPT-3.5, it still falls short of other statistical methods that employ reinforcement learning to train a dialogue policy. We must note that employing bigger models does not guarantee better results, especially when we see the trade-off between cost and performance. This was evident from our experiments with GPT-4 and it pushes us to try more cost-effective and open source models.

We also saw that the NLG capabilities of LLMs are extremely good – they effortlessly convert dialogue acts into beautifully framed natural language responses. However, during the intermediate steps of task-oriented dialogue like domain detection, slot filling and forming the dialogue state for performing the database query, we see that the LLM tries to imitate the examples supplied in the prompt and does not always reason or understand the requirements correctly.

While LLMs are truly advancing the research in NLP, our work shows that there is still a lot of potential for improvement in complex tasks such as task-oriented dialogue. In order to exploit the full potential of LLMs, we need to understand how the vast amounts of parameters and embeddings in LLMs interact with each other to perform various tasks. We should also keep in mind the effect these large models have on the environment, both as computing power and their effects in the carbon footprint and costs. Being mindful of the consequences of developing and deploying LLMs, we will be able to build efficient, cost effective systems that can perform our tasks with ease.

Bibliography

AHMAD, Wasi; CHI, Jianfeng; LE, Tu; NORTON, Thomas; TIAN, Yuan; CHANG, Kai-Wei, 2021. Intent Classification and Slot Filling for Privacy Policies. In: *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Online: Association for Computational Linguistics, pp. 4402–4417. Available from DOI: `10.18653/v1/2021.acl-long.340`.

ATHIKKAL, Sagina; JENQ, John, 2022. *Voice Chatbot for Hospitality*. Available from arXiv: `2208.10926 [cs.HC]`.

BANERJEE, Satanjeev; LAVIE, Alon, 2005. METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments. In: GOLDSTEIN, Jade; LAVIE, Alon; LIN, Chin-Yew; VOSS, Clare (eds.). *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*. Ann Arbor, Michigan: Association for Computational Linguistics, pp. 65–72. Available also from: `https://aclanthology.org/W05-0909`.

BROWN, Tom B.; MANN, Benjamin; RYDER, Nick; SUBBIAH, Melanie; KAPLAN, Jared; DHARIWAL, Prafulla; NEELAKANTAN, Arvind; SHYAM, Pranav; SASTRY, Girish; ASKELL, Amanda; AGARWAL, Sandhini; HERBERT-VOSS, Ariel; KRUEGER, Gretchen; HENIGHAN, Tom; CHILD, Rewon; RAMESH, Aditya; ZIEGLER, Daniel M.; WU, Jeffrey; WINTER, Clemens; HESSE, Christopher; CHEN, Mark; SIGLER, Eric; LITWIN, Mateusz; GRAY, Scott; CHESS, Benjamin; CLARK, Jack; BERNER, Christopher; MCCANDLISH, Sam; RADFORD, Alec; SUTSKEVER, Ilya; AMODEI, Dario, 2020. *Language Models are Few-Shot Learners*. Available from arXiv: `2005.14165 [cs.CL]`.

BUDZIANOWSKI, Paweł; WEN, Tsung-Hsien; TSENG, Bo-Hsiang; CASANUEVA, Iñigo; ULTES, Stefan; RAMADAN, Osman; GAŠIĆ, Milica, 2018. MultiWOZ - A Large-Scale Multi-Domain Wizard-of-Oz Dataset for Task-Oriented Dialogue Modelling. In: RILOFF, Ellen; CHIANG, David; HOCKENMAIER, Julia; TSUJII, Jun'ichi (eds.). *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium: Association for Computational Linguistics, pp. 5016–5026. Available from DOI: `10.18653/v1/D18-1547`.

CHAI, Joyce Yue; BUDZIKOWSKA, Malgorzata; HORVATH, Veronika; NICOLOV, Nicolas; KAMBHATLA, Nanda; ZADROZNY, Wlodek, 2001. Natural Language Sales Assistant-A Web-Based Dialog System for Online Sales. In: *IAAI*, pp. 19–26.

CHAO, Guan-Lin; LANE, Ian, 2019. *BERT-DST: Scalable End-to-End Dialogue State Tracking with Bidirectional Encoder Representations from Transformer*. Available from arXiv: `1907.03040 [cs.CL]`.

CHEN, Qian; ZHUO, Zhu; WANG, Wen, 2019. *BERT for Joint Intent Classification and Slot Filling*. Available from arXiv: `1902.10909 [cs.CL]`.

CHO, Kyunghyun; MERRIENBOER, Bart van; GULCEHRE, Caglar; BAHDANAU, Dzmitry; BOUGARES, Fethi; SCHWENK, Holger; BENGIO, Yoshua, 2014. *Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation*. Available from arXiv: 1406.1078 [cs.CL].

CORDIER, Thibault; URVOY, Tanguy; LEFÈVRE, Fabrice; ROJAS-BARAHONA, Lina M., 2022. *Graph Neural Network Policies and Imitation Learning for Multi-Domain Task-Oriented Dialogues*. Available from arXiv: 2210.05252 [cs.CL].

CORDIER, Thibault; URVOY, Tanguy; ROJAS-BARAHONA, Lina M.; LEFÈVRE, Fabrice, 2020. *Diluted Near-Optimal Expert Demonstrations for Guiding Dialogue Stochastic Policy Optimisation*. Available from arXiv: 2012.04687 [cs.CL].

DEVLIN, Jacob; CHANG, Ming-Wei; LEE, Kenton; TOUTANOVA, Kristina, 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In: BURSTEIN, Jill; DORAN, Christy; SOLORIO, Thamar (eds.). *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, pp. 4171–4186. Available from DOI: 10.18653/v1/N19-1423.

ERIC, Mihail; GOEL, Rahul; PAUL, Shachi; KUMAR, Adarsh; SETHI, Abhishek; KU, Peter; GOYAL, Anuj Kumar; AGARWAL, Sanchit; GAO, Shuyang; HAKKANI-TUR, Dilek, 2019. *MultiWOZ 2.1: A Consolidated Multi-Domain Dialogue Dataset with State Corrections and State Tracking Baselines*. Available from arXiv: 1907.01669 [cs.CL].

GAO, Shuyang; SETHI, Abhishek; AGARWAL, Sanchit; CHUNG, Tagyoung; HAKKANI-TUR, Dilek, 2019. Dialog State Tracking: A Neural Reading Comprehension Approach. In: NAKAMURA, Satoshi; GASIC, Milica; ZUKERMAN, Ingrid; SKANTZE, Gabriel; NAKANO, Mikio; PAPANGELIS, Alexandros; ULTES, Stefan; YOSHINO, Koichiro (eds.). *Proceedings of the 20th Annual SIGdial Meeting on Discourse and Dialogue*. Stockholm, Sweden: Association for Computational Linguistics, pp. 264–273. Available from DOI: 10.18653/v1/W19-5932.

GUNASEKARA, Chulaka; KIM, Seokhwan; D'HARO, Luis Fernando; RASTOGI, Abhinav; CHEN, Yun-Nung; ERIC, Mihail; HEDAYATNIA, Behnam; GOPALAKRISHNAN, Karthik; LIU, Yang; HUANG, Chao-Wei; HAKKANI-TÜR, Dilek; LI, Jinchao; ZHU, Qi; LUO, Lingxiao; LIDEN, Lars; HUANG, Kaili; SHAYANDEH, Shahin; LIANG, Runze; PENG, Baolin; ZHANG, Zheng; SHUKLA, Swadheen; HUANG, Minlie; GAO, Jianfeng; MEHRI, Shikib; FENG, Yulan; GORDON, Carla; ALAVI, Seyed Hossein; TRAUM, David; ESKENAZI, Maxine; BEIRAMI, Ahmad; EUNJOON; CHO; CROOK, Paul A.; DE, Ankita; GERAMIFARD, Alborz; KOTTUR, Satwik; MOON, Seungwhan; PODDAR, Shivani; SUBBA, Rajen, 2020. *Overview of the Ninth Dialog System Technology Challenge: DSTC9*. Available from arXiv: 2011.06486 [cs.CL].

HAKKANI-TÜR, Dilek; TUR, Gokhan; CELIKYILMAZ, Asli; CHEN, Yun-Nung; GAO, Jianfeng; DENG, Li; WANG, Ye-Yi, 2016. Multi-Domain Joint Semantic Frame Parsing Using Bi-Directional RNN-LSTM. In: *Proc. Interspeech 2016*, pp. 715–719. ISSN 2958-1796. Available from DOI: 10.21437/Interspeech.2016-402.

HAM, Donghoon; LEE, Jeong-Gwan; JANG, Youngsoo; KIM, Kee-Eung, 2020. End-to-End Neural Pipeline for Goal-Oriented Dialogue Systems using GPT-2. In: JURAFSKY, Dan; CHAI, Joyce; SCHLUTER, Natalie; TETREAULT, Joel (eds.). *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, pp. 583–592. Available from DOI: 10.18653/v1/2020.acl-main.54.

HAN, Ting; LIU, Ximing; TAKANABU, Ryuichi; LIAN, Yixin; HUANG, Chongxuan; WAN, Dazhen; PENG, Wei; HUANG, Minlie, 2021. MultiWOZ 2.3: A Multi-domain Task-Oriented Dialogue Dataset Enhanced with Annotation Corrections and Co-Reference Annotation. In: WANG, Lu; FENG, Yansong; HONG, Yu; HE, Ruifang (eds.). *Natural Language Processing and Chinese Computing*. Cham: Springer International Publishing, pp. 206–218. ISBN 978-3-030-88483-3.

HECK, Michael; NIEKERK, Carel van; LUBIS, Nurul; GEISHAUSER, Christian; LIN, Hsien-Chin; MORESI, Marco; GASIC, Milica, 2020. TripPy: A Triple Copy Strategy for Value Independent Neural Dialog State Tracking. In: PIETQUIN, Olivier; MURESAN, Smaranda; CHEN, Vivian; KENNINGTON, Casey; VANDYKE, David; DETHLEFS, Nina; INOUE, Koji; EKSTEDT, Erik; ULTES, Stefan (eds.). *Proceedings of the 21th Annual Meeting of the Special Interest Group on Discourse and Dialogue*. 1st virtual meeting: Association for Computational Linguistics, pp. 35–44. Available from DOI: 10.18653/v1/2020.sigdial-1.4.

HENDERSON, Matthew; THOMSON, Blaise; YOUNG, Steve, 2013. Deep Neural Network Approach for the Dialog State Tracking Challenge. In: ESKENAZI, Maxine; STRUBE, Michael; DI EUGENIO, Barbara; WILLIAMS, Jason D. (eds.). *Proceedings of the SIGDIAL 2013 Conference*. Metz, France: Association for Computational Linguistics, pp. 467–471. Available also from: https://aclanthology.org/W13-4073.

HOCHREITER, Sepp; SCHMIDHUBER, Jürgen, 1997. Long Short-Term Memory. *Neural Comput.* Vol. 9, no. 8, pp. 1735–1780. ISSN 0899-7667. Available from DOI: 10.1162/neco.1997.9.8.1735.

HOSSEINI-ASL, Ehsan; MCCANN, Bryan; WU, Chien-Sheng; YAVUZ, Semih; SOCHER, Richard, 2022. *A Simple Language Model for Task-Oriented Dialogue*. Available from arXiv: 2005.00796 [cs.CL].

HU, Yushi; LEE, Chia-Hsuan; XIE, Tianbao; YU, Tao; SMITH, Noah A.; OSTENDORF, Mari, 2022. In-Context Learning for Few-Shot Dialogue State Tracking. In: GOLDBERG, Yoav; KOZAREVA, Zornitsa; ZHANG, Yue (eds.). *Findings of the Association for Computational Linguistics: EMNLP 2022*. Abu Dhabi, United Arab Emirates: Association for Computational Linguistics, pp. 2627–2643. Available from DOI: 10.18653/v1/2022.findings-emnlp.193.

HUDEČEK, Vojtěch; DUSEK, Ondrej, 2023. Are Large Language Models All You Need for Task-Oriented Dialogue? In: STOYANCHEV, Svetlana; JOTY, Shafiq; SCHLANGEN, David; DUSEK, Ondrej; KENNINGTON, Casey; ALIKHANI, Malihe (eds.). *Proceedings of the 24th Annual Meeting of the Special Interest Group on Discourse and Dialogue*. Prague, Czechia: Association for Computational Linguistics, pp. 216–228. Available from DOI: 10.18653/v1/2023.sigdial-1.21.

JACQMIN, Léo; DRUART, Lucas; ESTÈVE, Yannick; FAVRE, Benoît; M ROJAS, Lina; VIELZEUF, Valentin, 2023. OLISIA: a Cascade System for Spoken Dialogue State Tracking. In: CHEN, Yun-Nung; CROOK, Paul; GALLEY, Michel; GHAZARIAN, Sarik; GUNASEKARA, Chulaka; GUPTA, Raghav; HEDAYATNIA, Behnam; KOTTUR, Satwik; MOON, Seungwhan; ZHANG, Chen (eds.). *Proceedings of The Eleventh Dialog System Technology Challenge*. Prague, Czech Republic: Association for Computational Linguistics, pp. 95–104. Available also from: `https://aclanthology.org/2023.dstc-1.12`.

JACQMIN, Léo; ROJAS BARAHONA, Lina M.; FAVRE, Benoit, 2022. “Do you follow me?”: A Survey of Recent Approaches in Dialogue State Tracking. In: LEMON, Oliver; HAKKANI-TUR, Dilek; LI, Junyi Jessy; ASHRAFZADEH, Arash; GARCIA, Daniel Hernández; ALIKHANI, Malihe; VANDYKE, David; DUŠEK, Ondřej (eds.). *Proceedings of the 23rd Annual Meeting of the Special Interest Group on Discourse and Dialogue*. Edinburgh, UK: Association for Computational Linguistics, pp. 336–350. Available from DOI: `10.18653/v1/2022.sigdial-1.33`.

JURAFSKY, Daniel; MARTIN, James H., 2000. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. 1st. USA: Prentice Hall PTR. ISBN 0130950696.

KALE, Mihir; RASTOGI, Abhinav, 2020. Template Guided Text Generation for Task-Oriented Dialogue. In: WEBBER, Bonnie; COHN, Trevor; HE, Yulan; LIU, Yang (eds.). *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Online: Association for Computational Linguistics, pp. 6505–6520. Available from DOI: `10.18653/v1/2020.emnlp-main.527`.

KELLEY, J. F., 1984. An iterative design methodology for user-friendly natural language office information applications. *ACM Trans. Inf. Syst.* Vol. 2, no. 1, pp. 26–41. ISSN 1046-8188. Available from DOI: `10.1145/357417.357420`.

LEE, Sungjin; ZHU, Qi; TAKANOBU, Ryuichi; ZHANG, Zheng; ZHANG, Yaoqin; LI, Xiang; LI, Jinchao; PENG, Baolin; LI, Xiujun; HUANG, Minlie; GAO, Jianfeng, 2019. ConvLab: Multi-Domain End-to-End Dialog System Platform. In: COSTA-JUSSÀ, Marta R.; ALFONSECA, Enrique (eds.). *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. Florence, Italy: Association for Computational Linguistics, pp. 64–69. Available from DOI: `10.18653/v1/P19-3011`.

LEI, Wenqiang; JIN, Xisen; KAN, Min-Yen; REN, Zhaochun; HE, Xiangnan; YIN, Dawei, 2018. Sequicity: Simplifying Task-oriented Dialogue Systems with Single Sequence-to-Sequence Architectures. In: GUREVYCH, Iryna; MIYAO, Yusuke (eds.). *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Melbourne, Australia: Association for Computational Linguistics, pp. 1437–1447. Available from DOI: `10.18653/v1/P18-1133`.

LEWIS, Mike; LIU, Yinhan; GOYAL, Naman; GHAZVININEJAD, Marjan; MOHAMED, Abdelrahman; LEVY, Omer; STOYANOV, Ves; ZETTLEMOYER, Luke, 2019. *BART: Denoising Sequence-to-Sequence Pre-training for Natural Language*

Generation, Translation, and Comprehension. Available from arXiv: 1910.13461 [cs.CL].

LIPTON, Zachary C.; LI, Xiujun; GAO, Jianfeng; LI, Lihong; AHMED, Faisal; DENG, Li, 2017. *BBQ-Networks: Efficient Exploration in Deep Reinforcement Learning for Task-Oriented Dialogue Systems*. Available from arXiv: 1608.05081 [cs.LG].

LIU, Bing; LANE, Ian, 2016. *Attention-Based Recurrent Neural Network Models for Joint Intent Detection and Slot Filling*. Available from arXiv: 1609.01454 [cs.CL].

MADOTTO, Andrea; LIU, Zihan; LIN, Zhaojiang; FUNG, Pascale, 2020. *Language Models as Few-Shot Learner for Task-Oriented Dialogue Systems*. Available from arXiv: 2008.06239 [cs.CL].

MCTEAR, Michael F., 2020. Conversational AI: Dialogue Systems, Conversational Agents, and Chatbots. *Synthesis Lectures on Human Language Technologies*. Available also from: https://api.semanticscholar.org/CorpusID:228935772.

MILLE, Simon; DASIOPOULOU, Stamatia; FISAS, Beatriz; WANNER, Leo, 2019. Teaching FORGe to Verbalize DBpedia Properties in Spanish. In: DEEMTER, Kees van; LIN, Chenghua; TAKAMURA, Hiroya (eds.). *Proceedings of the 12th International Conference on Natural Language Generation*. Tokyo, Japan: Association for Computational Linguistics, pp. 473–483. Available from DOI: 10.18653/v1/W19-8659.

MRKŠIĆ, Nikola; SÉAGHDHA, Diarmuid Ó; THOMSON, Blaise; GAŠIĆ, Milica; SU, Pei-Hao; VANDYKE, David; WEN, Tsung-Hsien; YOUNG, Steve, 2015. *Multi-domain Dialog State Tracking using Recurrent Neural Networks*. Available from arXiv: 1506.07190 [cs.CL].

NICULESCU, Andreea I.; JIANG, Ridong; KIM, Seokhwan; YEO, Kheng Hui; D'HARO, Luis F.; NISWAR, Arthur; BANCHS, Rafael E., 2014. SARA: Singapore's Automated Responsive Assistant, A Multimodal Dialogue System for Touristic Information. In: AWAN, Irfan; YOUNAS, Muhammad; FRANCH, Xavier; QUER, Carme (eds.). *Mobile Web Information Systems*. Cham: Springer International Publishing, pp. 153–164. ISBN 978-3-319-10359-4.

NOVIKOVA, Jekaterina; DUŠEK, Ondřej; CERCAS CURRY, Amanda; RIESER, Verena, 2017. Why We Need New Evaluation Metrics for NLG. In: PALMER, Martha; HWA, Rebecca; RIEDEL, Sebastian (eds.). *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Copenhagen, Denmark: Association for Computational Linguistics, pp. 2241–2252. Available from DOI: 10.18653/v1/D17-1238.

O'SHEA, Keiron; NASH, Ryan, 2015. *An Introduction to Convolutional Neural Networks*. Available from arXiv: 1511.08458 [cs.NE].

OPENAI et al., 2024. *GPT-4 Technical Report*. Available from arXiv: 2303.08774 [cs.CL].

OUYANG, Long; WU, Jeff; JIANG, Xu; ALMEIDA, Diogo; WAINWRIGHT, Carroll L.; MISHKIN, Pamela; ZHANG, Chong; AGARWAL, Sandhini; SLAMA, Katarina; RAY, Alex; SCHULMAN, John; HILTON, Jacob; KELTON, Fraser; MILLER, Luke; SIMENS, Maddie; ASKELL, Amanda; WELINDER, Peter; CHRISTIANO, Paul; LEIKE, Jan; LOWE, Ryan, 2022. *Training language models to follow instructions with human feedback*. Available from arXiv: 2203.02155 [cs.CL].

PAPINENI, Kishore; ROUKOS, Salim; WARD, Todd; ZHU, Wei-Jing, 2002. Bleu: a Method for Automatic Evaluation of Machine Translation. In: ISABELLE, Pierre; CHARNIAK, Eugene; LIN, Dekang (eds.). *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*. Philadelphia, Pennsylvania, USA: Association for Computational Linguistics, pp. 311–318. Available from DOI: 10.3115/1073083.1073135.

PELLOM, B.; WARD, W.; HANSEN, J.; COLE, R.; HACIOGLU, K.; ZHANG, J.; YU, X.; PRADHAN, S., 2001. University of Colorado Dialogue Systems for Travel and Navigation. In: *Proceedings of the First International Conference on Human Language Technology Research*. Available also from: https://aclanthology.org/H01-1073.

PENG, Baolin; LI, Chunyuan; LI, Jinchao; SHAYANDEH, Shahin; LIDEN, Lars; GAO, Jianfeng, 2021. Soloist: Building Task Bots at Scale with Transfer Learning and Machine Teaching. *Transactions of the Association for Computational Linguistics*. Vol. 9, pp. 807–824. Available from DOI: 10.1162/tacl_a_00399.

RADFORD, Alec; NARASIMHAN, Karthik; SALIMANS, Tim; SUTSKEVER, Ilya, et al., 2018. Improving language understanding by generative pre-training.

RADFORD, Alec; WU, Jeff; CHILD, Rewon; LUAN, David; AMODEI, Dario; SUTSKEVER, Ilya, 2019. Language Models are Unsupervised Multitask Learners.

RAFFEL, Colin; ELLIS, Daniel PW, 2015. Feed-forward networks with attention can solve some long-term memory problems. *arXiv preprint arXiv:1512.08756*.

RAFFEL, Colin; SHAZEER, Noam; ROBERTS, Adam; LEE, Katherine; NARANG, Sharan; MATENA, Michael; ZHOU, Yanqi; LI, Wei; LIU, Peter J., 2020. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *Journal of Machine Learning Research*. Vol. 21, no. 140, pp. 1–67. Available also from: http://jmlr.org/papers/v21/20-074.html.

RASTOGI, Abhinav; HAKKANI-TUR, Dilek; HECK, Larry, 2018. *Scalable Multi-Domain Dialogue State Tracking*. Available from arXiv: 1712.10224 [cs.CL].

RASTOGI, Abhinav; ZANG, Xiaoxue; SUNKARA, Srinivas; GUPTA, Raghav; KHAITAN, Pranav, 2020. *Towards Scalable Multi-domain Conversational Agents: The Schema-Guided Dialogue Dataset*. Available from arXiv: 1909.05855 [cs.CL].

REITER, Ehud, 2018. A Structured Review of the Validity of BLEU. *Computational Linguistics*. Vol. 44, no. 3, pp. 393–401. Available from DOI: 10.1162/coli_a_00322.

RIESER, Verena; LEMON, Oliver, 2011. Reinforcement Learning for Adaptive Dialogue Systems: a data-driven methodology for Dialogue Management and Natural Language Generation. In.

Rojas-Barahona, Lina M.; Gašić, Milica; Mrkšić, Nikola; Su, Pei-Hao; Ultes, Stefan; Wen, Tsung-Hsien; Young, Steve, 2016. Exploiting Sentence and Context Representations in Deep Neural Models for Spoken Language Understanding. In: Matsumoto, Yuji; Prasad, Rashmi (eds.). *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. Osaka, Japan: The COLING 2016 Organizing Committee, pp. 258–267. Available also from: `https://aclanthology.org/C16-1025`.

Santhanam, Sashank; Shaikh, Samira, 2019. Towards Best Experiment Design for Evaluating Dialogue System Output. In: Deemter, Kees van; Lin, Chenghua; Takamura, Hiroya (eds.). *Proceedings of the 12th International Conference on Natural Language Generation*. Tokyo, Japan: Association for Computational Linguistics, pp. 88–94. Available from doi: `10.18653/v1/W19-8610`.

Sarikaya, R.; Crook, P. A.; Marin, A.; Jeong, M.; Robichaud, J.P.; Celikyilmaz, A.; Kim, Y.B.; Rochette, A.; Khan, O. Z.; Liu, X.; Boies, D.; Anastasakos, T.; Feizollahi, Z.; Ramesh, N.; Suzuki, H.; Holenstein, R.; Krawczyk, E.; Radostev, V., 2016. An overview of end-to-end language understanding and dialog management for personal digital assistants. In: *2016 IEEE Spoken Language Technology Workshop (SLT)*, pp. 391–397. Available from doi: `10.1109/SLT.2016.7846294`.

Schatzmann, Jost; Thomson, Blaise; Weilhammer, Karl; Ye, Hui; Young, Steve, 2007. Agenda-Based User Simulation for Bootstrapping a POMDP Dialogue System. In: Sidner, Candace; Schultz, Tanja; Stone, Matthew; Zhai, ChengXiang (eds.). *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Companion Volume, Short Papers*. Rochester, New York: Association for Computational Linguistics, pp. 149–152. Available also from: `https://aclanthology.org/N07-2038`.

Shah, Harshil; Wilcke, Arthur; Cobzarenco, Marius; Cobzarenco, Cristi; Challis, Edward; Barber, David, 2023. *Generalized Multiple Intent Conditioned Slot Filling*. Available from arXiv: `2305.11023 [cs.CL]`.

Shridhar, Mohit; Yuan, Xingdi; Côté, Marc-Alexandre; Bisk, Yonatan; Trischler, Adam; Hausknecht, Matthew, 2021. ALFWorld: Aligning Text and Embodied Environments for Interactive Learning. In: *Proceedings of the International Conference on Learning Representations (ICLR)*. Available also from: `https://arxiv.org/abs/2010.03768`.

Su, Pei-Hao; Budzianowski, Pawel; Ultes, Stefan; Gasic, Milica; Young, Steve, 2017. *Sample-efficient Actor-Critic Reinforcement Learning with Supervised Data for Dialogue Management*. Available from arXiv: `1707.00130 [cs.CL]`.

Takanobu, Ryuichi; Zhu, Qi; Li, Jinchao; Peng, Baolin; Gao, Jianfeng; Huang, Minlie, 2020. Is Your Goal-Oriented Dialog Model Performing Really Well? Empirical Analysis of System-wise Evaluation. In: Pietquin, Olivier; Muresan, Smaranda; Chen, Vivian; Kennington, Casey; Vandyke, David; Dethlefs, Nina; Inoue, Koji; Ekstedt, Erik; Ultes, Stefan (eds.). *Proceedings of the 21th Annual Meeting of the Special Interest Group on Discourse and Dialogue*.

1st virtual meeting: Association for Computational Linguistics, pp. 297–310. Available from DOI: 10.18653/v1/2020.sigdial-1.37.

Tseng, Bo-Hsiang; Cheng, Jianpeng; Fang, Yimai; Vandyke, David, 2020. *A Generative Model for Joint Natural Language Understanding and Generation*. Available from arXiv: 2006.07499 [cs.CL].

Tuan, Yi-Lin; Zhang, Jinzhi; Li, Yujia; Lee, Hung-yi, 2018. *Proximal Policy Optimization and its Dynamic Version for Sequence Generation*. Available from arXiv: 1808.07982 [cs.CL].

Ultes, Stefan; Rojas-Barahona, Lina M.; Su, Pei-Hao; Vandyke, David; Kim, Dongho; Casanueva, Iñigo; Budzianowski, Paweł; Mrkšić, Nikola; Wen, Tsung-Hsien; Gašić, Milica; Young, Steve, 2017. PyDial: A Multi-domain Statistical Dialogue System Toolkit. In: Bansal, Mohit; Ji, Heng (eds.). *Proceedings of ACL 2017, System Demonstrations*. Vancouver, Canada: Association for Computational Linguistics, pp. 73–78. Available also from: https://aclanthology.org/P17-4013.

Vaswani, Ashish; Shazeer, Noam; Parmar, Niki; Uszkoreit, Jakob; Jones, Llion; Gomez, Aidan N; Kaiser, Łukasz; Polosukhin, Illia, 2017. Attention is All you Need. In: Guyon, I.; Luxburg, U. Von; Bengio, S.; Wallach, H.; Fergus, R.; Vishwanathan, S.; Garnett, R. (eds.). *Advances in Neural Information Processing Systems*. Curran Associates, Inc. Vol. 30. Available also from: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Wahde, Mattias; Virgolin, Marco, 2022. Conversational agents: Theory and applications. In: *HANDBOOK ON COMPUTER LEARNING AND INTELLIGENCE: Volume 2: Deep Learning, Intelligent Control and Evolutionary Computation*. World Scientific, pp. 497–544.

Walker, Marilyn A.; Litman, Diane J.; Kamm, Candace A.; Abella, Alicia, 1997. PARADISE: a framework for evaluating spoken dialogue agents. In: *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics and Eighth Conference of the European Chapter of the Association for Computational Linguistics*. Madrid, Spain: Association for Computational Linguistics, pp. 271–280. ACL '98/EACL '98. Available from DOI: 10.3115/976909.979652.

Wang, Sihan; Zhou, Kaijie; Lai, Kunfeng; Shen, Jianping, 2020. Task-Completion Dialogue Policy Learning via Monte Carlo Tree Search with Dueling Network. In: Webber, Bonnie; Cohn, Trevor; He, Yulan; Liu, Yang (eds.). *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Online: Association for Computational Linguistics, pp. 3461–3471. Available from DOI: 10.18653/v1/2020.emnlp-main.278.

Wei, Jason; Wang, Xuezhi; Schuurmans, Dale; Bosma, Maarten; Chi, Ed Huai-hsin; Xia, F.; Le, Quoc; Zhou, Denny, 2022. Chain of Thought Prompting Elicits Reasoning in Large Language Models. *ArXiv*. Vol. abs/2201.11903. Available also from: https://api.semanticscholar.org/CorpusID:246411621.

WEISZ, Gellért; BUDZIANOWSKI, Paweł; SU, Pei-Hao; GAŠIĆ, Milica, 2018. Sample efficient deep reinforcement learning for dialogue systems with large action spaces. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*. Vol. 26, no. 11, pp. 2083–2097.

WEIZENBAUM, Joseph, 1966. ELIZA—a computer program for the study of natural language communication between man and machine. *Commun. ACM*. Vol. 9, no. 1, pp. 36–45. ISSN 0001-0782. Available from DOI: `10.1145/365153.365168`.

WEN, Tsung-Hsien; GAŠIĆ, Milica; MRKŠIĆ, Nikola; SU, Pei-Hao; VANDYKE, David; YOUNG, Steve, 2015. Semantically Conditioned LSTM-based Natural Language Generation for Spoken Dialogue Systems. In: MÀRQUEZ, Lluís; CALLISON-BURCH, Chris; SU, Jian (eds.). *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal: Association for Computational Linguistics, pp. 1711–1721. Available from DOI: `10.18653/v1/D15-1199`.

WEN, Tsung-Hsien; VANDYKE, David; MRKŠIĆ, Nikola; GAŠIĆ, Milica; ROJAS-BARAHONA, Lina M.; SU, Pei-Hao; ULTES, Stefan; YOUNG, Steve, 2017. A Network-based End-to-End Trainable Task-oriented Dialogue System. In: LAPATA, Mirella; BLUNSOM, Phil; KOLLER, Alexander (eds.). *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*. Valencia, Spain: Association for Computational Linguistics, pp. 438–449. Available also from: `https://aclanthology.org/E17-1042`.

WILLIAMS, Jason D.; ASADI, Kavosh; ZWEIG, Geoffrey, 2017. *Hybrid Code Networks: practical and efficient end-to-end dialog control with supervised and reinforcement learning*. Available from arXiv: `1702.03274 [cs.AI]`.

WU, Chien-Sheng; HOI, Steven C.H.; SOCHER, Richard; XIONG, Caiming, 2020. TOD-BERT: Pre-trained Natural Language Understanding for Task-Oriented Dialogue. In: WEBBER, Bonnie; COHN, Trevor; HE, Yulan; LIU, Yang (eds.). *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Online: Association for Computational Linguistics, pp. 917–929. Available from DOI: `10.18653/v1/2020.emnlp-main.66`.

WU, Yonghui; SCHUSTER, Mike; CHEN, Zhifeng; LE, Quoc V.; NOROUZI, Mohammad; MACHEREY, Wolfgang; KRIKUN, Maxim; CAO, Yuan; GAO, Qin; MACHEREY, Klaus; KLINGNER, Jeff; SHAH, Apurva; JOHNSON, Melvin; LIU, Xiaobing; KAISER, Łukasz; GOUWS, Stephan; KATO, Yoshikiyo; KUDO, Taku; KAZAWA, Hideto; STEVENS, Keith; KURIAN, George; PATIL, Nishant; WANG, Wei; YOUNG, Cliff; SMITH, Jason; RIESA, Jason; RUDNICK, Alex; VINYALS, Oriol; CORRADO, Greg; HUGHES, Macduff; DEAN, Jeffrey, 2016. *Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation*. Available from arXiv: `1609.08144 [cs.CL]`.

YAO, Shunyu; CHEN, Howard; YANG, John; NARASIMHAN, Karthik, 2023. *WebShop: Towards Scalable Real-World Web Interaction with Grounded Language Agents*. Available from arXiv: `2207.01206 [cs.CL]`.

YAO, Shunyu; ZHAO, Jeffrey; YU, Dian; DU, Nan; SHAFRAN, Izhak; NARASIMHAN, Karthik; CAO, Yuan, 2022. ReAct: Synergizing Reasoning and Acting in Language Models. *arXiv preprint arXiv:2210.03629*.

ZANG, Xiaoxue; RASTOGI, Abhinav; SUNKARA, Srinivas; GUPTA, Raghav; ZHANG, Jianguo; CHEN, Jindong, 2020. *MultiWOZ 2.2 : A Dialogue Dataset with Additional Annotation Corrections and State Tracking Baselines*. Available from arXiv: `2007.12720 [cs.CL]`.

ZHANG, Jianguo; QIAN, Kun; LIU, Zhiwei; HEINECKE, Shelby; MENG, Rui; LIU, Ye; YU, Zhou; SAVARESE, Silvio; XIONG, Caiming, 2023. DialogStudio: Towards Richest and Most Diverse Unified Dataset Collection for Conversational AI. *arXiv preprint arXiv:2307.10172*.

ZHANG, Tianyi; KISHORE, Varsha; WU, Felix; WEINBERGER, Kilian Q.; ARTZI, Yoav, 2020. *BERTScore: Evaluating Text Generation with BERT*. Available from arXiv: `1904.09675 [cs.CL]`.

ZHAO, Lin; FENG, Zhe, 2018. Improving Slot Filling in Spoken Language Understanding with Joint Pointer and Attention. In: GUREVYCH, Iryna; MIYAO, Yusuke (eds.). *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Melbourne, Australia: Association for Computational Linguistics, pp. 426–431. Available from DOI: `10.18653/v1/P18-2068`.

ZHU, Qi; GEISHAUSER, Christian; LIN, Hsien-chin; NIEKERK, Carel van; PENG, Baolin; ZHANG, Zheng; HECK, Michael; LUBIS, Nurul; WAN, Dazhen; ZHU, Xiaochen; GAO, Jianfeng; GAŠIĆ, Milica; HUANG, Minlie, 2022. ConvLab-3: A Flexible Dialogue System Toolkit Based on a Unified Data Format. *arXiv preprint arXiv:2211.17148*. Available also from: `http://arxiv.org/abs/2211.17148`.

ZHU, Qi; ZHANG, Zheng; FANG, Yan; LI, Xiang; TAKANOBU, Ryuichi; LI, Jinchao; PENG, Baolin; GAO, Jianfeng; ZHU, Xiaoyan; HUANG, Minlie, 2020. ConvLab-2: An Open-Source Toolkit for Building, Evaluating, and Diagnosing Dialogue Systems. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*.

ZHU, Yukun; KIROS, Ryan; ZEMEL, Rich; SALAKHUTDINOV, Ruslan; URTASUN, Raquel; TORRALBA, Antonio; FIDLER, Sanja, 2015. Aligning Books and Movies: Towards Story-Like Visual Explanations by Watching Movies and Reading Books. In: *The IEEE International Conference on Computer Vision (ICCV)*.

ZILKA, Lukas; JURCICEK, Filip, 2015. *Incremental LSTM-based Dialog State Tracker*. Available from arXiv: `1507.03471 [cs.CL]`.

List of Figures

List of Tables

List of Abbreviations

NLP Natural Language Processing

LLM Large Language Models

ToD Task-oriented Dialogue

AI Artificial Intelligence

NLU Natural Language Understanding

HDC Handcrafted

RNN Recurrent Neural Networks

DST Dialogue State Tracking

DM Dialogue Management

ACER Actor-Critic with Experience Replay

PPO Proximal Policy Optimization

NLG Natural Language Generation

BLEU BiLingual Evaluation Understudy

JGA Joint Goal Accuracy

SER Slot Error Rate

METEOR Metric for Evaluation of Translation with Explicit ORdering

RL Reinforcement Learning

GPT Generative Pretrained Transformer

LSTM Long Short Term Memory

GRU Gated Recurrent Unit

CNN Convolutional Neural Networks

RLHF Reinforcement Learning from Human Feedback

CoT Chain-of-Thought

A Attachments

Additional materials related to this work are provided as electronic attachment as a `.zip` archive file. The contents of the file include:

- The code developed for the project.
- The files containing the dialogue simulations run on GPT-3.5 and GPT-4.
- The files containing the NLG outputs from our experiments.
- The prompts used in our experiments.
- The randomly sampled dialogues from the simulations, which were selected for the qualitative analysis.
- The examples used for few-shot prompting in our experiments.
- The files containing the evaluation metrics generated for our experiments.