Institut für Maschinelle Sprachverarbeitung

Institut für Maschinelle Sprachverarbeitung
Universität Stuttgart
Pfaffenwaldring 5b
70569 Stuttgart

Master Thesis

Prompt-based methods for Dialog State Tracking

Start Date: 01.08.2022
End Date: 01.02.2023

Mandava, Sai Pavan
M.Sc. Computational Linguistics
Mat.Nr.: 3461015
st169661@stud.uni-stuttgart.de

Supervisor/Examiner	Prof. Dr. Thang Vu
Examiner	Dr. Antje Schweitzer

DRAFT

Erklärung (Statement of Authorship)

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig verfasst habe und dabei keine andere als die angegebene Literatur verwendet habe. Alle Zitate und sinngemäßen Entlehnungen sind als solche unter genauer Angabe der Quelle gekennzeichnet. Die eingereichte Arbeit ist weder vollständig noch in wesentlichen Teilen Gegenstand eines anderen Prüfungsverfahrens gewesen. Sie ist weder vollständig noch in Teilen bereits veröffentlicht. Die beigefügte elektronische Version stimmt mit dem Druckexemplar überein.

(Mandava, Sai Pavan)

Non-binding Translation:

This thesis is the result of my own independent work, and any material from work of others which is used either verbatim or indirectly in the text is credited to the author including details about the exact source in the text. This work has not been part of any other previous examination, neither completely nor in parts. It has neither completely nor partially been published before. The submitted electronic version is identical to this print version.

(Mandava, Sai Pavan)

Abstract

Dialog State Tracking (DST) is an essential component of task-oriented dialogue systems, which helps the dialog system understand the user's requirements for completing specific tasks. In this thesis, prompt-based methods for DST in task-oriented dialogue systems are explored by utilizing the MultiWOZ dataset. This approach does not rely on the pre-defined set of slots and their possible values. It can also be costly to label all the slots and values to train the DST models, especially for new domains. The prompt-based approach focuses on learning the DST model efficiently under low-resource few-shot settings. To examine the impact of prompt-based methods, a baseline pre-trained language model, SOLOIST[1], is fine-tuned to generate belief states as a word sequence. In the prompt-based approach, the SOLOIST model is fine-tuned on limited labeled training data to generate the slots directly from values. Further, multi-prompt methods[2] are applied to investigate the potential improvement in the slot generation performance. Experimental results show prompt-based methods significantly outperformed the baseline model under low-resource settings. Analysis of outputs shows prompt-based approach has some drawbacks due to the existing belief states annotation system in the MultiWOZ dataset. One limitation is that the prompt-based methods cannot generate multiple slots for repeated value candidates, as the slots are generated by passing values to the prompt function. The data, code and steps to reproduce results are publicly available here[3].

[1]A Single Pre-trained Language Model (GPT-2) for task-oriented dialog systems
[2]Multi-prompt methods: *Prompt Decomposition, Prompt Ensembling, Prompt Augmentation*
[3]https://git.pavanmandava.com/pavan/master-thesis

Contents

1 Introduction

Dialog State Tracking (DST) is an essential module in dialog systems, which is responsible for tracking the user goals in the form of dialog states based on the entire dialog history. In dialog systems, "dialog states" - also known as "belief states" contains a set of *(slot, value)* pairs for each turn of the dialog history. The *(slot, value)* pairs hold specific pieces of information required for the dialog system to perform the task and help in generating the responses. The values of the slots can change when the user provides more information or accepts the system recommendations. Existing data-driven methods and neural models for individual dialog modules (NLU, DST, NLG) and end-to-end dialog systems show promising results, but they need large amounts of task-specific training data, which is rarely available for new tasks. These neural DST models do not generalize well on new domains with limited data (Li et al., 2021). For task-specific DST, collecting dialog state labels can be costly and time-consuming, requiring domain experts to indicate all possible (*slot, value*) pairs for each turn of the dialog history. A typical task-oriented dialog system contains an ontology for each domain, with a pre-defined set of slots and all possible values for each domain. In real-world applications, defining all possible slots and values for DST is difficult due to the increasing number of new domains and the evolving needs of the users.

Prompt-based learning *("pre-train, prompt, and predict")* is a new paradigm in NLP that aims to predict the probability of text directly from the pre-trained LM. This framework is powerful as it allows the language model to be *pre-trained* on massive amounts of raw text, and by defining a new prompting function the model can perform *few-shot* or even *zero-shot* learning (Liu et al., 2021). The large pre-trained language models (PLMs) are supposed to be useful in few-shot scenarios where the task-related training data is limited, as they can be probed for task-related knowledge efficiently by using a prompt. One example of such large pre-trained language models is GPT-3 (Brown et al., 2020) - *"Language Models are Few-Shot Learners"*. Madotto et al. (2021) created an end-to-end chatbot (Few-

Shot Bot) using *prompt-based few-shot learning* learning and achieved comparable results to those of state-of-the-art. Prompting methods are particularly helpful in few-shot learning where domain-related data is limited. *Fixed-prompt LM tuning* is a fine-tuning strategy for downstream tasks, where the LM parameters are tuned with fixed prompts to help LM understand the task. This can be achieved by applying a discrete textual prompt template to the data used for fine-tuning the PLM.

Prompt-based learning for few-shot DST with limited labeled domains is still under-explored. Recently, Yang et al. (2022) proposed a new prompt learning framework for few-shot DST. This work designed a *value-based prompt* and an *inverse prompt* mechanism to efficiently train a DST model for domains with limited training data. This approach doesn't depend on the ontology of slots and the results show that it can generate slots by prompting the tuned PLM and outperforms the existing state-of-the-art methods under few-shot settings.

The main research objective of this thesis is to investigate the effectiveness of prompt-based methods for DST and to understand the limitations of this approach. Prompt-based methods are adopted for the DST task to answer the following research questions: Q: Can the dialogue belief states be extracted directly from PLM using prompt-based methods? Q: Can the prompt-based methods learn the DST task under low-resource settings without depending on the ontology of domains? Q: How does the prompt-based approach perform overall compared to a baseline model? Q: What are the drawbacks and limitations of prompt-based methods? Q: Can different multi-prompt techniques help the PLM understand the DST task better? Q: What impact do various multi-prompt methods have on the performance of the DST task?

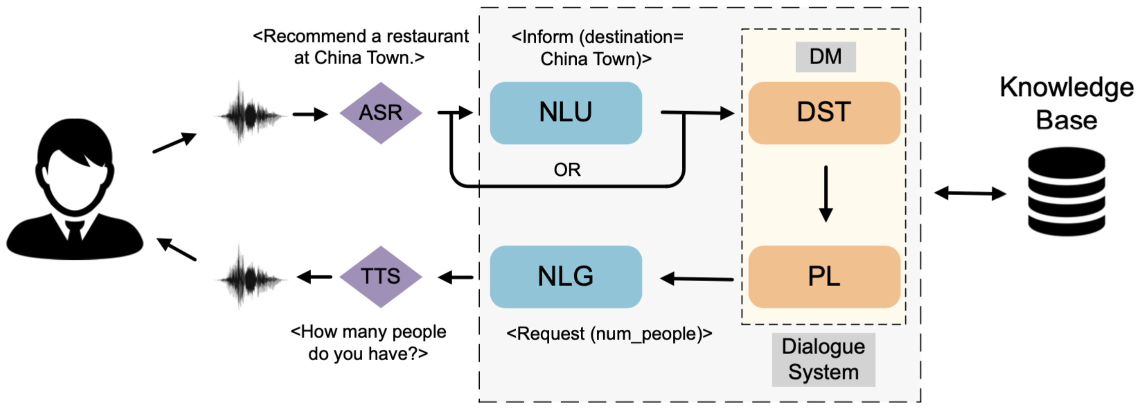
To accomplish the research objectives, the prompt learning framework designed by Yang et al. (2022), which includes a *value-based prompt* and *inverse prompt*, is utilized to generate the belief states by prompting the PLM. Few-shot experiments are performed on different proportions of data to evaluate the prompt-based methods under low-resource settings. A baseline model, which also does not depend on the ontology of dialogue domains, is trained on the DST task to compare with the prompt-based methods. A detailed error analysis is conducted to identify the limitations of prompt-based methods. Further, multi-prompt methods are adopted to help the PLM better understand the DST task.

This section introduced the overview of the thesis topic, motivation, and research objectives. The next section presents the background and related work (section 2) with details on the following topics: dialog state tracking (DST), pre-trained language models (PLMs), the baseline model, prompting methods, and the dataset used. The description of the research methods used in the thesis experiments, including the few-shot experiments of baseline and prompt-based methods, multi-prompt methods, and evaluation metrics, are detailed in section 3. Section 4 provides all the few-shot experimental results of the research methods adopted. Analysis and discussion of results are presented in section 5. Finally, the conclusion (section 6) highlights the summary of the main findings.

2 Background & Related Work

2.1 Dialog State Tracking (DST)

Task-oriented dialog systems, both modular and end-to-end systems, are capable of handling a wide range of tasks (such as ticket booking, restaurant booking, etc.) across various domains. A task-oriented dialogue system has stricter requirements for responses because it needs to accurately understand and process the user's message. Therefore, modular methods were suggested as a way to generate responses in a more controlled manner. The architecture of a typical modular-based task-oriented dialog system is depicted in Figure 1. A typical modular-based system uses a modular pipeline, which has four modules that execute sequentially - Natural Language Understanding (NLU), Dialog State Tracking (DST), Policy Learning (PL), and Natural Language Generation (NLG). The NLU module extracts the semantic values from user messages, together with intent detection and domain classification. The DST module takes the extracted values and fills the slot-value pairs based on the entire dialog history. The Policy Learning (PL) module takes the slot-value pairs and decides the next action to be performed by the dialog system. The NLG module converts the dialog actions received from the PL module into the natural language text, which is usually the system response to the user.



<Recommend a restaurant at China Town.>
ASR
<Inform (destination= China Town)>
NLU
OR
DM
DST
PL
Knowledge Base
TTS
NLG
<How many people do you have?>
<Request (num_people)>
Dialogue System

Figure 1: Modular-based task-oriented dialog system (Ni et al., 2021)

The DST module is essential for enabling the system to comprehend the user's requests by tracking them in the form of slots and values (belief states) at every turn. In recent years, some dialog systems take the user utterances and provide them directly to the DST module. This approach fills the slot-value pairs directly from the raw user message and eliminates the need for NLU module. For example, consider the user message - "*Plan a train trip to Berlin this Friday for two people*" - the DST module is supposed to fill (*slot, value*) pairs as follows: {(*destination, Berlin*), (*day, Friday*), (*people, 2*)}.

A typical task-oriented dialog system can assist users across multiple domains (restaurant, hotel, train, booking). Each dialog domain contains an ontology, which represents the knowledge of the domain and information required for specific tasks. The ontology of a domain consists of a pre-defined set of slots and all the possible values for each slot. Neural-based models were proposed to solve the DST task by multi-class classification, where the model predicts the correct class from multiple values. This approach depends on the ontology of the domains and needs to track a lot of slot-value pairs. Ontology is hard to obtain in real-world scenarios, especially for new domains. The neural-based DST model also needs a lot of training data which is rarely available for new domains.

A dialog state or belief state in DST contains the required information for the system to process the user's request. At each turn of the dialogue, the dialog state can have *informable slots* and *requestable slots.* Informable slots are specified by the user about the preferences and requirements for the system action. For example, in the restaurant domain, the user can ask for a specific type of food or desired price range of the restaurants for table booking. The belief state slots with such information are called *informable slots.* Users can ask the dialog system for an address or phone number of a restaurant, such slots are known as *requestable slots.* This thesis work focuses on the DST module to extract informable slots and their values without depending on the ontology.

2.2 Pre-trained Language Models (PLMs)

Large pre-trained language models are trained on huge amounts of textual data and they achieved state-of-the-art performance in a variety of NLP tasks, such as machine translation, text classification, text generation, and summarization. These PLMs trained on large-scale datasets can encode significant linguistic knowledge into their huge amount of parameters. Pre-trained language models based on transformer architectures (Vaswani et al., 2017), such as BERT (Devlin et al., 2019) and GPT (Radford et al., 2018), have also achieved state-of-the-art performance on many NLP tasks. GPT-2 (Radford et al., 2019) is a transformer-based left-to-right auto-regressive language model trained on large amounts of open web text data. The main training objective of GPT-2 is to predict the next word, given all the previous words. A left-to-right auto-regressive language model predicts the next word given all the previous words or assigns the probability of word sequences. Consider the sequence of words $x = x_1, x_2, \ldots, x_n$, the probability distribution can be written using the chain rule from left to right:

$$P(x) = P\left(x_1\right) \times P\left(x_2 \mid x_1\right) \times \ldots \times P\left(x_n \mid x_1 \cdots x_{n-1}\right)$$

The PLMs trained on large amounts of text can be fine-tuned using task-specific data to solve the downstream tasks efficiently. Previous work by Wu et al. (2020) pre-trained the BERT model with nine different task-oriented dialog datasets and later used it to fine-tune the downstream tasks. This approach improved the performance of downstream tasks over fine-tuning directly on BERT. SOLOIST (Peng et al., 2021) used a similar approach to pre-train the GPT-2 model on two task-oriented dialog corpora and fine-tuned the pre-trained Soloist on the DST task. The pre-trained SOLOIST is the baseline model of this thesis, which uses the publicly available 12-layer GPT-2 (117M) model. The prompt-based methods in this thesis also utilize the pre-trained SOLOIST to fine-tune the prompt-based DST task.

2.3 SOLOIST Model

SOLOIST (Peng et al., 2021) uses the *pre-train, fine-tune* paradigm for building a task-oriented dialog system using an auto-regressive language model GPT-2 (Radford et al., 2019). This dialog system is built in two phases: In the pre-training phase, SOLOIST is initialized with GPT-2 and further trained on two large task-oriented datasets, Schema and Taskmaster. The primary goal at this stage is to learn task completion skills such as *belief prediction* and *response generation*. In the belief predictions task of the pre-training stage, the SOLOIST model takes dialog history as input and generates belief states as a sequence of words. The generated belief state sequences take the form - "*belief:* $slot_1 = value_1; slot_2 = value_2, \ldots$". The pre-training objective for predicting belief states is:

$$\mathcal{L} = \log P(b \mid s) = \sum_{t=1}^{T_b} \log P\left(b_t \mid b_{<t}, s\right)$$

where T_b is the generated belief states sequence length, $b_{<t}$ indicates all tokens before t, s is the dialog history up to the current turn. Overall, $log(b \mid s)$ represents the probability of generating the belief states sequence given the dialog history.

In the fine-tuning stage, the pre-trained SOLOIST model can be used to solve new tasks by just using a small amount of task-specific dialogs. The belief predictions task of the SOLOIST model can be fine-tuned on new task-oriented dialog datasets for solving the DST tasks. At inference time, the fine-tuned SOLOIST uses top-K (Fan et al., 2018) and nucleus (Holtzman et al., 2020) sampling for generating belief states as a sequence of words. In top-K sampling, the K most likely next words are filtered out and the probability is redistributed among only those K next words. In nucleus sampling (also known as *top-p* sampling), only the words that exceed the probability threshold p are chosen. This approach of generating belief states does not depend on the ontology of slots and values. The fine-tuning of belief predictions task on pre-trained SOLOIST is the baseline model for this thesis. The same pre-trained SOLOIST is used to fine-tune the prompt-based methods.

2.4 Prompt Learning

Prompt-based learning is a new way of using pre-trained language models more efficiently for solving language tasks. It involves changing the task using textual prompts, and the language model generates the desired output directly from the prompts. The main idea behind this approach is to efficiently use the generation capabilities of PLMs. Table 1 introduces some terminology, notations, and an emotion classification example. The original input x is modified using the *prompting function* which generates the *prompt* x'. The *prompt function* or *prompt template* typically contains text and two slots: the input slot $[X]$ for filling the input x and the answer slot $[Z]$ for generating the answer z. The prompt x' is given to the PLM to directly generate the answer z. For tasks such as emotion classification, another step of answer mapping is required to get to the final output y from answer z. For example, multiple emotion-related words (such as *happy, joyful, delighted, pleased*) can belong to the same output class (e.g. "*joy*"). In this case, if the PLM generates an answer "*happy*", it is mapped to the output class "*joy*". For some tasks involving text generation, answer mapping is usually not required, the generated answer z becomes the output y.

Name	**Notation**	**Example**
Input	x	I missed the bus today.
Output	y	sad
Prompt Function	$f_{prompt}(x)$	$[X]$ I felt so $[Z]$
Prompt	x'	I missed the bus today. I felt so $[Z]$
Answered Prompt	$f_{fill}(x', z)$	I missed the bus today. I felt so sad
Answer	z	*happy*, *sad*, *scared*

Table 1: Terminology and notations of prompting methods

Consider the emotion classification example from table 1, in order to recognize the emotion in the text, where input $x =$ "I missed the bus today.", given the prompt function "[X] I felt so [Z]". $[X]$ is filled with input x, then the prompt x' would become "I missed the bus today. I felt so [Z]" and the PLM is supposed to fill the slot $[Z]$ with the emotion word "sad".

Prompt types There are two main varieties of prompts: *prefix prompts* and *cloze prompts*. In prefix prompts, the entire prompt text comes before the slot $[Z]$. For example, consider the prompt - "I like this movie. The movie is $[Z]$", the slot $[Z]$ is at the end. In cloze prompts, the slot to be filled $[Z]$ appears in the middle or beginning of the prompt text. For example, consider the prompt - "Berlin is the capital of Germany. $[Z]$ is the capital of Japan", the slot $[Z]$ is in the middle of the prompt text. For tasks that are solved using a left-to-right auto-regressive language model, using prefix prompts is more helpful. This is because prefix prompts are well-suited to the left-to-right nature of the language model. There are multiple ways of creating prompts: *manual prompts, discrete prompts, and continuous prompts*. For *manual prompts*, the templates are hand-crafted by humans based on the intuition of the task. These manual prompts generally contain a few natural language phrases and are usually task-specific. This approach can be time-taking and often requires a lot of experimenting. For *discrete prompts*, the templates are searched by using automated methods such as prompt mining, gradient-based search, and generation from LM. These templates are also in the form of natural language phrases. This approach might require a large amount of training data to find prompts. For *continuous prompts*, the templates are directly expressed in the embedding space of the language model. These prompts have their own parameters and can be tuned based on the training data of the task.

Training strategy Prompting methods can be used without any training to the PLM for the downstream tasks. This can be done by taking a suitable pre-trained LM and applying the prompts directly to the inputs of the task. This approach is traditionally known as *zero-shot learning*. However, this zero-shot approach has a risk of bias, as the PLM is not fine-tuned for the task and is also less effective on tasks that are different from what the PLM is trained on. *Few-shot learning* is another approach where only a small amount of task-specific training samples are used to train the language model. Prompting methods are particularly useful in this approach when there is not enough task-specific training data to fully train the model. There are different training methods to fine-tune the prompts and the LM: fix the LM parameters and fine-tune the prompts, fix the prompts and update the LM parameters, and fine-tune both LM parameters and prompts. From these training strategies, *fixed-prompt LM tuning* is a way to improve the PLM by fine-tuning it with the prompts. This is similar to the standard fine-tuning paradigm, by applying fixed prompts to the training inputs and fine-tuning them on PLM. This approach helps the PLM understand the downstream task and can potentially lead to improvements under few-shot settings.

2.5 Prompt-based DST

In the previous work TOD-BERT (Wu et al., 2020), the BERT language model is pre-trained on nine different task-oriented dialogue datasets. The pre-trained TOD-BERT is fine-tuned on multiple task-oriented dialogue tasks (intent recognition, dialog state tracking) to evaluate them under few-shot settings. The DST task in this work is a multi-class classification problem, by predicting the slots and values from the pre-defined ontology. Soloist (Peng et al., 2021) also used a similar approach by pre-training the GPT-2 language model with two different dialogue datasets. The downstream DST task of the Soloist model does not depend on the ontology of domains and directly generates belief states as word sequences (described in section 2.3). Both TOD-BERT and Soloist models do not use prompt-based methods and perform poorly under extremely low-resource settings. The pre-trained Soloist model is adopted as a baseline in this thesis to explore the prompt-based methods for DST.

Previous work by Lee et al. (2021) used prompting methods on a PLM to solve the DST task. This work introduced schema-driven prompting (*slot-based prompt*) that takes domain names, slots, and natural language descriptions of slots to fill in the prompts and generates the corresponding values. This method relies on the complete ontology of the domains and their slot descriptions. This method also requires a lot of training data to fine-tune the PLM. Yang et al. (2022) proposed a new prompt-learning framework for DST that uses values in prompts (*value-based prompt*) and generates the slots directly from the PLM. This value-based prompt approach does not depend on the ontology of the domains. This work designed a *value-based prompt* and *inverse prompt* to help the PLM solve the DST task during the fine-tuning stage. Figure 2 shows an overview of value-based prompt and inverse prompt for DST.

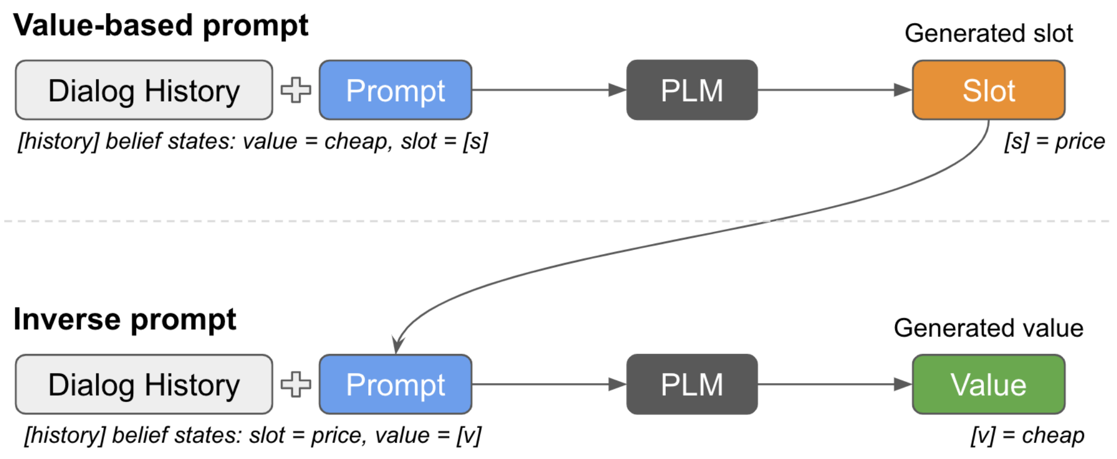


Figure 2: Overview of value-based prompt and inverse prompt mechanism.

First, the belief state slots are generated using value-based prompts. The generated slots from the value-based prompt are given to the inverse prompt to generate back the values. The inverse prompt function can be considered as an auxiliary task that helps the PLM to understand the DST task. The loss from the value-based prompt and inverse prompt are combined during the training phase. At inference time, the value-based prompt is directly used to generate the slots without depending on the ontology. Yang et al. (2022) showed the prompt-learning framework can efficiently learn the DST task even under extremely low-resource settings. These prompt-based methods are further explored in this thesis. The experimental methods for prompt-based DST are detailed in section 3.3.

2.6 MultiWOZ Dataset

MultiWOZ (Budzianowski et al., 2018) is a widely used dialogue dataset for benchmarking task-oriented dialog systems. MultiWOZ is a collection of human-human written conversations that are centered around a wide range of topics, such as hotel booking, restaurant reservation, attraction recommendations, and booking a train. It contains over 10K dialogues across 8 domains, with multiple turns for each dialogue. Each dialogue contains multiple user and system utterances and belief states with slot-value pairs for each turn. Eric et al. (2019) released MultiWOZ 2.1 after fixing the noisy dialog state annotations and utterances that negatively impact the performance of DST models. In this thesis, MultiWOZ 2.1 dataset is used to evaluate the baseline and prompt-based methods on the DST task.

3 Methods

This section describes the research methods and experimental setup of the work conducted in this thesis. This thesis work can be divided into the following tasks: SOLOIST baseline implementation for few-shot DST, prompt-based methods for few-shot DST, evaluation and analysis of belief state predictions, and multi-prompt methods for DST.

3.1 Dataset

The baseline and prompt-based methods are benchmarked on MultiWOZ 2.1 (Eric et al., 2019) dataset. The MultiWOZ dataset contains 8438/1000/1000 single-domain and multi-domain dialogues for training/validation/testing respectively. Each dialogue can have multiple turns and each turn can include multiple *(slot, value)* pairs. Dialogues from only five domains (*Restaurant, Hotel, Attraction, Taxi, Train*) and one sub-domain (*Booking*) are used in the experiments, as the other two domains (*Hospital, Police*) only appear in the training set. To observe the performance under few-shot settings, dialogues are randomly sampled for each domain and six different data splits are created. Each data split contains dialogues with all five domains and the dialogues are evenly distributed for each domain. Only single-domain dialogues including booking sub-domain are picked for creating the data splits. Validation and test sets are not sampled after domain filtering. Table 2 provides data statistics and the summary of data splits used in few-shot experiments.

Data Splits	**# Dialogues**	**# Total Turns**	**# (slot, value)**
5-dpd	25	100	294
10-dpd	50	234	758
50-dpd	250	1114	3535
100-dpd	500	2292	7408
125-dpd	625	2831	9053
250-dpd	1125	5187	17214
valid	190	900	3106
test	193	894	3411

Table 2: Data statistics and data split summary for few-shot experiments. The term *dpd* means *"dialogues per domain"*. Each split contains dialogues for all five domains. In data split *250-dpd*, the domain "*Attraction*" contains only 125 dialogues.

In the MultiWOZ 2.1 dataset, 16 dialog slots are used to understand the user requirements. For the prompt-based experiments, these slots are converted to look like natural language words for fine-tuning the slot generation process. Table 3 lists the slots from all five domains and *booking* sub-domain.

Slots
area, arrive, day, departure, destination, food, internet, leave, name, parking, people, price, stars, stay, time, type

Table 3: Slots from MultiWOZ 2.1 dataset used in prompt-based experiments

3.2 SOLOIST Baseline

Soloist (Peng et al., 2021) is the baseline model for the prompt-based methods. Soloist is initialized with the 12-layer GPT-2 (Radford et al., 2019) and further trained on two task-oriented dialog corpora (Schema and Taskmaster). The task-grounded pre-training helps the Soloist model to solve two dialog-related tasks: *belief state prediction* and *response generation*. In the belief state predictions task, the model takes dialog history as input and generates the belief states as a sequence of words. In this thesis, for the baseline implementation, the pre-trained Soloist is fine-tuned on MultiWOZ 2.1 data splits to perform the belief predictions task. During inference time, the fine-tuned Soloist baseline doesn't need the pre-defined set of slots and their possible values, and it uses top-K (Fan et al., 2018) and top-p or nucleus (Holtzman et al., 2020) sampling for generating the belief states. In the prompt-based DST task, the same pre-trained Soloist model is fine-tuned for prompt-based slot generation.

3.3 Prompt-based few-shot DST

This task aims to apply prompt-based methods proposed by (Yang et al., 2022) and reproduce the results. This task utilizes the *value-based prompt* and *inverse prompt* for fine-tuning the pre-trained Soloist, which can generate the belief state slots directly at inference time. The prompt-based methods are evaluated on the same data splits (Table 2) of the MultiWOZ 2.1 dataset.

Value-based prompt An intuitive idea for generating (*slot, value*) pairs is to use slots in prompts and generate the corresponding values (Lee et al., 2021). For example, given the utterance - "*Plan a trip to Berlin*" and slot (*destination*), the prompt to the PLM could become "*Plan a trip to Berlin. destination = [z]*" and the PLM is expected to generate *[z]* as "*Berlin*". However, this approach relies on the ontology of the slots, and the fixed set of slots can change in real-world applications. Yang et al. (2022) proposed *value-based prompt* that uses values in the prompts and generates corresponding slots. This method doesn't require any pre-defined set of slots and can generate slots directly from the PLM. Consider this prompt template: "*belief states: value = [v], slot = [s]*", the prompt function f can be of form $f(v)$ = *[dialog history] belief states: value = [v], slot = [s]*, given the value candidate v = "*Berlin*", the PLM can generate *slot [s] = "destination"*. The overall training objective of value-based prompt generation is minimizing the negative log-likelihood of slots in the training dataset D:

$$\mathcal{L} = -\sum_{t}^{|D|} \log P\left(s_t \mid c_t, f\left(v_t\right)\right) \tag{1}$$

where $P\left(s_t \mid c_t, f\left(v_t\right)\right)$ is the probability of generating slot s_t given dialog history c_t and prompt-function f is filled with value v_t for each turn t. The loss $\mathcal{L}$ from this step is combined with the loss from inverse prompt (next step) in order to compute the final loss. During training, the annotated values from the dataset are utilized to fill in the value-based prompts.

Inverse Prompt The *inverse prompt* mechanism (Yang et al., 2022) aims to generate the values by filling the prompts with generated slots. After generating slot s using the value-based prompt, this generated slot is presented to the inverse prompt function I. The inverse prompt aims to generate the value v' which is supposed to be close to the original value v. The prompt template for inverse prompt function can be of form I = "*belief states: slot = [s], value = [v]*". *[s]* is filled with the generated slot from value-based prompt, *[v]* is expected to be generated by the PLM. This inverse prompt can be considered as an auxiliary task for the value-based prompt, which can improve performance by helping the PLM understand the task, especially under low-resource scenarios. The loss function $\tilde{\mathcal{L}}$ for the inverse prompt mechanism:

$$\tilde{\mathcal{L}} = -\sum_{t}^{|D|} \log P\left(v_t' \mid c_t, I\left(s_t\right)\right) \tag{2}$$

where $P\left(v_t' \mid c_t, I\left(s_t\right)\right)$ is the probability of generating the value v_t' by filling in the inverse prompt $I\left(s_t\right)$ with the generated slot s_t.

The final loss $\mathcal{L}^*$ is computed by combining loss from value-based prompt $\mathcal{L}$ and the inverse prompt loss $\tilde{\mathcal{L}}$:

$$\mathcal{L}^* = \mathcal{L} + w * \tilde{\mathcal{L}} \tag{3}$$

where w is a decimal value (0,1) which is used to adjust the influence of inverse prompt.

Training For training the prompt-based methods, the pre-trained Soloist (GPT-2 117M) is fine-tuned on value-based prompt and inverse prompt. All the MultiWOZ 2.1 data splits (Table 2) are used in the fine-tuning process in order to evaluate the performance under few-shot settings. The training strategy fixed-prompt LM tuning is adapted for tuning the prompt-based methods, where the fixed discrete prompts are used to fine-tune the parameters of the LM. Table 4 shows the prompt templates used in the fine-tuning process. The prompts are appended to the dialog history before providing them as input to the PLM and probabilistically generate the missing slots. The inverse prompt is only used during the training phase. Experiments are also performed to evaluate the influence of inverse prompt by omitting it during the training process.

Type	**Prompt templates**
value-based prompt	belief states: value = [v], slot = [s]
inverse prompt	belief states: slot = [s], value = [v]

Table 4: Prompt templates used during the training phase.

Testing (Slot Generation) During the testing phase, only value-based prompts are used to generate the slots. The filled prompt together with the dialog history is given as input to the PLM, and the next word with the highest probability is the generated slot. While testing, the value candidates are not known. A set of rules are applied to extract the candidate values directly from the user utterances. This sort of value extraction from utterances is previously explored by Min et al. (2020).

Value Extraction Value candidates are extracted directly from the dialog history and are provided to the value-based prompts for generating slots at inference time. Stanford CoreNLP Stanza (Qi et al., 2020) tool is used to first extract POS tags and named entities, a set of rules are then applied to extract the candidate values.

- Adjectives (JJ) and Adverbs (RB) are considered as possible values
  - E.g., *expensive*, *moderate*, *important*
- Consider the previous negator 'not'
  - E.g., *not expensive*, *not important (= dont care)*
- Consider all named entities (name of place, time, date/day, numbers)
  - E.g., *cambridge*, *friday*, *08:30*
- Custom set of Regex NER rules are applied for recognizing named entities
  - E.g., *restaurant names*, *attraction names*
- Stop words and repeated candidate values are filtered out

Prompt Decomposition For utterances where multiple *(slot, value)* pairs are expected to be predicted, directly using a single prompt for generating multiple slots is challenging. Prompt decomposition is a multi-prompt method that breaks down the prompt into sub-prompts and generates the slots separately for each sub-prompt. For each extracted value from the utterances, a value-based prompt is constructed and the corresponding slot is generated. This sort of prompt decomposition has been explored by Cui et al. (2021) for the named entity recognition (NER) task. This approach is applied in both the training and testing phases.

Utterance:	Book a flight to Berlin on friday at 08:30.
Prompt 1:	belief states: value = *Stuttgart*, slot = [s]
Prompt 2:	belief states: value = *friday*, slot = [s]
Prompt 3:	belief states: value = *08:30*, slot = [s]

Table 5: Sub-prompts for an utterance with multiple values.

3.4 Multi-prompt methods for DST

The *value-based* prompt described in the previous section utilizes a *single* prompt for making predictions. However, a significant body of research has demonstrated that the use of multiple prompts can further improve the efficacy of prompting methods (Liu et al., 2021). There are different ways to extend the single prompt learning to use multiple prompts. This task explores two more multi-prompt learning methods: *Prompt Ensembling* and *Prompt Augmentation*. Experiments are performed on all the data splits of MultiWOZ 2.1 dataset. This task aims to answer the following questions - Q1: Can different *multi-prompt* techniques together help the PLM better understand the DST task? Q2: Can the use of multiple discrete prompts improve the performance of prompt-based model?

Prompt Ensembling This method uses multiple *value-based* prompts during the training and inference time. This idea can leverage the complementary advantages of different prompts and stabilize the performance on the downstream task. Yang et al. (2022) applied prompt ensembling to the value-based prompt by training a separate model for each prompt. Another way is to train a single model with multiple prompts as it is much faster and more memory efficient than having to train a separate model for each prompt (Schick and Schütze, 2021). Prompt ensembling is applied only to value-based prompts, and the inverse prompt uses a single prompt. In this task, four hand-crafted prompt templates are chosen for value-based prompts and trained on a single model. The probability of generated slot s_t over multiple prompt functions is calculated by weighted averaging the probability of each prompt:

$$P\left(s_t \mid c_t\right)=\sum_{k}^{|K|} \alpha_k * P\left(s_t \mid c_t, f_k\left(v_t\right)\right) \tag{4}$$

where $|K|$ represents the number of prompt functions, f_k is the k-th prompt function, α_k is the weight of prompt k. During inference, a simple majority voting is used to pick the generated slot from multiple prompts. When there's no simple majority in the generated slots, the slot with the highest probability is picked. Table 6 lists all the prompt templates used in prompt ensembling.

	Prompt ensemble templates
f_1	belief states: [v] = [s]
f_2	[v] is the value of [s]
f_3	[v] is of slot type [s]
f_4	belief states: value = [v], slot = [s]

Table 6: Prompt templates used for prompt ensemble.

Prompt Augmentation *Prompt Augmentation*, sometimes also called *demonstration learning* (Gao et al., 2021), provides a few additional *answered prompts* that can demonstrate to the PLM, how the actual value-based prompt can be answered. These demonstrations take advantage of the language models' ability to learn repetitive patterns. The sample selection of answered prompts is hand-crafted from the training data. At inference time, the answered prompts are appended to the input before asking the PLM to generate the slot. Table 7 below provides an example of demonstration learning.

Demonstration learning	
Book a cheap flight to Frankfurt.	*Frankfurt* is of slot *destination*
Plan a train trip to Berlin.	*Berlin* is of slot *destination*
Book a taxi to the University.	*University* is of slot *destination*
Book a train to Stuttgart.	*Stuttgart* is of slot [s]

Table 7: Example prompt augmentation with answered prompts

3.5 Evaluation Metrics

The standard evaluation metric joint goal accuracy (JGA) is adopted to evaluate the belief state predictions of baseline and prompt-based methods. This metric compares all the predicted belief states to the ground-truth states at each turn. The prediction is correct only if all the predicted belief states match the ground-truth states. Both slots and values must exactly match for the belief state prediction to be correct. The rule-based methods used in value extraction can lead to many false positives in the value candidates. In order to exclude the influence of wrongly extracted values, Yang et al. (2022) proposed JGA*, the joint goal accuracy is computed only for the belief states where the values are extracted correctly. These evaluation metrics answer the following questions: Q1: How do the prompt-based methods perform overall compared to the Soloist baseline? Q2: Can the prompt-based methods perform better under low-resource settings? Q3: For prompt-based methods, does JGA* metric hold a better score than JGA? Q4: Can multi-prompt techniques together perform better than a single-prompt?

Analysis The belief state predictions from the SOLOIST baseline and prompt-based methods are analyzed to identify the potential improvements and drawbacks. A detailed qualitative analysis is performed on the wrong belief state predictions. Additionally, error analysis is also performed on the rule-based value extraction methods to identify the impact on the slot generation process.

3.6 Implementation Details

For the Soloist baseline, the existing implementation by Peng et al. (2021) is adapted to the few-shot experiments conducted in this thesis. There's no publicly available implementation of the prompt-based methods for DST. Huggingface Transformers (Wolf et al., 2020) library is used to implement the prompt-based DST methods from scratch. Adam (Kingma and Ba, 2015) optimization algorithm is used during the fine-tuning process of both baseline and prompt-based methods. The rule-based value extraction methods are implemented using Stanford CoreNLP client stanza (Qi et al., 2020). The inverse prompt weight w in Eq. 3 is set to 0.1. The prompt weight α_k in Eq. 4 is set to the same value (1/4) for all the prompts used in prompt ensembling.

4 Results

This section presents the experimental results evaluated on all the methods described in the previous sections. Few-shot experiments are performed on all the data splits (see Table 2) for every method. The baseline Soloist model is evaluated only on the JGA metric. For the prompt-based methods, in addition to the JGA metric, JGA* is also computed.

4.1 SOLOIST Baseline

Table 8 shows the results of the baseline model under few-shot experiments. Experimental results show the baseline model performed poorly and struggled to generate belief states under low-resource settings (*5-dpd*, *10-dpd*, *50-dpd*). Under low-resource data splits, the limited size of data samples made it challenging for the baseline to generate unseen belief states. The results also show that more data may be necessary as the model achieves better results on the data splits with a higher number of data samples (*125-dpd*, *250-dpd*).

Data split (# dialogs)	**JGA**
5-dpd (25)	9.06
10-dpd (50)	14.20
50-dpd (250)	28.64
100-dpd (500)	33.11
125-dpd (625)	35.79
250-dpd (1125)	**40.38**

Table 8: Few-shot experimental results of the SOLOIST baseline model. The term "*dpd*" stands for "*dialogues per domain*".

4.2 Prompt-based methods

Table 9 shows the results of the prompt-based model under few-shot experiments. Only a single value-based prompt is used in these experiments. Experimental results show the prompt-based model significantly outperforms the baseline model in all data splits. For low-resource data splits like *5-dpd*, *10-dpd*, and *50-dpd*, the prompt-based model shows a substantial improvement over the baseline, achieving an increase in the JGA metric by *21*, *28*, and *18* points respectively.

Data split (# dialogs)	**JGA**	**JGA***
5-dpd (25)	30.66	71.04
10-dpd (50)	42.65	86.43
50-dpd (250)	47.06	91.63
100-dpd (500)	**47.74**	**92.31**
125-dpd (625)	46.49	91.86
250-dpd (1125)	47.06	92.08

Table 9: Few-shot experimental results from the prompt-based model. Only a single *value-based prompt* is used. The term "*dpd*" stands for "*dialogues per domain*".

The prompt-based results also show that by increasing the number of data samples in experiments, the model only achieved minor performance improvements. For example, the prompt-based methods perform nearly identical on the data splits 50-dpd and 250-dpd. This suggests the prompt-based approach understands the DST task better under low-resource scenarios. The higher values of the JGA* metric across all data splits indicate the potential drawbacks of the rule-based value extraction methods.

4.3 Multi-prompt methods

4.3.1 Prompt Ensembling results

Table 10 shows the results of prompt ensembling under few-shot settings. The results from the prompt ensemble show a slight improvement over a single value-based prompt. Contrary to expectations, the prompt ensemble model did not show significant performance improvement on the JGA metric. The results also show that the performance of the prompt ensemble model is similar when trained on large data splits, i.e. *50-dpd*, *100-dpd*, *125-dpd*, *250-dpd*.

Data split (# dialogs)	**JGA**	**JGA***
5-dpd (25)	30.09	69.23
10-dpd (50)	42.84	86.99
50-dpd (250)	47.62	91.74
100-dpd (500)	**48.08**	**92.87**
125-dpd (625)	46.96	92.08
250-dpd (1125)	**48.08**	**92.87**

Table 10: Few-shot experimental results from prompt ensembling (multi-prompt method). Four *value-based prompts* are used at training and inference time. The term "*dpd*" stands for "*dialogues per domain*".

4.3.2 Prompt Augmentation results

Table 11 shows the results of prompt augmentation under few-shot settings.

Data split (# dialogs)	**JGA**	**JGA***
5-dpd (25)	27.8	68.1
10-dpd (50)	38.91	74.43
50-dpd (250)	39.52	82.81
100-dpd (500)	**42.42**	**83.71**
125-dpd (625)	40.16	82.92
250-dpd (1125)	41.52	85.07

Table 11: Experimental results from demonstration learning (multi-prompt method). The term "*dpd*" stands for "*dialogues per domain*".

Prompt augmentation (also called *demonstration learning*) provides additional context to the language models in the form of "*answered prompts*" at inference time. The hand-crafted answered prompts are supposed to help the language model understand the DST task better and generate accurate responses. Table 11 shows the experimental results from the prompt augmentation method. Results show that the demonstration learning struggled to generate the belief states accurately. The performance is inadequate across all data splits when compared to other prompt-based methods. Only a limited number of answered prompts can be provided to the GPT-2 LM due to the max input sequence length of 1024, which led to bias during the slot generation process.

Overall, the multi-prompt methods (prompt ensembling and prompt augmentation) struggled to improve the performance of the DST task. However, the prompt ensembling approach with multiple value-based prompts showed minor improvements over a single value-based prompt.

5 Analysis

5.1 Error analysis of baseline model

	Wrong belief state predictions
Dialog History	user: we need to find a guesthouse of moderate price. system: do you have any special area you would like to stay? or possibly a star request for the guesthouse? user: i would like it to have a 3 star rating.
True belief states	*(type, guesthouse) (pricerange, moderate) (stars, 3)*
Generated states	*(parking, yes) (stars, 3)*
Dialog History	user: i need an expensive place to eat in the west. system: is there a specific type of food you would like? user: yes, i would like eat indian food.
True belief states	*(area, west) (food, indian) (pricerange, expensive)*
Generated states	*(area, west) (food, indian) (pricerange, cheap) (area, east)*

Table 12: Examples of a wrongly generated belief states by the baseline model.

The belief predictions task of the SOLOIST baseline utilizes *top-k* and *top-p* sampling in order to generate the *(slot, value)* pairs. As the baseline model uses open-ended generation, it is susceptible to generating random slot-value pairs that are not relevant. The baseline performance was also affected by the repeated slot generations and in some cases incorrect values. Table 12 shows examples of some of the errors made by the baseline model. In the first example, the baseline system missed two true states and generated a totally incorrect belief state. For the second example, the slot *area* is repeated with a different value and the value for the slot *pricerange* is incorrectly generated.

5.2 Analysis of prompt-based methods

5.2.1 Value-based Prompt

Dialog History	user: I need to be picked up from pizza hut city centre after 04:30
True belief states	*(departure, pizza hut city centre) (leave, 04:30)*
Generated states	*(destination, pizza hut city centre) (arrive, 04:30)*
Dialog History	user: I need a taxi to arrive by 16:45 to take me to the parkside police station.
True belief states	*(destination, parkside police station) (leave, 16:45)*
Generated states	*(destination, parkside police station) (arrive, 16:45)*

Table 13: Incorrect belief states generated by value-based prompt.

The value-based prompt trained on low-resource data splits (i.e., *5-dpd*, *10-dpd*) struggled to distinguish between the slots like *departure* vs *destination* and *leave* vs *arrive*. In many instances, it wrongly generated the slot *destination* instead of *departure* and slot *arrive* instead of *leave*. Table 13 shows some example outputs where the slots are generated incorrectly. In both examples, the slot arrive is incorrectly generated. These incorrect slot generations are due to the limited training available for these examples. Overall, the prompt-based methods perform significantly better than the baseline even under low-resource settings, due to the constrained generation of slots using value-based prompts.

5.2.2 Impact of Inverse Prompt

The inverse prompt mechanism can be considered as an auxiliary task that complements the value-based prompt and helps generate the slots more accurately. Experiments are performed to analyze the impact of the inverse prompt by omitting it while training the value-based prompt.

Data split	**w/o inverse prompt**		**with inverse prompt**	
(# dialogs)	**JGA**	**JGA***	**JGA**	**JGA***
5-dpd (25)	26.81	64.25	30.66	71.04
10-dpd (50)	41.1	82.35	42.65	86.43
50-dpd (250)	45.7	90.7	47.06	91.63
100-dpd (500)	**47.74**	**91.86**	**47.74**	**92.31**
125-dpd (625)	45.02	90.61	46.49	91.86
250-dpd (1125)	46.15	91.4	47.06	92.08

Table 14: Experimental results showing value-based prompt performance with and without inverse prompt mechanism while training. The term "*dpd*" stands for "*dialogues per domain*", "*w/o*" means "*without*".

The results from table 14 show the inverse prompt mechanism helped improve the performance of the prompt-based model, especially under low-resource data splits (i.e., *5-dpd*, *10-dpd*). For the data split *5-dpd*, where the training data is very limited, the inverse prompt brings noticeable improvements by achieving a *5%* increase in performance on JGA and a *7%* increase on JGA* metric. For the data splits with a higher number of data samples (i.e., *100-dpd*, *125-dpd*, *250-dpd*), only minor improvements can be observed in the performance when the inverse prompt is included in the training. The experimental results conclude that the inverse prompt mechanism has a noticeable impact on the prompt-based model under extremely low-resource settings.

5.2.3 Repeated values in Belief States

In the prompt-based methods, the value-based prompt takes the candidate values and generates the corresponding slots. The belief states can have repeated values in the (slot, value) pairs. In other words, the user requirements may lead to having repeated values in the belief state (slot, value) pairs.

History	user: hi, can you help me find a 3 star place to stay? system: Is there a particular area or price range you prefer? user: how about a place in centre of town that is of type hotel. system: how long would you like to stay, and how many people? user: I'll arrive on saturday and stay for 3 nights with 3 people.
True states	*(area, centre) (stars, 3) (type, hotel) (day, saturday) (stay, 3) (people, 3)*

Table 15: An example instance with repeated values in the (slot, value) pairs

The data instance listed in table 15 contains multiple (slot, value) pairs. For the belief slots *stars*, *stay*, and *people*, the value is the same. The value-based prompt can only generate one slot with the repeated value 3. This is a main drawback of the value-based prompt under the existing belief state annotation system.

5.2.4 Error Analysis of Value Extraction

History	user: I want a place to stay that has free wifi and free parking. system: do you have a preference for area or price range? user: I don't have a preference. I want a hotel not guesthouse.
True states	*(area, <u>dont care</u>) (internet, <u>yes</u>) (parking, <u>yes</u>) (price, <u>dont care</u>) (type, hotel)*
Extracted values	*free, hotel*
History	user: I need a guesthouse with free wifi please. system: which area would you prefer? user: I also need free parking, and I prefer a 4 star place.
True states	*(internet, <u>yes</u>) (parking, <u>yes</u>) (stars, 4) (type, guesthouse)*
Extracted values	*free, guesthouse, 4*

Table 16: Example data instances where values cannot be extracted (underlined).

At inference time, the value-based prompt requires the belief state values in order to generate slots. The value extraction methods apply a set of rules on POS tags and named entities to extract value candidates directly from utterances. The rule-based extraction has an accuracy of *79%* over all the values and a turn-level accuracy of *49%* on the test split. Table 16 highlights instances where the values cannot be extracted using rule-based methods. In the first example, the value "*dont care*" does not appear in the utterances and cannot be extracted from POS tags. When the user requirement is *free* wifi or *free* parking, the existing annotation system for belief states considers it as the value "*yes*". The rule-based methods adopted for value extraction can only extract the value "*free*" from the utterances. The values "*dont care*" and "*yes*" also occur twice in the examples shown in table 16, as described in the previous section (sec 5.2.4) the value-based prompt cannot handle repeated values for slot generation.

History	user: I kind of need some help finding a nice hotel in the north part of town.
True states	*(area, north) (price, expensive) (type, hotel)*
Extracted values	<u>*kind*</u>, <u>*nice*</u>, *hotel*, *north*

Table 17: Example instance where values are extracted incorrectly (underlined).

After extracting POS tags using the CoreNLP client, all the *adjectives* and *adverbs* from the utterances are considered as candidate values. This approach can lead to false positives in value candidates. Table 17 shows a data instance where some values are extracted incorrectly. The existing annotation system associates the user utterance "a nice hotel" with the value "*expensive*" for slot price, this cannot be achieved under the current rule-based methods. The value "*kind*" is also extracted incorrectly due to considering all the *adverbs* as possible values. The rule-based value extraction methods used in this thesis have limitations, which led to the performance degradation of prompt-based DST.

6 Conclusion

This work explored the use of prompt-based methods for dialog state tracking (DST) in task-oriented dialogue systems. The prompt-based methods, which include value-based prompt and inverse prompt, learned the DST task efficiently under low-resource few-shot settings without relying on the pre-defined set of slots and values. Experiments show that the prompt-based methods significantly outperformed the baseline SOLOIST model under low-resource settings. Analysis of generated belief states shows the prompt-based approach has some limitations. Additionally, multi-prompt methods such as prompt ensembling and prompt augmentation are applied to the DST task. Results show that the prompt ensemble model achieved minor improvements, and the performance of prompt augmentation is limited due to the bias in answered prompts. Error analysis of value extraction highlights the limitations of the rule-based methods. Further research is necessary to overcome the limitations of prompt-based methods and value extraction methods.

References

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

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

Leyang Cui, Yu Wu, Jian Liu, Sen Yang, and Yue Zhang. Template-based named entity recognition using BART. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 1835–1845, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.findings-acl.161. URL https://aclanthology.org/2021.findings-acl.161.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019.

Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://aclanthology.org/N19-1423.

Mihail Eric, Rahul Goel, Shachi Paul, Abhishek Sethi, Sanchit Agarwal, Shuyang Gao, and Dilek Hakkani-Tür. Multiwoz 2.1: Multi-domain dialogue state corrections and state tracking baselines. *CoRR*, abs/1907.01669, 2019. URL http://arxiv.org/abs/1907.01669.

Angela Fan, Mike Lewis, and Yann Dauphin. Hierarchical neural story generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 889–898, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-1082. URL https://aclanthology.org/P18-1082.

Tianyu Gao, Adam Fisch, and Danqi Chen. Making pre-trained language models better few-shot learners. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3816–3830, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.295. URL https://aclanthology.org/2021.acl-long.295.

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

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1412.6980.

Chia-Hsuan Lee, Hao Cheng, and Mari Ostendorf. Dialogue state tracking with a language model using schema-driven prompting. *CoRR*, abs/2109.07506, 2021. URL https://arxiv.org/abs/2109.07506.

Shiyang Li, Semih Yavuz, Kazuma Hashimoto, Jia Li, Tong Niu, Nazneen Rajani, Xifeng Yan, Yingbo Zhou, and Caiming Xiong. Coco: Controllable counterfactuals for evaluating dialogue state trackers. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=eom0IUrF__F.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *CoRR*, abs/2107.13586, 2021. URL https://arxiv.org/abs/2107.13586.

Andrea Madotto, Zhaojiang Lin, Genta Indra Winata, and Pascale Fung. Few-shot bot: Prompt-based learning for dialogue systems. *CoRR*, abs/2110.08118, 2021. URL https://arxiv.org/abs/2110.08118.

Qingkai Min, Libo Qin, Zhiyang Teng, Xiao Liu, and Yue Zhang. Dialogue state induction using neural latent variable models. In Christian Bessiere, editor, *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pages 3845–3852. International Joint Conferences on Artificial Intelligence Organization, 7 2020. doi: 10.24963/ijcai.2020/532. URL https://doi.org/10.24963/ijcai.2020/532. Main track.

Jinjie Ni, Tom Young, Vlad Pandelea, Fuzhao Xue, Vinay Adiga, and Erik Cambria. Recent advances in deep learning based dialogue systems: A systematic survey. *CoRR*, abs/2105.04387, 2021. URL https://arxiv.org/abs/2105.04387.

Baolin Peng, Chunyuan Li, Jinchao Li, Shahin Shayandeh, Lars Liden, and Jianfeng Gao. SOLOIST: building task bots at scale with transfer learning and machine teaching. *Transactions of the Association for Computational Linguistics*, 9:807–824, 2021. doi: 10.1162/tacl_a_00399. URL https://aclanthology.org/2021.tacl-1.49.

Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning. Stanza: A python natural language processing toolkit for many human languages. In *Proceedings of the 58th Annual Meeting of the Association for Computational*

Linguistics: System Demonstrations, pages 101–108, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-demos.14. URL https://aclanthology.org/2020.acl-demos.14.

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

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

Timo Schick and Hinrich Schütze. Few-shot text generation with natural language instructions. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 390–402, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.32. URL https://aclanthology.org/2021.emnlp-main.32.

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

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, October 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-demos.6. URL https://aclanthology.org/2020.emnlp-demos.6.

Chien-Sheng Wu, Steven C.H. Hoi, Richard Socher, and Caiming Xiong. TOD-BERT: Pre-trained natural language understanding for task-oriented dialogue. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 917–929, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.66. URL https://aclanthology.org/2020.emnlp-main.66.

Yuting Yang, Wenqiang Lei, Juan Cao, Jintao Li, and Tat-Seng Chua. Prompt learning for few-shot dialogue state tracking. *CoRR*, abs/2201.05780, 2022. URL https://arxiv.org/abs/2201.05780.