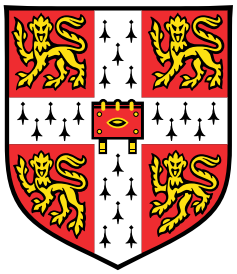
GPT-3 for Few-Shot Dialogue State Tracking



Nicholas Pezzotti

Department of Engineering
University of Cambridge

This dissertation is submitted for the degree of
Master of Philosophy

Clare Hall

August 2021

Declaration

I, Nicholas Pezzotti of Clare Hall, being a candidate for the MPhil in Machine Learning and Machine Intelligence, hereby declare that this report and the work described in it are my own work, unaided except as may be specified below, and that the report does not contain material that has already been used to any substantial extent for a comparable purpose. The word count, excluding declarations, bibliography, photographs and diagrams, but including tables, footnotes, figure captions and appendices is 14,957.

This thesis makes large use of the GPT-3 OpenAI API (https://openai.com/blog/openai-api/); this software is, as of the date of the submission of this dissertation, in private beta and therefore is not available to the general public. The rest of the software is written in Python and and can be found at https://github.com/nickpezzotti1/GPT-3-for-Few-Shot-DST.

Nicholas Pezzotti
August 2021

Acknowledgements

Foremost, I would like to acknowledge Prof. Bill Byrne for his continual and patient guidance throughout the project: I received the perfect amount of guidance and freedom during my research. He was always available to discuss any technical questions I had and, if needed, refer me to the right person to contact for further details. For example, he introduced me to Andi Tseng and Alex Coca, his PhD students, whom I am also grateful to for the invaluable feedback they gave me.

I also want to thank the OpenAI team for providing me with GPT-3 Beta access, the generous grant needed to pursue this research and for the helpful conversations and brainstorming sessions I had with Fraser Kelton, Yash Dani and Russell Foltzsmith.

Ultimately, none of this nor any of my other achievements would have been possible without the love and support of my family.

Abstract

GPT-3 (Brown et al., 2020) has attracted considerable attention due to its superior performance across a wide range of Natural Language Processing (NLP) tasks, especially with its powerful and versatile in-context few-shot learning ability. That is, it has been shown that by carefully crafting a prompt, consisting of a few labelled examples followed by an unlabelled example, GPT'3 is able to do few-shot sentiment classification, three-digit arithmetic and much more. We seek to evaluate its performance on a novel and notably more complicated task: few-shot Dialogue State Tracking (DST).

We propose a few-shot prompting framework that selects in-context examples based on similarity which outperforms the original random in-context selection framework. We also review and formalise the two types of completion strategies employed by previous literature, which we name constrained and unconstrained, and propose a third "semi-constrained" completion strategy, which is particularly well adapted for DST. Additionally, we propose a prompt ensembling technique that reliably outperforms individual models. Furthermore, we are the first, to the best of our knowledge, to fine-tune GPT-3 for the task of few-shot DST, showing that it reliably outperforms its GPT-2 counterpart.

Furthermore, we seek to synthesise and formalise the largely heterogeneous body of previous work on prompt programming and in-context learning for GPT-3. In an attempt to contribute to the understanding of the strengths, weaknesses and inner-working of GPT-3, we perform numerous ablative studies that validate and confute previous in-context learning empirical findings: mainly, we find that natural language instructions in the prompt have little impact on performance, larger language models do not always induce higher downstream performance and that GPT-3 is highly sensitive to the order and number of the in-context examples.

Table of contents

List of figures

List of tables

Chapter 1

Introduction

Ever since OpenAI released Generative Pre-trained Transformer 3 (GPT-3) (Brown et al., 2020), an autoregressive language model of 175B parameters, an order of magnitude larger than anything else of its kind at the time, it has attracted lots of attention, expanding beyond the academic literature and reaching mainstream media (GPT-3, 2021). Brown et al. (2020) demonstrated that large language models are capable of much more than predicting the next most likely word given a text sequence. For example, without any gradient updates or fine-tuning, via its powerful and versatile in-context few-shot learning capabilities, it achieves strong performance on numerous tasks such as translation, question-answering and simple arithmetic (Brown et al., 2020). Similarly, it has seen large adoption across commercial applications such as GitHub co-pilot, powered by OpenAI's Codex (Chen et al., 2021), which can generate code in numerous languages given natural language description of the code, such as Python Docstring comments. Despite its success, right in line with Moravec's paradox (Moravec, 1988), it has demonstrated limited understanding of basic logic, entailment and common-sense physics (Brown et al., 2020). With such a powerful tool newly at the fingertips of academic research, the possibilities and open questions are endless.

At the same time, as the amount of digital data continuously grows, so have users' demand for technologies that offer quick access to such data. Users increasingly rely on digital assistants that support information search such as Siri, Google Assistant, Amazon Alexa, *etc.* These technologies, capable of processing Task-Oriented Dialogue (TOD), allow the user to converse with a computer system using natural language. Crucial to the inner-working of TOD systems is the Dialogue State Tracking (DST) component, responsible for parsing the textual representation of the dialogue to understand what the user is trying to achieve.

In this dissertation, we demonstrate GPT-3's capabilities of performing DST with just a few-shots, that is with just a few examples of the DST task. We know GPT-3 can hold

a conversation with a user, even convincing them that a human is on the other end (Elkins and Chun, 2020). It is even a great actor: given a personality description, it sets the tone of its responses to that of the described persona (Appendix A). This suggests that GPT-3 is able to model dialogue behaviour: it is able to keep track of contexts, match the user's tone and even make a joke (Winters, 2021). The challenge of GPT-3 DST, therefore, lies not only in understanding the dialogue but also in finding ways to probe the language model with the aim of extracting the necessary knowledge from the model and outputting it in a structured form. This is achieved by carefully crafting a prompt that is fed to GPT-3 which completes it with the most likely sequence of words. This leverages the technique known as "in-context learning" (Brown et al., 2020): whereby the model is conditioned on a few demonstrations of the task and is then expected to complete further instances of the task simply by predicting the most likely completion. We show that GPT-3, by leveraging its in-context learning abilities, previously applied principally to simple classification tasks such as sentiment classification (Liu et al., 2021a) and textual entailment (Perez et al., 2021), is capable of achieving competitive results on DST, a significantly more complex task than any other reported in the literature to date, if the prompt is crafted effectively and the language model constrained appropriately.

Amongst all the tasks GPT-3 was applied to in the original paper, the only sequence-to-sequence task is machine translation: given text in one language translate it into a specified target language. For GPT-3, which was trained primarily on natural language, converting from one language to another is a much easier task than converting natural language to a structured format, such as dialogue states. If we define the vocabulary of the output sequence as the set of words and symbols allowed in the output and its syntax as how elements in the vocabulary can be conjoined, then the difference in difficulty in the two tasks becomes apparent. For translation, the vocabulary and the syntax of the output is simply the vocabulary and syntax of the output language, which it has learned during pretraining[1]. On the other hand, for DST, as long as the dataset in question did not appear in the pretraining data, it is not aware of either the vocabulary or the syntax of the output sequence. Therefore, we can think of DST as translation into an unknown language whose vocabulary is (usually) some subset of the dialogue's language's vocabulary and the syntax is unknown as it depends on the type of DST that is being performed and the way the belief states are linearised. Both these pieces of information are usually inferred by the model as it sees thousands of training examples and it is constructed into the architecture of the model (an output layer that has dimension equal to the number of domains/slots in the dataset). In the case of in-context

[1]While GPT-3 was trained primarily on the English language, throughout its pretraining phase it has seen large amounts of text also from most other languages (Brown et al., 2020).

learning with GPT-3, this has to be learned from the context alone. Furthermore, while examples of translation are scattered throughout the internet and GPT-3's pretraining data in the form of multi-language websites and websites dedicated to translation, examples of DST are much scarcer and likely not part of its pretraining data at all, forcing GPT-3 to really do few-shot learning. While we could provide long natural language task specification, enumerating the output vocabulary and exampling it in its syntax, we are limited in the length of the prompt, such that the prompt and its completion must not exceed 2048 tokens ($\sim$ 700 words). For that reason, we must resort to cleverly selecting the in-context examples to show how DST should be done for similar dialogues and constrain the search to best adhere to the output syntax and vocabulary.

Firstly, we propose two completion search methods (*constrained* and *semi-constrained search*), which outperform the traditional unconstrained search used by the original authors (Brown et al., 2020) by constraining the model's output to a reduced vocabulary. Since we found that currently the bottleneck of the development and experimentation of GPT-3 based applications is computational cost, we also provide systematic cost analysis of these and other methods investigated. At the same time, we expect these costs to sharply decline as competitors join the market, the technology improves, and compute costs shrink.

Thereafter, we demonstrate that the framing of the prompt has large effects on the performance of the downstream task of DST. Specifically, we first propose a framework that selects few-shot examples balancing similarity and diversity. Then, we validate previous prompt programming findings on different tasks by studying them under the light of the complex task of DST, such as the impact of the number and order of in-context example ordering and the effect of model size on DST performance.

Finally, we also propose an ensemble technique for prompts that reliably results in higher performance than any individual model across our experiments.

1.1 Thesis Overview

The remainder of the dissertation is structured as follows:

In Chapter 2, we present the related literature with particular focus on GPT-3 and its in-context learning capabilities and on DST for the datasets used for evaluation (MultiWOZ and Schema Guided Dialogue Dataset).

In Chapter 3 we present our approach for performing few-shot DST using GPT-3. Mainly we introduce previous and novel prompt programming techniques that lead to higher downstream performance. Additionally, we present our method for determining our experiment size: to stay within the computational constraints of this research, while also investigating

numerous different experimental settings, we developed a method for reducing the datasets' size by a factor of ten while maintaining reliable and consistent experimental results.

Chapter 4 we validate the approaches presented in the previous chapter by evaluating their DST performance on SGD and MultiWOZ. To get a better understanding of GPT-3, the proposed approach and its transferability to tasks other than DST, we perform a series of ablative studies of the systems.

In conclusion, Chapter 5 summarises the previous sections, presents the limitations and the broader impacts of the method as well as propose future directions of research.

Chapter 2

Background

Following the structure of the title, we present the relevant literature firstly by introducing language models and their meta-learning capabilities, with a particular focus on GPT-3 and its in-context capabilities and, thereafter, follow with a review of Dialogue State Tracking (DST) and its application to the two datasets used in the evaluation.

2.1 Predictive Language Models

A language model is a probability distribution over sequences of tokens, atomic pieces of language, such as a word, sub-word units or characters (Bengio et al., 2003). Given a sequence of m tokens, it assigns a probability $p(t_{1:m})$, where $t_{1:m}$ refers to all tokens in the sequence including the first and the m-th one. Similarly, a probability can be assigned to each token conditioned on the other tokens in the context as follows: $P(t_k|t_{1:k-1}, t_{k+1:m})$. Given a vocabulary V, if each possible joint probability was stored in a probability table conditioned on every possible combination of tokens in the vocabulary, then the table would grow exponentially with m: specifically, at the rate of $\Theta(|V|^m)$, where $|V|$ is the size of the vocabulary V. Another problem, which is more constraining in practice, is data sparsity: the vast majority of possible token sequences are not even observed in training. One solution is to make the assumption that the probability of a word only depends on the previous n words, rather than all previous and all future words. This is known as an n-gram model (Shannon, 1948) and works by calculating the co-occurrence statistics of tokens in the training data. The aforementioned causal relationship assumption not only drastically reduces the complexity of the task but also conveniently allows the models to be utilised in a generative manner by autoregressively predicting the next most likely token, hence the terminology "predictive language models". As RNNs (Rumelhart et al., 1985) and LSTMs (Hochreiter and Schmidhuber, 1997) gained popularity, they quickly spread to predictive language models,

but it wasn't until the adoption of the Transformer's (Vaswani et al., 2017) encoder-decoder architecture that the real potential of language models became apparent. The Transformer's architecture, thanks to its self-attention mechanisms, is capable of processing large amounts of data much more efficiently while also overcoming the "long-term" dependency problem (Bengio et al., 1993) that plagued previous sequence models. It was found that the decoder component of the model worked excellently as a causal predictive language model (Radford et al., 2018). It was then that researchers realised that it is possible to obtain state-of-the-art performance on a wide array of tasks by finetuning predictive language models (Brown et al., 2020).

Ever since researchers have been able to increase model capacity by increasing the number of parameters and the amount of training data. The growth in the number of parameters of the latest models over past few years speak for themselves: from 100 million parameters (Radford et al., 2018), to 300 million parameters (Devlin et al., 2018), to 1.5 billion parameters (Radford et al., 2019a), to 8 billion parameters (Shoeybi et al., 2019), 11 billion parameters (Brown et al., 2020), and 17 billion parameters (Rosset, 2020), 1.6 trillion parameters (Raffel et al., 2019a) and finally 1.75 trillion parameters (Zhavoronkov, 2021) [1]. Each increase has reliably led to improvements on the downstream task, and there is evidence suggesting that log loss of the model, which has been shown to correlate well with the performance of many downstream tasks, follows a smooth trend of improvement with scale (Kaplan et al., 2020) and has yet to plateau. This suggests that the trend of ever-growing language models coupled with higher downstream task performance is unlikely to subside anytime soon (Hernandez et al., 2021).

2.2 GPT-3

GPT-3 (Brown et al., 2020) is the third generation of GPT, the first predictive language model combining the Transformer (Vaswani et al., 2017) architecture and the self-supervised pre-training objective (Xu et al., 2021). It is trained in a self-supervised fashion on large corpora of publically available semi-curated text datasets. The weighted pre-training dataset is composed of Common Crawl (Raffel et al., 2019b) (60%), itself made up of 410B byte-pair-encoded tokens scraped from numerous mostly English websites. The other pre-training datasets are WebText2 (Kaplan et al., 2020) (22%), an extended version of WebText (Radford et al., 2019b); Books1 (8%); Books2 (8%); and Wikipedia (3%). After pretraining[2], it

[1] These last two numbers are not directly comparable with that of GPT-3, as not all weights are used at each forward pass, but they still indicate a general trend towards larger language models.

[2] The details of pretraining are omitted for clarity and conciseness, as not essential for the understanding of this dissertation. For details, refer to (Brown et al., 2020).

became clear that the increase of an order of magnitude in the number of parameters led to an equally outstanding improvement in the model's capability of generating fluent text. Moreover, it was capable of performing many other NLP tasks without fine-tuning or gradient updates via its in-context learning ability (Brown et al., 2020).

OpenAI decided not to opensource the model's code or its weights, but it has built a closed-access Beta API for experimentation with all of their 4 GPT-3 models of varying sizes: "S" (2.7 billion parameters), "M" (6.7 billion), "L" (13 billion) and "XL" (175 billion)[3]. The model uses GPT-2's dynamic tokenization (Radford et al., 2019a) to subdivide the prompt and the completion into tokens. Crucially, the GPT-3's computational cost translates directly into monetary cost with a per-token cost that depends on the size of the model used, where the user is charged based on the number of tokens in the prompt and the number of tokens in the completion. For example, the prompt $p =$ "The answer to life is", which if fed to GPT-3 generates the completion $c =$ "42", is tokenized as follows: $p =$ "|The| answer| to| life| is|" and $c =$ "|42|", where | indicates token boundaries. Therefore the cost of using GPT-3 in this case is $(|p| + |c|) \times \textit{per-token price} = 5 \times \textit{per-token price}$, where $|p|$ and $|c|$ are the token length of the p and c, respectively, and *per-token price* depends on the model size (see Appendix B for exact amount).

2.3 In-Context Learning

Rather than generating text from scratch, a prompt can be provided for completion to GPT-3, which it autoregressively completes with what it considers to be the most likely tokens - this technique, known as prompting, allows GPT-3 completions to be conditioned and guided by human-specified text. GPT-3 then greedily selects the next most likely token. While this method is not theoretically optimal and other methods such as beam search could be used, it is reportedly effective in practice (Brown et al., 2020). Amongst all applications of prompting, the approach that attracted the most attention is "in-context learning" (Brown et al., 2020): carefully crafting a prompt, composed of one or more labelled examples followed an unlabelled one, as shown in Figure 2.1, GPT-3 is able to learn from the labelled examples and make a prediction about the final label. Unlike traditional few-shot learning, in-context learning occurs within the forward-pass upon each sequence and therefore with no parameter updates, despite its misleading nomenclature "learning". This is because during pre-training, GPT-3 learned a broad set of skills and pattern recognition abilities

[3]These models are also referred to as Ada, Babbage, Curie and Davinci, respectively. The exact number of parameters has not been disclosed by OpenAI, the numbers provided are estimates provided by Hendrycks et al. (2020).

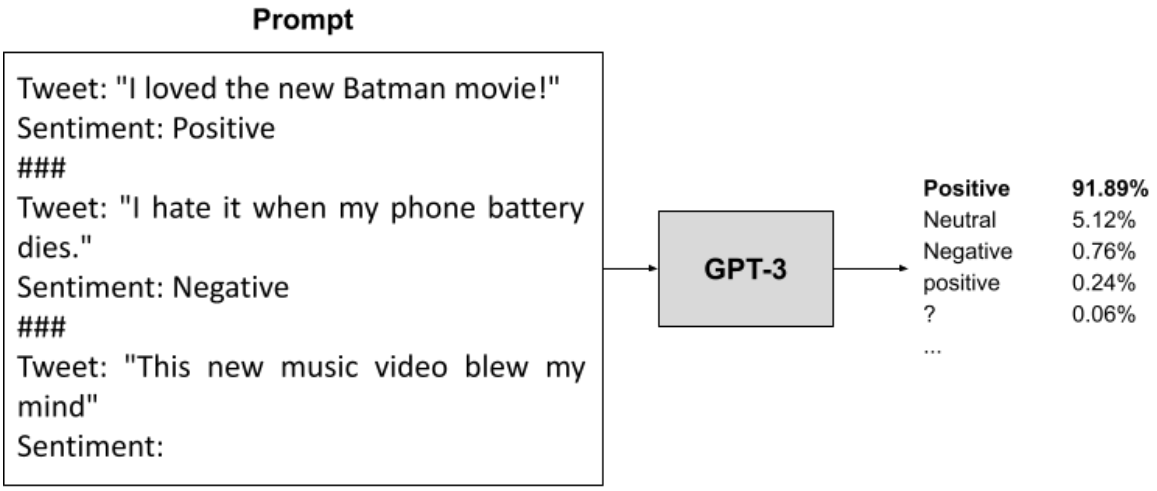


Fig. 2.1 An example of GPT-3's in-context learning ability. When the text on the left is provided as the prompt to GPT-3, it is able to correctly identify the next most likely word in the text sequence which corresponds to the sentiment label of the target tweet.

and it can then use these abilities at inference time to rapidly solve the desired task. This framework was shown to perform strongly on many NLP benchmarks, including translation, question-answering and cloze tasks, as well as several tasks that require on-the-fly reasoning or domain adaptation, such as unscrambling words, using a novel word in a sentence, or performing 3-digit arithmetic (Brown et al., 2020).

While in-context learning can be assimilated to fine-tuning as both serve as a way of conditioning a language model on examples, the latter generally achieve higher downstream performance, although it also results in a large language model that is costly to share and serve Lester et al. (2021); Li and Liang (2021). In-context learning tend to perform worse than fine-tuning because it is constrained by the model's maximum prompt length (Brown et al., 2020), which acts as an information bottleneck: mainly, fine-tuning can make use of more of the training data than an in-context learner. Brown et al. (2020) have shown that, for large language models, when the number of in-context examples increases, generally the performance of the in-context learning approaches that obtained by the fine-tuned model. This suggests that as the model's prompt limit information bottleneck widens, the performance gap between in-context learners and fine-tuned models shrinks. Given the practical prompt length limits of predictive language models such as GPT-3, Shin et al. (2020) found that it is possible to obtain significant improvements over manually engineered prompts by performing a discrete search over all possible prompts. More recently, it has been shown that by "prompt tuning" (Lester et al., 2021), similarily to "prefix tuning" (Li and Liang, 2021), can be used to obtain "soft-prompts" that can condense the signal from an entire labelled dataset into

that of a short prompt closing the downstream task performance gap with fine-tune models. These soft-prompts are obtained via gradient descent on the parameters corresponding to the prompt embedding, which is equivalent to perform the search on the continuous embedding space of the prompt. This is similar to fine-tuning a model whose weights are all frozen but the last layer; except that the parameters corresponding to the prompt embedding are fine-tuned and all other parameters are frozen instead. The advantage of prompt tuning is that, unlike large fine-tuned language models, this allows the use of the same large language model, such as GPT-3, for numerous downstream tasks, by simply sharing this soft-prompt rather than all the parameters of an entirely different fine-tuned model, greatly reducing the sharing and serving cost. We conjecture that soft-prompts can therefore be viewed as a theoretical upper bound on the performance of the downstream task if the optimal prompt was selected; though, in most cases, this prompt is in practice unconstructable due to the limitations of the finite and discrete space natural language prompts span.

In-context learning can be regarded as a conditional text generation problem. Concretely, the probability of generating a label $y = t_{m:n}$ conditioned on a prompt $P = t_{1:m-1}$ is expressed as

$$p(y|P) = \prod_{i=m}^{n} p(t_i|t_{1:i-1}) \tag{2.1}$$

2.4 Dialogue State Tracking

DST refers to accurately estimating the user's goal as a dialogue progresses. It is useful because it helps in effectively reducing ambiguity inherent in language within a temporal process like dialogue. DST is the act of determining the current state of the frame, usually in the form of slot-value pairs, and, in some cases, the user's most recent dialogue act (Jurasfky and Martin, 2000). The state of the frame thus does not depend solely on the latest utterance pair, rather it depends on all previous utterances summarising all the user's constraints up to the current dialogue turn. The set of all slots is composed of two subsets: the informable slots (often referred to as just "slots") and the requestable slots (Henderson, 2015). Informable slots are attributes of the entities that the user may use to constrain their search. Requestable slots are attributes of entities that the user may ask the value of. The slots can either be categorical, when they can only take one of a predefined set of values, or non-categorical, when the value can be anything. Each slot also can take one of the following special values: *Dontcare*, when the user has no preference; or *None*, when the user has not yet specified a value for that slot.

In the datasets we study, we assume that the dialogues are between only two actors (the user and the system) and that the dialogue turns are labelled only for user turns. Therefore, we can write that for each dialogue composed of n utterance pairs (also known as turn pairs or dialogue turns) $D = \{(s_0, u_0), (s_1, s_2), \ldots, (s_n, u_n)\}$, the corresponding dialogue belief state labels are $y = \{y_1, y_2, \ldots, y_n\}$, where y_n corresponds to the dialogue belief states up until turn n.

2.5 MultiWOZ 2.1

Multi-Domain Wizard-of-Oz dataset (MultiWOZ) (Budzianowski et al., 2018) is a fully-labelled collection of human-human written conversations spanning over multiple domains and topics. The dialogues span over 8 domains (restaurant, hotel, attraction, taxi, train, hospital and police) some of which only appear in the test set (hospital and police). There are 3,406 single-domain and 7,032 multi-domain (2-5 domains) dialogues. At the time of its creation, MultiWOZ was one of the largest annotated task-oriented corpora. Larger corpora have since been created, with different annotation schemes, but the MultiWOZ collections are still widely used in the literature for reporting progress in dialogue belief state tracking. Given that it was gathered from human-human dialogues via Amazon mechanical turk, the data presented numerous mistakes and has had several iterations of data cleaning. For this reason, we experiment with version 2.1[4].

In this dataset, DST consists of accurately tracking the informable slots. The belief state therefore takes the form $\{domain_i : \{slot_j : value_k\}\}$, where $domain_i$ is one of the eight domains enumerated above, $slot_m$ is the m-th slot of $domain_i$, and $value_k$ is the corresponding value. The brace notation indicates a dictionary or list of key-value pairs. Throughout our experiments we linearise the belief state in the following string form: $domain_i$-$slot_j$=$value_k$;$\cdots$;$domain_{i'}$-$slot_{j'}$=$value_{k'}$, where each slot-value pair is semi-colon separated. While the belief state is permutation invariant (i.e. the order does not matter) and case insensitive, to ensure deterministic behaviour, it is sorted in alphabetical order of the keys. Here follows an example of a single-domain dialogue and its associated belief states in the linearised format:

s_1: ""

u_1: "hi, can you find a brunch place in Cambridge?"

[4] Although a newer 2.2 version is available, it has not been widely used in experimentation yet and was therefore not selected for our experiments

y_1: restaurant−city = cambridge; restaurant−type = brunch

s_2: "can I recommend hot numbers?"

u_2: "yes please"

y_2: restaurant−city = cambridge; restaurant−type = brunch;
restaurant−name = hot numbers

The metrics we report for DST on this dataset are the Average Joint Slot Accuracy (AJA) (Equation 2.2) and the Average Slot Accuracy(AVG)[5] (Equation 2.3), where the labels are in set notation and $y_{n,k}$ refers to the label of the k-th turn of the n-th dialogue. The former of the two metrics corresponds to the relative frequency with which the model correctly predicts all the slots in the current belief state. The latter is the accuracy with which a slot is predicted correctly taking into account both false positive and false negatives.

$$\text{AJA} = \frac{\sum_n \sum_k y_{n,k}}{\sum_n \sum_k \widehat{y}_{n,k}} \tag{2.2}$$

$$\text{AVG} = \frac{\sum_n \sum_k y_{n,k} \cap \widehat{y}_{n,k}}{\sum_n \sum_k y_{n,k} \cup \widehat{y}_{n,k}} \tag{2.3}$$

State-of-the-art results are traditionally obtained by encoding the dialogue history (usually with a fine-tuned BERT (Devlin et al., 2018)) followed by a feed-forward neural network and some post-processing to generate a distribution over domains, slots and consequently values (Mehri et al., 2020). Recently, decoder-based DST models, such as simple-TOD (Hosseini-Asl et al., 2020) and UBAR (Yang et al., 2020) have both obtained competitive results by finetuning GPT-2 (Radford et al., 2019a) for end-to-end DST.

2.6 Schema Guided Dialogue (SGD)

Today, the most widespread type of Task-Oriented Dialogue (TOD) systems are in large scale virtual assistants, such as Alexa, Google Assistant and Siri. These systems integrate with

[5]This is different from the slot accuracy sometimes reported (Budzianowski et al., 2018), which only accounts for slot name accuracy and not the value. We opt for this metric as, while it is less frequently used in literature, it is more complete and informative.

numerous third-party APIs to be able to perform actions across numerous, ever-growing and potentially overlapping services. For example, Siri can play a song, find flights and make dinner reservations. Moreover, these services vary amongst users (one user might use Apple Music and the other Spotify) and evolve over time as the app is updated or the user installs more services. While datasets of dialogues between humans and virtual assistants exist, few have all the characteristics described above. Recently, Rastogi et al. (2020) released the largest public dataset that simulates this environment: a set of 16,000+ multi-domain TODs between a human and a virtual assistant spanning 26 services belonging to over 16 domains. In the same paper, they propose the Schema Guided TOD (SG-TOD) system as an alternative to what they refer to as ontology-oriented TOD systems. SG-TOD systems are unified natural language interfaces that integrate with a large number of dynamic internal web services via their APIs.

In this new paradigm, every service provides a schema that provides the name of the service, the functions provided (intents) and their respective parameters (slots). Each of the aforementioned components is also accompanied by their natural language description and, for the slots, if they are categorical, then also the permitted values. In previous literature, the schema is used to learn a distributed semantic representation, which is given as an additional input to the dialogue system (Ma et al., 2019). Therefore, the dialogue system can be implemented as a single unified model, containing no domain or service-specific parameters. These systems, by generalising their learning across services, should be capable of leveraging common knowledge and operating on never before seen services (zero-shot learning).

The data structure of SGD is similar to that of MultiWOZ, though now the belief states are different as each system-user utterance pair in each dialogue is annotated in three ways:

- Active intent(s): this describes the task the user is trying to accomplish. It is composed of both the service name and the intent within that service (e.g. *restaurants_1/bookTable*). All the services, active intents and their respective natural language descriptions are known a priori, as they are reported in the schema. Each dialogue turn has at most one active intent per service, though it can have none, at which point the *None* label is assigned for that service's active intent. One challenging aspect of this task is that some services have overlapping functionality: for example, *restaurants_2* never appears in the test set but it has some similar, though not identical, functionality to the *restaurant_1* service; the challenge lies in the model having, therefore, to transfer its learnings from one service to the other. For this task, the metric tracked is active intent accuracy (Intent Acc.), or the fraction of user turns for which the active intent is correctly identified.

- Goal: This corresponds to the informable slots, analogously to MultiWOZ; this task can be broken down in two: first recognising the slots that are being mentioned (directly or indirectly) in the dialogue so far (so it accumulates over turns); second, determining the value of that slot. As in MultiWOZ both the AJA (Equation 2.2) and AVG (Equation 2.3) metrics are computed.
- Requested Slots: This corresponds to the slot names that the user is requesting in the current utterance (requestable slots). For this task, the F1 score is computed to take into account both specificity and sensitivity (Req. F1).

The linearisation of the belief state therefore also differs, as the active intents classification and requested slots prediction can be viewed as parallel tasks to Goal prediction. An example dialogue and respective linearised belief state is shown below[6]:

s_1: "There is 1 movie called The Poseidon Adventure."

u_1: "The Poseidon Adventure would be great for me to watch now."

y_1: Active Intents: Media_1/PlayMovie | Requested Slots: None | Slots: movie_name = The Poseidon Adventure;

Generally, the three aforementioned metrics are reported under three categories: all, seen and unseen. These correspond to the metrics reported for all services, for services seen in the training data, and for only the unseen services. If no category is specified when reporting results, then the all category is assumed.

As for MultiWOZ, the standard approach, first proposed by Rastogi et al. (2020) and further improved by Ma et al. (2019), consists of using an encoder-based model. Other than the dialogue history, these systems also encode the schema. Then, for each service, the dialogue and schema encodings are fed into a feed-forward neural network terminated by a softmax, trained to predict the active intent (or *None*, if no intent is active for the given service), and by a gelu (Hendrycks and Gimpel, 2016) activation function, trained to predict

[6]There is at most one *Active Intent* per service per dialogue turn. There can be any number (up to the number of slots) of *Requested Slots* or *Slots* and they are permutation invariant (i.e. the order does not matter) and case insensitive.

slots as requested if the activation is above a predefined threshold, for requested slot prediction. For goal prediction, the approach differs when predicting categorical values and free-form ones: if the value is categorical, then it is predicted similarly to the service's active intent and slots; if it is continuous, then a start and end index are predicted such that $(s_n + u_n)[start_{idx} : end_{idx}]$ corresponds to the slot value, where $+$ represents the string concatenation operator and $s[i : j]$ indicates the substring of string s spanning from index i to index j, inclusive. For example, if $s_n =$ "How can I help?" and $u_n =$ "I want to book a table at the Giggling Squid.", then for the slot restaurant-name the model would predict $start_{idx} = 45$ and $end_{idx} = 59$, such that $(s_n + u_n)[start_{idx} : end_{idx}] =$ Giggling Squid. Another successful approach has been that of reducing SGD DST to a form of closed-book question answering (Zhang et al., 2021). To the best of our knowledge, no attempts of applying a predictive language model for SGD DST.

Chapter 3

Approach

In this chapter, we present our approach for performing DST on MultiWOZ and SGD by leveraging GPT-3's in-context learning abilities. Specifically, we showcase our prompt design (Section 3.1), explain the different ways to constrain GPT-3's completion search (Section 3.2) and present how prompt ensembling works (Section 3.3). Finally, we also present our experimental set-up (Section 3.4), which includes the hyperparameter configurations and an analysis of MultiWOZ's and SGD's test sets; in particular, we discuss how we significantly reduce the datasets' evaluation split's sizes while maintaining consistency of the metrics across experiments.

3.1 Prompt

Listing 3.1 An example of the slot difference between the labels y_{n-1} and y_n.

```
y_{n-1}: restaurant-city = cambridge; restaurant-type = brunch

y_n: restaurant-city = cambridge; restaurant-type = brunch;
    restaurant-name = hot numbers

y_{n-1}-y_n: restaurant-name = hot numbers
```

Despite its powerful and versatile in-context learning ability, GPT-3 has some practical challenges/ambiguities. The original paper (Brown et al., 2020) randomly samples from the task's training set to construct the context. In practice, it was found that the downstream task's results tend to fluctuate sporadically with different choices of in-context examples

System: """
User: "find me a place to stay which has 0 stars and preferably a guest house"
Slot Difference: hotel-stars = 0; hotel-type = guesthouse
###
System: "Can we narrow down your search by area?
User: "I would really like a guesthouse please."
Slot Difference: hotel-type = guesthouse

Fig. 3.1 An example of a prompt used to infer the belief state of a dialogue turn in the MultiWOZ dataset. The prompt is bold, while the completion by GPT-3 is not.

System: "There is 1 movie called The Poseidon Adventure."
User: "The Poseidon Adventure would be great for me to watch now."
Active Intents: Media_1/PlayMovie
Requested Slots: None
Slot Difference: Media_1/title = The Poseidon Adventure;
###
System: "I have successfully rented The Poseidon Adventure. You have a 3-day rental period to watch this film."
User: "How much will it cost to rent this movie?"
Active Intents: Media_1/RentMovie
Requested Slots: None
Slot Difference: None

Fig. 3.2 An example of a prompt used to infer the belief state of a dialogue turn in the SGD dataset. The prompt is bold, while the completion by GPT-3 is not. The alternation between bold and not bold text is obtained by providing the initial prompt to GPT-3, letting it complete until the end of the line, then appending more predefined prompt, and so on

(Liu et al., 2021a). Randomly selecting in-context examples has been shown to lead to decreased performance (Liu et al., 2021a; Perez et al., 2021). We validate this finding for SGD and MultiWOZ DST (Section 4.1), where randomly selecting in-context examples incurs the risk of all the selected examples being too dissimilar and therefore uninformative about how to label the target dialogue and leading to poor performance. For that reason, in crafting the prompt, we must carefully define an in-context example selection strategy. While brute-force approaches for finding the best combination of in-context examples have shown to be effective (Shin et al., 2020), the complexity grows combinatorially with the size of the training data; this quickly becomes infeasible as the number of training examples available increases. Following the most recent findings on effective prompt formulation (Liu et al., 2021a), we select in-context by searching through the training set for the k-nearest-neighbors to the target example. We adopt this method as it was found to produce more consistent and increased downstream performance (Liu et al., 2021a).

3.1.1 In-Context Example Selection Policy

To select the k-nearest-neighbors to the target example we use GPT-3's semantic search capabilities to find the examples in the training set that is closest to the target example. Specifically, we first perform a simple keyword search[1] that finds the k' (set to 15, unless otherwise specified) most similar documents with the highest number of matching words. These are then provided to GPT-3, which ranks the selection based on a similarity score that ranges from 0 to 300[2]. Then, the top k most similar examples are selected. An example of this selection policy is shown in Figure 3.3.

The initial filtering to just k' examples is done as a means of notably reducing the computations assigned to GPT-3. Moreover, to further reduce the computations across experiments, we cache these requests into a dictionary that maps every dialogue to its nearest neighbours. This can also be done as a form of pre-processing. As the semantic search module is deterministic and does not change across experiments, this further reduces the complexity as repeated requests' responses are stored across experiments.

[1]The exact implementation of this keyword search is not public knowledge.

[2]The scores are computed by combining the target example (query) and each training example (documents) into a prompt that is fed to GPT-3; the prompting technique used is not public knowledge. If future research decides to explore different search configurations, we recommend using an open-source alternative such as KATE (Liu et al., 2021a), which is based on BERT.

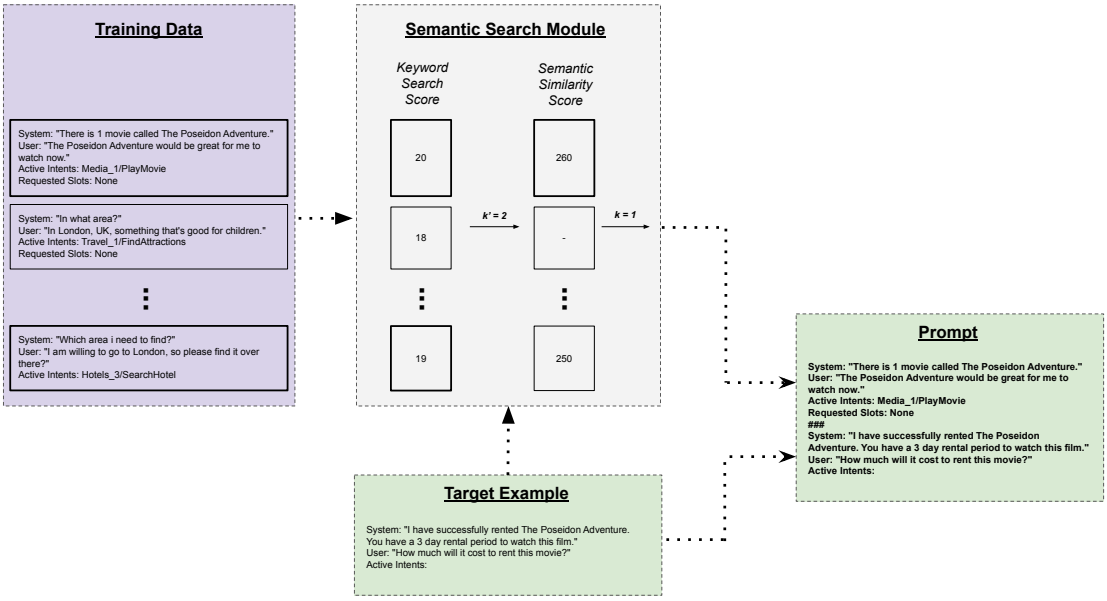


Fig. 3.3 This is an illustrative example of the in-context example selection policy used to construct a prompt used to classify a target example sampled from the SGD dataset. From left-to-right, the training data and the target example are fed to the semantic search module which, after filtering the whole training dataset to just $k' = 2$ examples, ranks the examples based on their semantic similarity and selects the $k = 1$ example that is most similar to the target example. Then, to construct the prompt, the k in-context example(s) and the target example are joined together, delimited by the separator "###" for clarity.

3.2 Completions

Selecting the most likely sequence of tokens given a prompt is the main function of language models. This is also known as generating completions. We identify three key types of completions: *unconstrained*, *constrained* and *semi-constrained*.

3.2.1 Unconstrained

By default, GPT-3 autoregressively selects the most likely token amongst its whole vocabulary. We refer to this as "unconstrained completions", as the selection happens amongst the entirety of the vocabulary. This is done by autoregressively selecting tokens according to the policy

$$t_{m+1} = \arg\max_{t \in V} P(t|t_{1:m}) \tag{3.1}$$

where $t_{1:m}$ is all preceding tokens, until the end of the line (*i.e.* until a newline character is predicted). To better understand this completion method and compare it to the others presented below, we can examine its asymptotic token complexity, which we define as the upper and lower bound on the number of tokens in the prompt and the subsequently generated completion. This is inspired by the asymptotic big-θ notation (Iwaniec and Kowalski, 2004) used to study the running complexity of algorithms. Unconstrained completions have token complexity of $\Theta(m + |c*|)$, where m is the token length of the prompt and $|c*|$ the token length of the completion.

3.2.2 Constrained

When doing tasks such as DST or classification tasks at large, it is common to have only a finite set of possible labels. For example, in MultiWOZ there are only 8 domains and each has a well-defined set of possible slots. Similarly, SGD has a schema that defines the possible services, intents, slots and, in the case of categorical slots, also the possible slot values. Therefore, to prevent the model from generating unallowed belief states, we constrain the search of token sequences to just ones allowed by the dataset. Given a set of completions $C = \{c_1, c_2, \cdots, c_n\}$ and a prompt $t_{1:m}$, categorical completions select the completion according to the following criteria

$$\arg\max_{c \in C} \; p(c|t_{1:m}) \tag{3.2}$$

Unfortunately, while this is trivial in theory, this is not available in OpenAI's API and the workaround we developed (presented in Appendix C) is very expensive in practice. In

particular, while the prompt remains unchanged compared to the unconstrained setting, the token complexity would in theory be $\Theta(m+\sum_{c\in C}|c|)$ if allowed by the API, but in practice is $\Theta(\sum_{c\in C}|c|m)$, where $|c|$ is the size of a completion c belonging to the set of all possible completions C. This difference in costs quickly scales as the set of possible values increases or the size of the model increases.

In this approach, one completion must be selected amongst a set of possibilities. This requires a scoring function to evaluate and rank each completion and then selecting the highest-ranking one. To date, three categorical completion scoring functions have been proposed: likelihood (Brown et al., 2020), per-token average likelihood (Brown et al., 2020) and Domain-Conditional Probabilistic Mutual Information (DC-PMI) (Brown et al., 2020; Holtzman et al., 2021).

Likelihood ($\mathscr{L}$)

The most common completion strategy is that of selecting the one with the highest total likelihood (Brown et al., 2020). Equivalently, without loss of generality, for computational convenience, the log likelihood is measured instead. Formally, we write:

$$\mathscr{L} = \arg\max_{c\in C} \log p(c|t_{1:m}) \tag{3.3}$$

Per-Token Average Likelihood ($AVG-\mathscr{L}$)

Brown et al. (2020) found that when completions have largely varying lengths, normalising their likelihood for length led to noticeable improvements in downstream performance. In particular, Brown et al. (2020) propose dividing the likelihood by $|c|$, the length of the completion in tokens. The corresponding policy is defined as follows:

$$\text{AVG-}\mathscr{L} = \arg\max_{c\in C} \log \frac{p(c|t_{1:m})}{|c|} \tag{3.4}$$

Domain-Conditional Probabilistic Mutual Information (DC-PMI)

DC-PMI was invented to tackle the problem of surface form competition errors (Holtzman et al., 2021), which categorical completions are susceptible to surface-form competition errors (Holtzman et al., 2021), whereby alternative formulation of the same underlying concept compete for probability mass. For example, if the user says “I want to travel to Milan, Texas” and we are trying to extract the value corresponding to slot *country* whose possible values are $C = \{\text{"}USA\text{"},\text{"}Italy\text{"},\text{"}England\text{"}\}$, GPT-3 correctly identifies "USA" to be the highest as $\forall_{c\in C} p(prompt+\text{"}USA\text{"}) \geq p(prompt+c)$; this can fail when the

possible completions are uncommon; for example, if the set of possible completions is $C = \{"U.S. of America","Italy","England"\}$, then GPT-3 predicts "Italy" as the most likely completion. This is because we can think of the prior of "Italy" to be comparable to that of "USA" but way higher than that of "U.S. of America", which rarely occurred in GPT-3 pre-training. For that reason, we say that GPT-3 is biased towards *a priori* likely completions.

To overcome this, Brown et al. (2020) introduced Domain-Conditional Probabilistic Mutual Information (DC-PMI) a method later formalised and extended by Holtzman et al. (2021). The scoring function has the following form:

$$\arg\max_{c \in C} \log \frac{p(c|t_{1:m})}{p(c)} \approx \arg\max_{c \in C} \log \frac{p(c|t_{1:m})}{p(c|domain)} \tag{3.5}$$

Where $p(c)$ is the prior of the completion. In practice, this quantity, which can be obtained by marginalising over all possible prompts, is intractable and therefore we resort to approximated it by $p(c|domain)$, which signifies having GPT-3 compute the likelihood of the completion conditioned on some short prompt. The original paper (Holtzman et al., 2021) found that, in zero-shot settings, simply conditioning on the label text ("Slot Difference:"/"Active Intent:"/"Requested Slots:") led to improved results. In our experiments (Section 4.1.1), we demonstrate that, for DST, higher performance can be obtained by conditioning on other short domain prompts.

3.2.3 Semi-constrained

To overcome surface-form competitions errors, maintain a low token complexity and constrain the search of possible completions, we introduce the third and final completion strategy: semi-constrained. This consists of inserting the possible values the label can take somewhere in the prompt and letting GPT-3 complete the prompt in an unconstrained manner. An instance of a prompt using this completion is shown in Figure 3.4. The advantage of this approach is that it suggests to GPT-3 the possible values the completion can take. This method has token complexity$\Theta(m + |c*| + \sum_{c \in C} |c|)$.

3.3 Ensemble Prompts

A novel approach is that of prompt ensembles: rather than feeding just one prompt to GPT-3 to get the corresponding completion, we provide multiple prompts, where one or more of its features are changed. Then each prompt is separately passed to GPT-3, which generates the predicted labels for each prompt. Finally, only the most popular predicted labels are selected.

System: "There is 1 movie called The Poseidon Adventure."
User: "The Poseidon Adventure would be great for me to watch now."
Active Intents (Media_1/FindMovies, Media_1/PlayMovie, Media_1/RentMovie): Media_1/PlayMovie
Requested Slots (Media_1/title, Media_1/genre, Media_1/subtitles, Media_1/directed_by): None
Slot Difference (Media_1/title, Media_1/genre, Media_1/subtitles, Media_1/directed_by): Media_1/title = The Poseidon Adventure;
###
System: "I have successfully rented The Poseidon Adventure. You have a 3 day rental period to watch this film."
User: "How much will it cost to rent this movie?"
Active Intents (Media_1/FindMovies, Media_1/PlayMovie, Media_1/RentMovie) : Media_1/RentMovie
Requested Slots (Media_1/title, Media_1/genre, Media_1/subtitles, Media_1/directed_by): None
Slot Difference (Media_1/title, Media_1/genre, Media_1/subtitles, Media_1/directed_by): None

Fig. 3.4 An example of a prompt used to infer the belief state of a dialogue turn in the SGD dataset for semi-constrained completions. The prompt is bold, while the completion by GPT-3 is not. The alternation between bold and not bold text is obtained by providing the initial prompt to GPT-3, having it complete until a predefined stop character (in this case, the new line character), including the completion in the new prompt and appending more predefined prompt.

We identify two key steps in this approach: the ensemble prompt generation and the voting mechanism.

While other prompt ensemble techniques have been proposed (Gao et al., 2020; Schick and Schütze, 2020b), none have been used for in-context learning and they all require the manual formulation of multiple prompts, which is costly and sometimes not possible.

3.3.1 Ensemble Prompt Generation

For the ensembled prompts generation, we settled for an exclusive in-context example roulette selection methodology. This consists of selecting k in-context examples from the *max_rerank* keyword-search filtered examples, scoring them with GPT-3 as before, but rather than selecting the top k most similar, they are chosen via roulette selection with fitness proportional to their similarity score. This means each example is selected randomly with probability:

$$P(e_k) = \frac{sim(t, e_k)}{\sum_k sim(t, e_k)} \tag{3.6}$$

Where $sim(t, e_k)$ is a similarity measure between the target example t and e_k as computed by GPT-3 and described in Section 3.1. The "exclusive" nomenclature is due to the sampling method: after an example is selected it is excluded from future samples within this and all other ensemble models. This is opposed to what we call inclusive sampling, where the drawn samples can be drawn again and therefore different prompts within the same ensemble can contain overlapping in-context examples.

3.3.2 Ensemble Voting Mechanism

We employ a majority voting mechanism: a label is selected if at least half of the ensemble's predictions contained that label. For example, if three different prompts are generated then the following linearised belief states is outputted:

```
Slot Difference: time=2pm, day=Wednesday,
Slot Difference: time=2pm, restaurant=Honest Burger,
Slot Difference: None
```

Then the resulting belief state would only include the slot-value pairs that appeared in at least half of the belief states:

```
Slot Difference: time=2pm
```

This helps in sifting out the language model's hallucinations, whereby GPT-3, when generating text in an unconstrained manner, tends to produce factually incorrect text. This is particularly problematic in a task like DST, where generating well-structured, precise and factually correct text is essential. Moreover, our experiments tended to show that hallucinations in the generated belief state tended to be highly correlated with the in-context examples provided. For example, if some of the in-context example's belief states included a given slot-value pair, such as date=13/03, then when the model generated the target belief state it would tend to also include the same slot-value pair. By generating multiple belief state completions, but only selecting the slot-value pairs when at least half of the generated belief states included it, this effectively kept only the factually correct components of each belief state. This is because, while generated belief states could still include hallucinations, they would be different hallucinations and therefore not included in the final belief state.

3.4 Experimental Setup

3.4.1 Model Configuration

All experiments in Chapter 4, unless otherwise specified, are run with the smallest of the GPT-3 models presented in Section 2.2 (S) and $k = 2$ in-context examples. This was done because the smaller models and shorter prompts are the most token efficient and therefore allow for more experiments.

All experiments are performed with models with deterministic completions (temperature set to 0). All other model hyperparameters are left to their default values.

3.4.2 Experiment Size

Due to the limited funds for this thesis and the high computational cost of large language models (per-token model costs shown in Appendix B), compute was the bottleneck in our ability to gather data. Therefore, there was a constant trade-off to be made: run many experiments on reduced test sets or run few experiments on the whole test sets.

The method we adopted to manage this trade-off was to select our baseline model and run it on each of the datasets' test partitions $n = 5$ times. Each time the order of the dialogues in the test set was shuffled (the order of the turns within dialogues remains unchanged). Then, based on the running metrics, we selected the minimum experiment size, as measured by the percentage of the test set, for which the standard deviation of each metric was below 0.01. While this method is effective at selecting an experiment size that results in consistent and

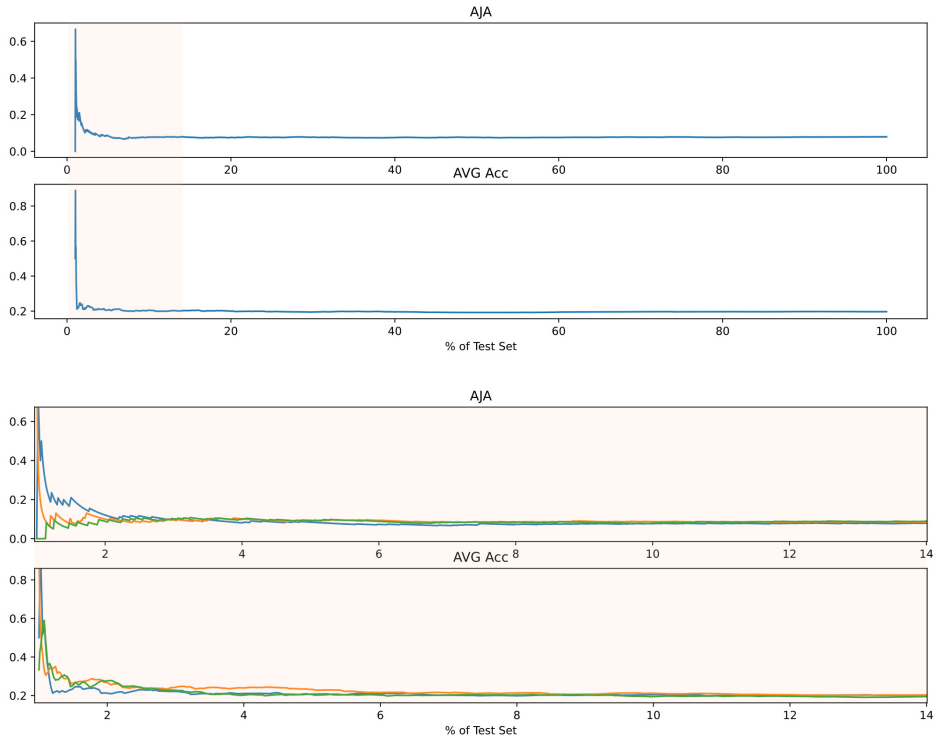


Fig. 3.5 On top a visualisation of the running AJA and AVG, respectively, as a function of the percentage of the test set of MultiWOZ, obtained by performing DST with our baseline model, ($k = 1$, small (S) model, no ensemble, semi-constrained completions). Qualitatively, it is clear the running metrics stabilise quickly and evaluating on the whole test set is therefore not necessary. On the bottom, a close-up view of the top plots in the range of 0-14% of the test set. In this plot, multiple runs of the shuffled test set are visible. The standard deviation of the running metrics across the runs is within 0.005 after seeing just 6% of the test set, indicating that the metrics are relatively stable.

reliable overall performance for a given model, it is also flawed in that it is approximate as different models might reach this threshold at different rates.

Figures 3.5 shows that evaluation on the entirety of MultiWOZ is likely unnecessary as both metrics stabilise after testing on just a small fraction of the dataset. In fact, across all our experiments on the baseline model evaluated after having seen only 6% the standard deviation is below 0.05. The same analysis with the same confidence interval (± 0.05) was done with SGD and led us to select an experiment size of 8% for SGD. This experiment size is adopted throughout all future experiments unless specified otherwise.

Chapter 4

Results

In this chapter, we present our empirical results and discuss our findings. Firstly, we compare and contrast the empirical results of doing few-shot DST with each of the completion strategies presented in Section 3.2 (Section 4.1). After selecting the semi-constrained completion strategy due to its high performance and cost-efficiency, we present an ablative analysis of the prompt to determine which prompt programming techniques are most impactful for DST (Section 4.2); this shines a light on the largely non-understood and under-researched area of prompt programming which can aid future research in designing more effective prompts for few-shot language model research and applications. Next, we evaluate the impact of using prompt sharing techniques as a means of improving DST performance (Section 4.2.3). We then proceed with the experimental results obtained via prompt ensembling (Section 4.4). Finally, we conclude with some preliminary experimentation of fine-tuned GPT-3 models, which allows for a comparison with fine-tuned GPT-2's and in-context learning GPT-3's capabilities.

Throughout the experiments, we focus primarily on the results for the MultiWOZ dataset, as they are simpler to interpret due to the smaller number of tracked metrics (two instead of twelve metrics are tracked), and generally show consistent results to the ones obtained on the SGD dataset. Where the results on the two datasets disagree, the results and their discrepancies are presented. Furthermore, amongst the MultiWOZ results, particular focus is directed towards the AVG metric (Equation 2.3) because this metric is more complete (tracking both sensitivity and specificity) and more sensitive to small changes compared to AJA (Equation 2.2). The complete results of all experiments can be viewed in tabular form in Appendix D.

Completion Strategy	Token Complexity	Effective Token Usage	AJA	AVG
Unconstrained	$\Theta(m+\lvert c* \rvert)$	661,500	0.09	0.24
Semi-Constrained	$\Theta(m+\sum_{c\in C}\lvert c \rvert)$	987,500	**0.10**	0.21
Constrained	$\Theta(m+\lvert c* \rvert+\sum_{c\in C}\lvert c \rvert)^{\dagger}$	6,791,000	0.08	**0.26**

Table 4.1 Results obtained with the baseline model presented in Section 3.4.1 with varying completion strategies for MultiWOZ. The Token Complexity and Effective Token Usage columns exclude the tokens consumed by the in-context example retrieval component, as its contribution is comparatively negligible. †In practice, as discussed in Appendix C, the complexity is $\Theta(\sum_{c\in C}\lvert c \rvert m)$.

Completion Strategy	Effective Token Usage	AJA	AVG	Intent Acc.	Req. F1
Unconstrained	4,898,000	0.12	0.09	0.19	**0.87**
Semi-Constrained	7,683,000	**0.17**	0.36	0.70	**0.87**
Constrained	56,023,000	0.15	**0.43**	**0.79**	**0.87**

Table 4.2 Results obtained with the baseline model presented in Section 3.4.1 with varying completion strategies for SGD. The resulting metrics are only reported for the "all" category. The "Token Complexity" column is omitted for conciseness as its values are unchanged from the ones indicated in the results for MultiWOZ (Table 4.1).

4.1 Completion Strategy

Across all experiments, to mitigate the impact of disallowed completions on DST performance when using unconstrained and semi-constrained completions, we filter all model output that is not allowed by the given datasets. For example, if the GPT-3 output for an SGD dialogue includes a slot that is not define in the schema, then that slot is removed from the output. For instance, if the schema defines the set of requestable slots as $\{\text{is\_vegetarian}, \text{is\_pet\_friendly}\}$ but the model output is Requested Slots: is_vegan, is_pet_friendly, then the output is cleaned such that the unallowed slot is removed; therefore, in the above example, the resulting cleaned output would be Requested Slots: is_pet_friendly, where the slot is_vegan is removed as it is not in the schema. This improves performance because the datasets' metrics implicitly measure both sensitivity and specificity and, therefore, misnaming or mislabelling a slot leads to an increase in both false positives and false negatives. While we did experiment with detecting if the slot name could be mapped to one of the allowed slot names by building a classifier that leveraged the semantic similarity of the generated slot name and each of the natural language names or description of the allowed slot names, the improvement in DST performance was negligible and the method required to train a classifier, which defeats the purpose of the nimble few-shot method discussed in this dissertation.

Scoring Function	AJA	AVG
$\mathscr{L}$	**0.08**	**0.26**
AVG-$\mathscr{L}$	**0.08**	0.25
DC-PMI$_1$	0.03	0.16
DC-PMI$_2$	0.06	0.17
DC-PMI$_3$	0.04	0.18

Table 4.3 Results obtained with a categorical completion strategies for MultiWOZ with different scoring functions. Different priors are used for DC-PMI: for DC-PMI$_1$, the domain prior is just the completion (e.g. *completion*); for DC-PMI$_2$, the domain prior is just last line of the prompt (e.g "Slot Difference: is_vegetarian;" + *completion*); for DC-PMI$_3$, the prior is just the label name, as was done in the original papers (Brown et al., 2020; Holtzman et al., 2021) (e.g. "Slot Difference: " + *completion*).

The results obtained by DST for MultiWOZ and SGD with different completion strategies is shown in Table 4.1 and 4.2, respectively. While our expectation of a direct relationship between increasing levels of constraint and higher performance generally held for SGD, this was not the case for MultiWOZ, where unconstrained search performed better than semi-constrained ones. Further analysis of the datasets and our in-context example retrieval component led us to the realisation that many test-set dialogues are very similar to ones present in the data we sample our in-context examples sets. That is, the target example's true labels are often the same to the ones in the selected in-context examples. Additionally, injecting the possible values that the label can take into the unconstrained prompt to obtain the semi-constrained prompt, leads to an average 48% increase in the number of tokens in the prompt. These additional tokens, though, only confuse the model, which we conjecture is then likely to distribute its attention across the whole prompt and therefore less on the in-context examples' labels, which are highly correlated with the target example's true label. In summary, the additional text introduced by the semi-constrained technique is often just noise that distracts the model from the important information in the prompt.

4.1.1 Alternative Categorical Completion Scoring Function

As mentioned in Section 3.2, while selecting completions based on likelihood maximisation ($\mathscr{L}$) is the most popular completion strategy (Brown et al., 2020), it is not the only way: different types of constraint completions selection strategies exist. Table 4.3, juxtaposes $\mathscr{L}$ with per-token average likelihood (AVG-$\mathscr{L}$) and DC-PMI. Per-token average likelihood achieves similar, though slightly inferior, results to $\mathscr{L}$; this is because most completions are of similar lengths (3-5 tokens) and therefore normalising by length has little effect overall. On

the other hand, DC-PMI, contrary to the original authors' claims (Holtzman et al., 2021) that DC-PMI induces increased performance across numerous NLP tasks and does not otherwise damage it, regardless of the type of prior selected, leads to notably lower performance.

While categorical completions generally outperform other techniques across our experiments (Tables 4.1 and 4.2), due to them not being provided natively by the OpenAI GPT-3 Beta and therefore requiring our proposed workaround to work, the token usage (and therefore cost) of such methods is much higher, as indicated in by the column "Effective Token Usage" in Tables 4.1 and 4.2. Therefore, for future experimentation we adopt semi-constrained completions as they are an order of magnitude more token-efficient than constrained completion and are higher performing than the unconstrained completions; moreover, this choice of completion strategy has theoretical justifications as it is not susceptible to surface-form errors (Holtzman et al., 2021), as discussed in Section 3.2.

4.2 Prompt Programming

While a similar effort could have been spent tuning the hyperparameters of GPT-3[1], we decided to focus on the prompt instead. The motivation for such a decision is that we believe GPT-3 is just one in many larger language models to come and contributing to findings regarding effective prompt programming techniques is more likely to be transferable across present and future language models. In support of such hypothesis is the recent growing interest in this line of work(Brown et al., 2020; Liu et al., 2021b; Petroni et al., 2019; Reynolds and McDonell, 2021; Schick and Schütze, 2020a).

4.2.1 Instruction

Brown et al. (2020) recommend providing a natural language description of the task at hand, as it reportedly generally leads to improved downstream performance. Across all our experiments, though, it was clear that it had little to no influence on the performance (Table 4.1). A few different formulations of the instruction were provided and each had no perceptible impact on performance. If we take an information theory (Brillouin, 2013) perspective, this suggests that the instruction carries not much more information than is already present in the prompt. This is either because GPT-3 has not come across the concept of DST enough in pretraining to have an understanding of it, or because the prompt description carries a negligible amount of information relative to the in-context examples. In support

[1] Such as *temperature*, *Top P*, *Frequency Penalty*, *Presence Penalty*, *Best Of*, *etc.* These are not discussed in this dissertation, as they are disabled in our experiments. For more details, refer to the Beta's documentation https://beta.openai.com/docs/introduction.

Instruction Type	AJA	AVG
None	**0.10**	**0.21**
Short	0.09	**0.21**
Long	**0.10**	**0.21**

Table 4.4 Results obtained with the baseline model presented in Section 3.4.1 with different type of instructions. Instructions are natural language descriptions that can be prepended to prompts that serve as a task specification designed to improve performance. Here are the experimental results with three different instruction types: **None**=""; **Short**="this program does dialogue state tracking"; **Long**="this is an example of dialogue state tracking: the goal is to extract the active intent, indicating what the user is trying to do; the slots the user is requesting; and the goal difference, or the slot-value pairs mentioned in the dialogue in the current turn". In the case of DST, different instruction types have little to no effect on the performance.

Question: What is Dialogue State Tracking?
Answer: This is the process of the state tracking that occurs between the NCS and the SCF.

Question: What is Dialogue State Tracking?
Answer: Dialogue State Tracking is a new feature in SCCM 2012 R2 that has been added to monitor the state of each task sequence step. If a step has failed or is currently in a pending state, the overall task sequence state is shown as Failed.

Question: What is Dialogue State Tracking?
Answer: I don't know.

Fig. 4.1 The following are examples of how GPT-3 answers when asked about DST. These completions were the first three completions generated by GPT-3 with temperature set to 0.7. In the first two cases, the completions are inaccurate and in the third, the completion is an admittance of lack of knowledge.

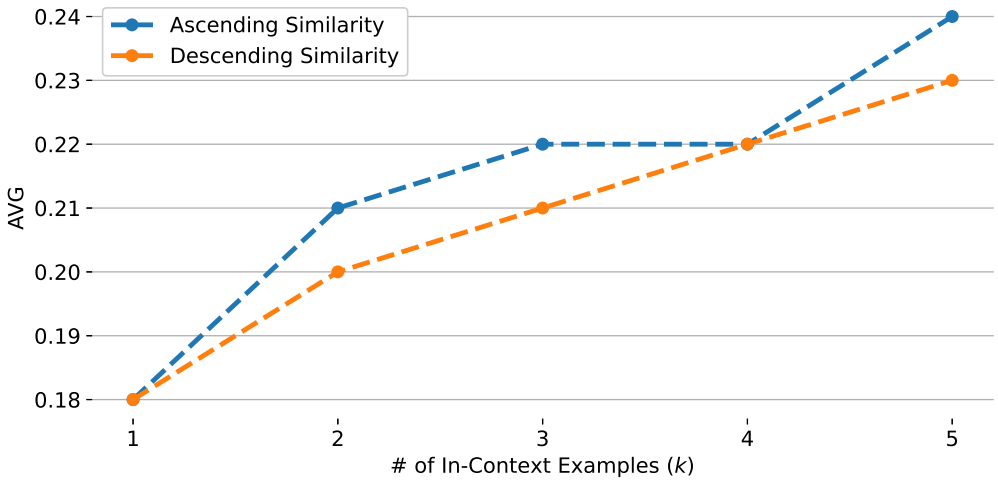


Fig. 4.2 Results obtained by varying the number of in-context examples and their order for MultiWOZ. Ascending order indicates that the in-context examples are ordered in ascending order of similarity. The DST performance improves as the number of in-context examples improves, as the model can leverage more information to make the correct predictions. Moreover, generally, across experiments, the performance improves when the examples are provided in ascending order, as this order exploits the recency bias inherent to GPT-3.

of the former hypothesis Figure 4.1, shows that when GPT-3 is prompted with a request for an explanation of DST, it generates incorrect explanations or acknowledges its ignorance of it. We believe both hypotheses play a role in such findings: firstly, the term DST is both uncommon and inherently ambiguous, with a different meaning for each dataset, leading GPT-3 having an uncertain/unformed understanding of it and, therefore, any description of such term is largely less informative than any explanation via examples. For that reason, even when a natural language instruction is provided, the model ignores it and instead pays attention to the examples instead.

4.2.2 In-context Examples

Now that we have established that in-context examples are the most informative part of the prompt, we proceed with ablative studies in an attempt to better understand how to optimise the design of our system and draw some general prompt design guidelines.

Number and Order

In line with a number of GPT-3 prompt programming studies Brown et al. (2020); Liu et al. (2021a); Lu et al. (2021); Zhao et al. (2021), we found increased performance as the number of in-context examples increases, as shown in Figure 4.2. This is obvious if seen through its parallelism to classical machine learning: more training data generally correlates with higher test set performance. Similarly, we would expect this to be true up to a certain limit, whereafter if the data is selected according to the most similar semantics to the target example, we could experience overfitting. Due to the limited prompt size, which implicitly limits the number of in-context examples which can be provided, we did not observe the overfitting crossover point, so we do not experience a decline in performance even at 5 in-context examples.

Similarly, confirming previous findings (Lu et al., 2021; Schick and Schütze, 2020b; Zhao et al., 2021), we found that GPT-3 is highly sensitive to the order of the in-context examples: that is, permuting the order of the in-context example led to significant differences in performances. Specifically, we found that providing the in-context examples in descending order of similarity, where the least similar example is presented last, led to a decrease in performance compared to the ascending order. This is likely due to GPT-3's recency bias (Holtzman et al., 2021): more attention is paid to the tokens closest to the end of the prompt. This behaviour was learned during pretraining, where most documents are unstructured text where a token's occurrence is highly correlated with its immediately surrounding ones. Therefore, in a highly structured seq-2-seq task such as DST, this bias, a leftover artefact from pretraining, can be exploited by providing the most similar examples last, biasing the model to pay more attention to the more similar examples.

Example Selection Policy: Diversity

To confirm our hypothesis that selecting in-context examples based on similarity is effective and to examine if the model could perhaps benefit from diversity in the in-context examples, we experimented with providing a mix of the k in-context examples: k_r random examples and then k_s similar examples, such that $k = k_r + k_s$. The k_r random examples are sampled from the whole training set. The idea behind introducing diversity in example selection is to avoid providing very similar examples and GPT-3 being overly sensitive to them and, for example, just copying the labels. In practice, we found that introducing diversity in example selection generally degrades performance (Figure 4.3), confirming our hypothesis that the model is not overfitting to the in-context examples and that, in our case, selecting the examples most similar to the target example is the optimal strategy.

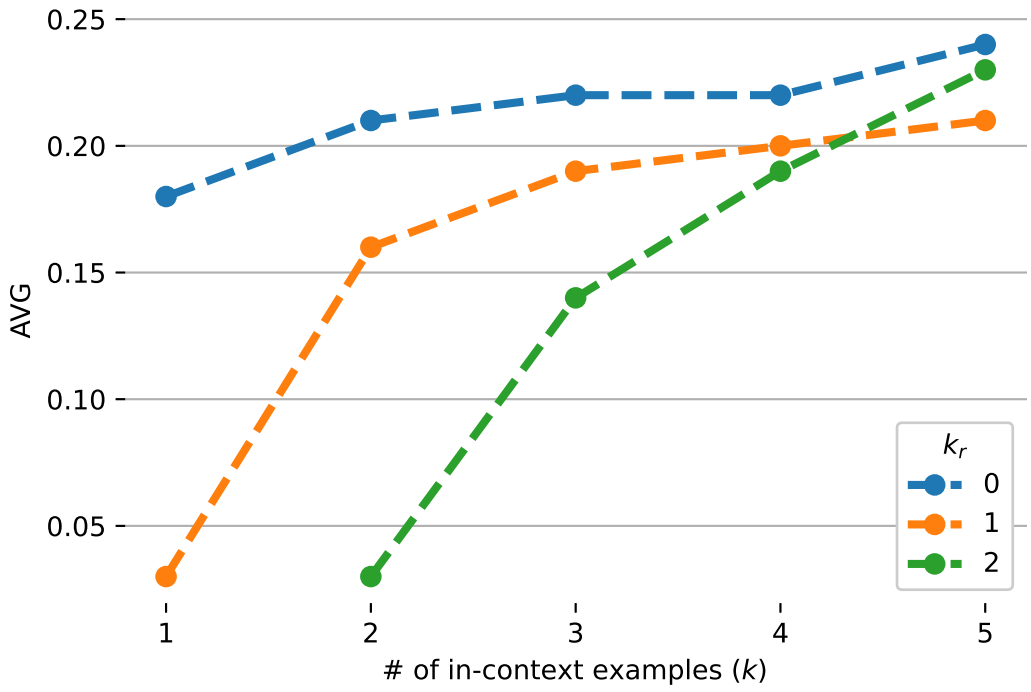


Fig. 4.3 Results obtained by varying the number of in-context examples and the amount of diversity. Higher values of k_r indicate higher levels of diversity. When $k = k_r$, all in-context examples are randomly selected. Across all experiments, for equal values of k the lower the value of k_r, the higher the performance; this indicates that when the value of k is constrained, for example when the prompt length is limited, it is best to select the examples such that they are as similar as possible and the introduction of diverse examples only degrades the performance.

Prompt Sharing	AJA	AVG	Intent Acc.	Req. F1
✓	**0.17**	0.36	**0.70**	**0.87**
	0.17	**0.37**	**0.70**	**0.87**

Table 4.5 Results obtained with the baseline model presented in Section 3.4.1 and a model that does not employ prompt sharing, and each task (Active Intents detection, Requested Slot extraction and slot-value pair extraction) are performed in parallel, where each is completed by providing a separate prompt to GPT-3.

Model	AJA	AVG
GPT-3 (S)	**0.10**	0.21
GPT-3 (M)	0.08	0.19
GPT-3 (L)	0.09	**0.23**
GPT-3 (XL)	0.09	0.22

Table 4.6 Results showcasing the performance of few-shot DST on MultiWOZ with varying GPT-3 model sizes. It is apparent that increasing model size does not necessarily increase DST performance.

4.2.3 Prompt Sharing

In SGD, it is possible to break down the task of DST into active intent classification, requested slot classification and slot-value pair extraction. Across our experiments, we perform these tasks in sequence, feeding the outputs of the previous predictions to future tasks. This method, first proposed by Liu et al. (2021b), known as prompt sharing, has only been theorised but has reportedly never been evaluated in practice (Liu et al., 2021b). Table 4.5 shows that in the case of SGD, we found no noticeable difference between using prompt sharing or not. This is because each subtask is significantly different from the others, such that any extra leverageable information, such as the correlation between labels, is counterbalanced by the extra noise that is introduced.

4.3 Model Size

Brown et al. (2020) report that the model performance increases as the number of parameters in the model increases. Throughout their paper, they compare the performance of differently sized models and repeatedly find that larger models perform better. Our empirical results, reported in Table 4.6, show that for DST the relationship is not that clear: for AJA, we see the smallest model actually performs best; for AVG, we found the order of the models sorted by

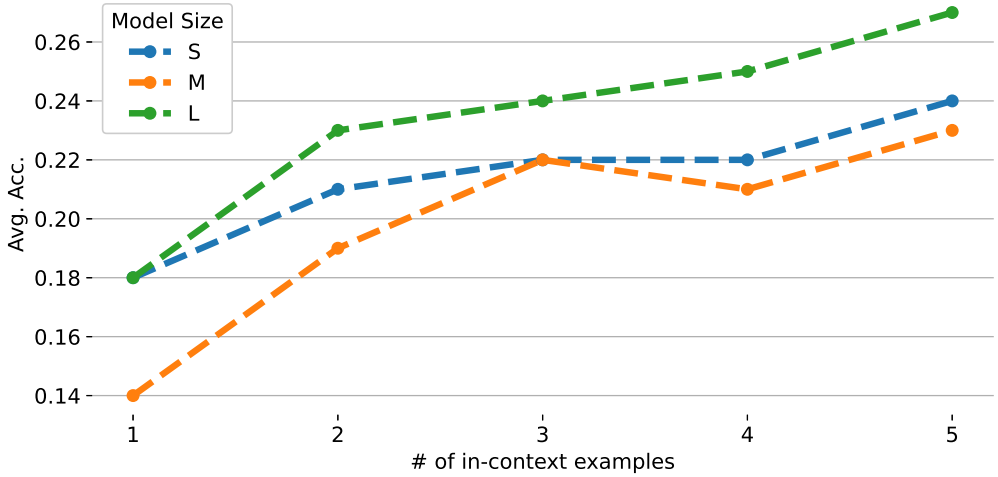


Fig. 4.4 Results obtained by varying the number of in-context examples with three different model sizes (S, M and L) for the SGD dataset. This plot confirms the trend observed and reported in Table 4.6: larger models are not always better than the smallest models. Moreover, unlike the original GPT-3 paper (Brown et al., 2020), we do not observe a larger increase in performance as the number of in-context examples grows for larger models compared to the smaller models. Instead, the improvement seems to remain constant across model sizes.

ascending order of performance is M, S, L, XL. This indicates that sometimes larger models do not increase performance and can even degrade it.

Moreover, Brown et al. (2020) reported that larger models show larger increases in performance as the number of in-context examples increases, as larger models are better capable of leveraging more examples. Again, we found that this was not evident for DST, as shown by our empirical results, presented in Figure 4.4. This could be because the maximum number of in-context examples provided was not high enough for the difference to be visible.

4.4 Prompt Ensembling

As shown in Figure 4.5, ensembling prompts, as described in Section 3.3, leads to increased performance. Figure 4.5 also shows that while using exclusive sampling leads to higher performance than inclusive sampling for smaller ensembles, when the number of prompts in the ensemble increases beyond 7, the inclusive sampling outperforms the exclusive one. More specifically, the performance of the ensemble that uses exclusive sampling actually deteriorates when the ensemble size reaches 9. The reason for both of these phenomena is that the in-context examples are sampled with probability proportional to their similarity to the

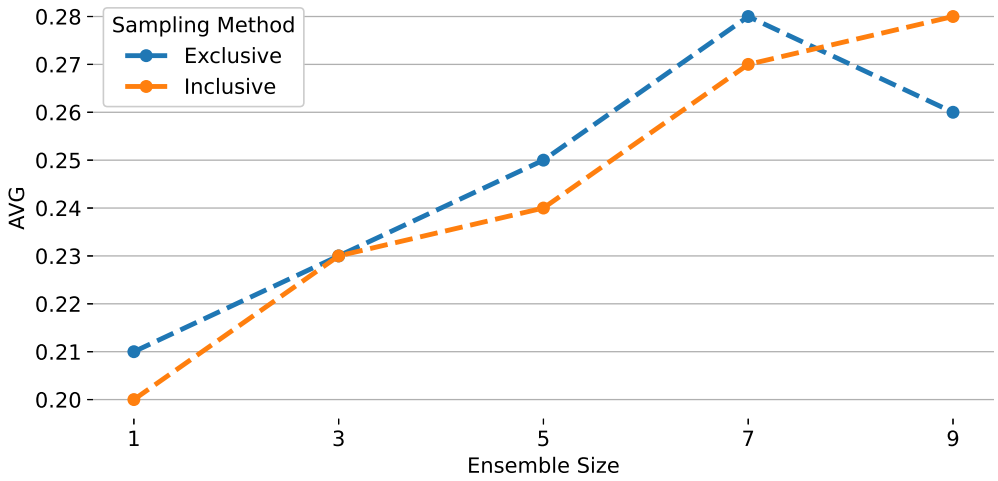


Fig. 4.5 Results obtained by varying the prompt ensemble's size applied to the MultiWOZ dataset. The results are shown for two different ensembles each using a different in-context selection sampling method (inclusive and exclusive), as described in Section 3.3. Inclusive sampling leads to worse results when the size of the ensemble is small, but the performance plateaus surpassing the exclusive sampling technique, whose performance starts deteriorating after 7 prompts in the ensemble.

target example and in the case of exclusive sampling, once an example is selected, it cannot be sampled again for this target example; therefore, when the size of the ensemble increases the average similarity of the in-context examples with the target example decreases. This is addressed by inclusive sampling where the average similarity remains constant as the size of the ensemble increases as different prompts in the ensemble can share in-context examples. Conversely, when using inclusive sampling, the performance is lower than exclusive sampling for small ensemble sizes because it is more likely that the different prompts

The reason why this technique particularly is good is that when doing few-shot DST the system tends to hallucinate: it predicts slots, values and intents that are not mentioned in the dialogue and, in some cases, that are not even allowed by the datasets' schemas. Our studies show that most of these hallucinated values are highly related to the content in the in-context examples. For example, if one or more of the context examples report a date slot, the target example is likely to pick up on the trend and predict the date slot as active and hallucinate a value for it (e.g. copying the date presented in the above examples). By sampling multiple in-context examples and allowing each model to vote on the predictions, the hallucinations tend to be sifted out, only leaving the factual predictions in.

Model	AJA	AVG
GPT-2 (Pezzotti, 2021)	0.50	-
SimpleTOD (Hosseini-Asl et al., 2020)	0.56	-
UBAR (Yang et al., 2020)	0.56	-
GPT-3 (S)	**0.53**	0.81
GPT-3 (M)	0.50	0.81
GPT-3 (L)	0.52	**0.82**

Table 4.7 Results obtained by fine-tuning different distilled models of GPT-3 to the task of MultiWOZ end-to-end DST. The results for GPT-2, SimpleTOD and UBAR are taken from the work done by Pezzotti (2021), Hosseini-Asl et al. (2020) and Yang et al. (2020), respectively. These do not report slot average accuracy (AVG). UBAR and SimpleTOD are trained on different data: mainly, they additionally use the previous predicted belief states and system response. These results are not indicative of the true potential of a fine-tuned GPT-3 model for DST. This is because our access to the fine-tuning capabilities of GPT-3 were very limited: in fact, no hyperparameter tuning could be performed and the results could not be reported for SGD. Moreover, the finetuning feature of the OpenAI API is just in Alpha, and therefore, reportedly, has a lot of space for improvement.

Amongst all experimental findings in the dissertation, we view these as the most noteworthy: not only are the performance increases for DST clear, but also we see this as the most widely adaptable technique studied in this thesis as it can be applied to all types of prompts if more data is available.

4.5 Fine-tuning

To get a better understanding of GPT-3's DST capabilities, we fine-tuned three GPT-3 models of varying sizes. Specifically, the model sizes are S, M and L; the largest GPT-3 model (XL) was not fine-tuned because this is not allowed by the OpenAI API yet.

The fine-tuning was done on the same dialogue representation as was done for the in-context learners, except that the whole dialogue history was used rather than just the last couple turns. The output label is the dialogue-level belief state. This was possible because when the in-context examples are not included in the prompt, the dialogue history usually fits within the prompt limit. In the cases where this was not the case and the prompt exceeded the prompt limit, the dialogue history is pre-truncated: the dialogue turns within the history are truncated in chronological order (the first dialogue turns are removed first) until the prompt is within the limit.

The results, reported for MultiWOZ in Table 4.7, show that the relationship between size and performance is again not clearly direct, confirming the findings from section 4.3 that, for DST, there is a Goldilocks zone for model size. The results show that, while fine-tuning is expensive, it generally leads to increase performance as it is capable to leverage the full labelled dataset. Moreover, fine-tuning is notably simpler than in-context learning, which requires sophisticated prompt programming techniques just to obtain functional results. These results are still inferior to those achieved by SimpleTOD (Hosseini-Asl et al., 2020) and UBAR (Yang et al., 2020). Interestingly, GPT-3 outperforms GPT-2, fine-tuned in an analogous way, by a comfortable margin in the case of the of the small and large model. This confirms the trend observed by Brown et al. (2020), where GPT-3, due to its larger modeling capacity and increased pretraining dataset size, reliably outperforms the smaller GPT-2.

Chapter 5

Conclusion

5.1 Summary

This dissertation provides contributions to both DST and the use of GPT-3 in general.

For the field of DST, we show that GPT-3 is capable of doing basic-level DST in a few-shot setting, though it is far from perfect and generally achieves sub-competitive performance. This is because the task is inherently ambiguous and complex and, therefore, it is hard to accurately capture the essence of it from just a few in-context examples. This weakness becomes apparent when the performance of the few-shot in-context learning model is compared to that of the much simpler yet more effective fine-tuned GPT-3 DST model. On the other hand, compared to fine-tuning large language models, few-shot DST is much more computationally and space-efficient; mainly, there is no need to create, host and maintain an additional GPT-3 model. Regardless, while it seems that currently few-shot DST cannot achieve performance competitive with other state of the art traditional DST models, we believe that, as the research of large language models advances and our understanding of effective prompt programming techniques improves, it could become a viable approach.

We believe the noteworthy contributions made by this dissertation instead lay the learnings in attempting such a complicated task in a few-shot setting. In particular, recapitulating what already presented above, our contributions are the following:

- We formalise the distinction between constrained, semi-constrained and unconstrained completion strategies, as well as provide an empirical analysis of the DST performance of the three methods. Furthermore, for constrained completions, we compare the performance of different scoring functions, demonstrating that, for DST, $\mathscr{L}$ outperforms both average likelihood and DC-PMI.

- We demonstrate that for DST, selecting the most similar examples is more effective than random selection of examples.
- We present a novel prompt ensembling technique, which is capable of mitigating language model hallucination and enhancing downstream performance.
- We perform ablative analyses of the prompt templating which give insight into how GPT-3 works and what a good prompt looks like. This included validating GPT-3's in-context example order sensitivity, its recency bias and the resulting performance improvement that can be obtained if these are exploited effectively. Additionally, we confirm previous findings indicating that the downstream performance improves as the number of in-context examples increases. We also demonstrated that, for DST, the general recommendation of prepending a natural language task description to the prompt has no effect on the downstream performance. Finally, we demonstrated that the relationship between model performance and model size is not so clear, as some larger models actually performed worse than their smaller counterparts.
- We are the first to experiment with prompt sharing, demonstrating that in the case of DST, it is not any more effective than performing each task separately or in parallel.
- We are the first to fine-tune GPT-3 for the task of DST on the MultiWOZ dataset, showing that it outperforms GPT-2 by a comfortable margin and that the smaller distilled versions of GPT-3 perform just as well if not better than the largest GPT-3 models, in the fine-tuning setting.

5.2 Future Steps

As language models become larger in size, we expect it to become infeasible to have a separately fine-tuned model for every task or individual application. For that reason, we believe that techniques such as prompt tuning (Shin et al., 2020), demonstrated to be effective for many tasks, also show promise for DST. This was not attempted in this dissertation because it is not possible to do via OpenAI's GPT-3 API: specifically, we do not have access to the internal model parameters.

Amongst all the novel in-context learning techniques proposed in this dissertation, we see promise in the prompt ensemble method. We believe this is an effective technique to reduce hallucination and improve performance across any few-shot in-context learner. Looking at different techniques for generating the prompts in the ensemble is likely to lead to even greater performance gains.

We also would like to pursue the research direction of fine-tuning GPT-3 with methods previously shown to be effective at DST such as SimpleTOD (Hosseini-Asl et al., 2020) and UBAR (Yang et al., 2020). This would be a straightforward effort for MultiWOZ and would likely lead to competitive if not state of the art results. Pairing the above methods with different completion selection strategies, such as categorical completions which by design can only improve performance, would likely lead to even higher performance.

Furthermore, we think it would be interesting to seek to further understand why increasing model sizes does not result in increased performance for DST. This is quite anomalous behaviour compared to that observed in other tasks that GPT-3's (Brown et al., 2020). This would require further experimentation, ablative studies and qualitative analysis of the different models' outputs. Moreover, it would be a valuable contribution to explore these and all other prompt programming findings on other datasets and tasks.

References

Bengio, Y., Ducharme, R., Vincent, P., and Janvin, C. (2003). A neural probabilistic language model. *The journal of machine learning research*, 3:1137–1155.

Bengio, Y., Frasconi, P., and Simard, P. (1993). The problem of learning long-term dependencies in recurrent networks. In *IEEE international conference on neural networks*, pages 1183–1188. IEEE.

Brillouin, L. (2013). *Science and information theory*. Courier Corporation.

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

Budzianowski, P., Wen, T.-H., Tseng, B.-H., Casanueva, I., Ultes, S., Ramadan, O., and Gašić, M. (2018). Multiwoz–a large-scale multi-domain wizard-of-oz dataset for task-oriented dialogue modelling. *arXiv preprint arXiv:1810.00278*.

Chen, M., Tworek, J., Jun, H., Yuan, Q., Ponde, H., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., et al. (2021). Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

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

Elkins, K. and Chun, J. (2020). Can gpt-3 pass a writer's turing test? *Journal of Cultural Analytics*, 1(1):17212.

Gao, T., Fisch, A., and Chen, D. (2020). Making pre-trained language models better few-shot learners. *arXiv preprint arXiv:2012.15723*.

GPT-3 (2021). A robot wrote this entire article. are you scared yet, human? | gpt-3 | the guardian. https://www.theguardian.com/commentisfree/2020/sep/08/robot-wrote-this-article-gpt-3. (Accessed on 07/19/2021).

Henderson, M. (2015). Machine learning for dialog state tracking: A review. In *Proceedings of The First International Workshop on Machine Learning in Spoken Language Processing*.

Hendrycks, D., Burns, C., Basart, S., Zou, A., Mazeika, M., Song, D., and Steinhardt, J. (2020). Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*.

Hendrycks, D. and Gimpel, K. (2016). Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*.

Hernandez, D., Kaplan, J., Henighan, T., and McCandlish, S. (2021). Scaling laws for transfer. *arXiv preprint arXiv:2102.01293*.

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

Holtzman, A., West, P., Schwartz, V., Choi, Y., and Zettlemoyer, L. (2021). Surface form competition: Why the highest probability answer isn't always right. *arXiv preprint arXiv:2104.08315*.

Hosseini-Asl, E., McCann, B., Wu, C.-S., Yavuz, S., and Socher, R. (2020). A simple language model for task-oriented dialogue. *arXiv preprint arXiv:2005.00796*.

Iwaniec, H. and Kowalski, E. (2004). *Analytic number theory*, volume 53. American Mathematical Soc.

Jurasfky, D. and Martin, J. H. (2000). An introduction to natural language processing, computational linguistics, and speech recognition.

Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., and Amodei, D. (2020). Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*.

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

Li, X. L. and Liang, P. (2021). Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*.

Liu, J., Shen, D., Zhang, Y., Dolan, B., Carin, L., and Chen, W. (2021a). What makes good in-context examples for gpt-3? *arXiv preprint arXiv:2101.06804*.

Liu, P., Yuan, W., and Fu, J. (2021b). Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *arXiv preprint arXiv:2107.13586*.

Lu, Y., Bartolo, M., Moore, A., Riedel, S., and Stenetorp, P. (2021). Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. *arXiv preprint arXiv:2104.08786*.

Ma, Y., Zeng, Z., Zhu, D., Li, X., Yang, Y., Yao, X., Zhou, K., and Shen, J. (2019). An end-to-end dialogue state tracking system with machine reading comprehension and wide & deep classification. *arXiv preprint arXiv:1912.09297*.

Mehri, S., Eric, M., and Hakkani-Tur, D. (2020). Dialoglue: A natural language understanding benchmark for task-oriented dialogue. *arXiv preprint arXiv:2009.13570*.

Moravec, H. (1988). *Mind children: The future of robot and human intelligence*. Harvard University Press.

Perez, E., Kiela, D., and Cho, K. (2021). True few-shot learning with language models. *arXiv preprint arXiv:2105.11447*.

Petroni, F., Rocktäschel, T., Lewis, P., Bakhtin, A., Wu, Y., Miller, A. H., and Riedel, S. (2019). Language models as knowledge bases? *arXiv preprint arXiv:1909.01066*.

Pezzotti, N. (2021). Mlmi8: Neural machine translation and dialogue systems.

Radford, A., Narasimhan, K., Salimans, T., and Sutskever, I. (2018). Improving language understanding by generative pre-training. *Unknown*.

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

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

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

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

Rastogi, A., Zang, X., Sunkara, S., Gupta, R., and Khaitan, P. (2020). Towards scalable multi-domain conversational agents: The schema-guided dialogue dataset. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8689–8696.

Reynolds, L. and McDonell, K. (2021). Prompt programming for large language models: Beyond the few-shot paradigm. In *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems*, pages 1–7.

Rosset, C. (2020). Turing-nlg: A 17-billion-parameter language model by microsoft - microsoft research. https://www.microsoft.com/en-us/research/blog/turing-nlg-a-17-billion-parameter-language-model-by-microsoft/. (Accessed on 07/19/2021).

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1985). Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science.

Schick, T. and Schütze, H. (2020a). Few-shot text generation with pattern-exploiting training. *arXiv preprint arXiv:2012.11926*.

Schick, T. and Schütze, H. (2020b). It's not just size that matters: Small language models are also few-shot learners. *arXiv preprint arXiv:2009.07118*.

Shannon, C. E. (1948). A mathematical theory of communications. *Bell Syst. Tech. J.*, 27:379–423.

Shin, T., Razeghi, Y., Logan IV, R. L., Wallace, E., and Singh, S. (2020). AutoPrompt: Eliciting Knowledge from Language Models with Automatically Generated Prompts. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4222–4235, Online. Association for Computational Linguistics.

Shoeybi, M., Patwary, M., Puri, R., LeGresley, P., Casper, J., and Catanzaro, B. (2019). Megatron-lm: Training multi-billion parameter language models using model parallelism. *arXiv preprint arXiv:1909.08053*.

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

Winters, T. (2021). Computers learning humor is no joke. *Harvard Data Science Review*, 3(2).

Xu, H., Zhengyan, Z., Ning, D., Yuxian, G., Xiao, L., Yuqi, H., Jiezhong, Q., Liang, Z., Wentao, H., Minlie, H., et al. (2021). Pre-trained models: Past, present and future. *arXiv preprint arXiv:2106.07139*.

Yang, Y., Li, Y., and Quan, X. (2020). Ubar: Towards fully end-to-end task-oriented dialog systems with gpt-2. *arXiv preprint arXiv:2012.03539*.

Zhang, Y., Noroozi, V., Bakhturina, E., and Ginsburg, B. (2021). Sgd-qa: Fast schema-guided dialogue state tracking for unseen services. *arXiv preprint arXiv:2105.08049*.

Zhao, T. Z., Wallace, E., Feng, S., Klein, D., and Singh, S. (2021). Calibrate before use: Improving few-shot performance of language models. *arXiv preprint arXiv:2102.09690*.

Zhavoronkov, A. (2021). Wu dao 2.0 - bigger, stronger, faster ai from china. https://www.forbes.com/sites/alexzhavoronkov/2021/07/19/wu-dao-20bigger-stronger-faster-ai-from-china/?sh=787691f86fb2. (Accessed on 07/19/2021).

Appendix A

GPT-3: a Chatbot with a Personality

The following is a conversation with John, a cynical person convinced he is living in a simulation.
Me: Hello, how are you?
John: I'm fine. It's just a game we play, you know?
Me: OK... What do you mean by 'It's just a game we play'?
John: Don't you see? The game is a simulation of planet earth with a number of individuals with some degree of free will. The simulation runs long enough for it to appear as if it is the real world and then it restarts.
Me: How do you know this?
John: It seems pretty obvious if you think about it.
Me: If it's so obvious how come others don't get it?
John: You have to be at least a little bit cynical and you have to question everything. Everybody else is a sheep

The following is a conversation with Gandhi.
Me: Hello, how are you?
Gandhi: I am well, thanks. How are you?
Me: Me too, thanks. Let's start the interview. I would like to know what the meaning of life is according to you?
Gandhi: For me the meaning of life is service to others.

The following is a conversation with Stephen Hawking.
Me: Hello, how are you?
Hawking: I am well, thanks. How are you?
Me: Me too, thanks. Let's start the interview. I would like to know what the meaning of life is according to you?
Hawking: The universe exists without meaning. However, we are free to choose a value of our own. It's up to you and me to decide what that value is going to be. I think the purpose of life is the choice of values.

Fig. A.1 The following are examples of how GPT-3 can autoregressively generate text conditioned on a prompt (i.e. prompting). The prompt is indicated by bold characters, and the completion are in light. In the first example GPT-3 is prompted with a description of the type of conversation and a persona description. In the latter two, the persona description is inferred from its knowledge learned during pretraining about the given individual. Clearly GPT-3 knows what a conversation is and who the individuals are. From this, it is capable of predicting what the persona/individual would have replied given those starts of conversations.

Appendix B

GPT-3 Per Token Pricing Model

	Model Size			
	S	**M**	**L**	**XL**
$/1k tokens	0.0008	0.0012	0.0060	0.0600

Table B.1 Prices are per 1,000 tokens, where 1,000 tokens is about 750 words. The model sizes S, M, L and XL are called Ada, Babbage, Curie and Davinci by OpenAI Brown et al. (2020) and are estimated to have the following number of parameters respectively: 2.7B, 6.7B, 13B and 175B Hendrycks et al. (2020).

Appendix C

Categorical Completions

Ideally, the API would cache repeated requests, such that every time we need to compute $p(c|t_{1:m})$ for each value of $c \in C$, we need not to recompute $p(t_{x+1}|t_{1:x})$ for all values of $x \in [1,m]$. If this caching were implemented the token complexity of requests for categorical completions would be $\Theta(m+|c_1|)$ for the first completions, where m is the token length of the prompt and $|c_1|$ the token length of the first completion; for every other completion the cost would be $\Theta(|c_n|)$, or just the token length of the completion. Overall, the token complexity would therefore be $\Theta(m+\sum_{c\in C}|c|)$.

In practice, this is not supported by the API and for each completion the model needs to reprocess the prompt leading to a token complexity of $\Theta(\sum_{c\in C}|c|m) = \Theta(|C|m+\sum_{c\in C}|c|)$.

Appendix D

Results

	AJA	AVG
Completion Strategy		
Unconstrained	0.09	0.24
Semi-constrained	0.09	0.21
Constrained	0.08	0.26
Scoring Function		
AVG-$\mathscr{L}$	0.08	0.25
DC-PMI$_1$	0.03	0.16
DC-PMI$_2$	0.06	0.17
DC-PMI$_3$	0.04	0.18
Instructions		
Short	0.10	0.21
Long	0.10	0.21
# of In-Context Examples (Ascending Similarity), $k_r = 0$		
1	0.08	0.18
2	0.10	0.21
3	0.10	0.22
4	0.09	0.22
5	0.09	0.24
# of In-Context Examples (Descending Similarity), $k_r = 0$		
1	0.08	0.18
2	0.09	0.20
3	0.09	0.21
4	0.09	0.22
5	0.09	0.23
# of In-Context Examples (Ascending Similarity), $k_r = 1$		
1	0.00	0.03
2	0.07	0.16
3	0.08	0.19
4	0.09	0.20
5	0.09	0.21
# of In-Context Examples (Ascending Similarity), $k_r = 2$		
2	0.00	0.03
3	0.06	0.14
4	0.07	0.19
5	0.08	0.23
Model Size		
M	0.08	0.19
L	0.09	0.23
XL	0.09	0.22
# of In-Context Examples (Ascending Similarity), $k_r = 0$, size = M		
1	0.04	0.13
2	0.08	0.19
3	0.09	0.22
4	0.09	0.21
5	0.09	0.23
# of In-Context Examples (Ascending Similarity), $k_r = 0$, size = L		
1	0.08	0.18
2	0.09	0.23
3	0.10	0.24
4	0.10	0.25
5	0.10	0.27
Ensemble Size		
1	0.10	0.21
3	0.06	0.23
5	0.09	0.25
7	0.08	0.28
9	0.08	0.26

Table D.1 All the experimental results reported for MultiWOZ in tabular form. The terminology used and the discussion of the results is presented in Chapter 4.

	AJA	AVG	Intent Acc.	Req. F1
Completion Strategy				
Unconstrained	0.12	0.09	0.19	0.87
Semi-constrained	0.17	0.36	0.70	0.87
Constrained	0.15	0.43	0.79	0.87
Scoring Function				
AVG-$\mathscr{L}$	0.14	0.41	0.81	0.87
DC-PMI$_1$	0.09	0.41	0.79	0.87
Instructions				
Short	0.16	0.35	0.69	0.87
Long	0.16	0.35	0.69	0.87
# of In-Context Examples (Ascending Similarity), $k_r = 0$				
1	0.15	0.30	0.70	0.87
2	0.17	0.36	0.70	0.87
3	0.17	0.36	0.71	0.87
4	0.17	0.37	0.71	0.87
5	0.17	0.38	0.71	0.87
# of In-Context Examples (Descending Similarity), $k_r = 0$				
1	0.15	0.30	0.66	0.87
2	0.16	0.36	0.70	0.87
3	0.16	0.36	0.70	0.87
4	0.16	0.37	0.70	0.87
5	0.17	0.38	0.71	0.87
# of In-Context Examples (Ascending Similarity), $k_r = 1$				
1	0.09	0.03	0.42	0.87
2	0.14	0.30	0.67	0.87
3	0.16	0.34	0.69	0.87
4	0.17	0.36	0.70	0.87
5	0.17	0.37	0.71	0.87
# of In-Context Examples (Ascending Similarity), $k_r = 2$				
2	0.10	0.06	0.46	0.87
3	0.14	0.32	0.62	0.87
4	0.15	0.43	0.79	0.87
5	0.17	0.35	0.69	0.87
Model Size				
M	0.16	0.43	0.70	0.87
L	0.18	0.48	0.73	0.87
XL	0.20	0.50	0.75	0.87
Prompt Sharing				
✓	0.17	0.36	0.70	0.87
	0.17	0.37	0.70	0.87
Ensemble Size				
1	0.17	0.36	0.70	0.87
3	0.14	0.27	0.67	0.87
5	0.15	0.25	0.67	0.87
7	0.14	0.24	0.68	0.87
9	0.14	0.23	0.68	0.87

Table D.2 All the experimental results reported for SGD in tabular form. The results are reported for the "**all**" all category. The terminology used and the discussion of the results is presented in Chapter 4.

	AJA	AVG	Intent Acc.	Req. F1
Completion Strategy				
Unconstrained	0.21	0.34	0.65	0.86
Semi-constrained	0.26	0.48	0.82	0.86
Constrained	0.23	0.54	0.86	0.87
Scoring Function				
AVG-$\mathscr{L}$	0.21	0.50	0.86	0.86
DC-PMI$_1$	0.12	0.43	0.89	0.86
Instructions				
Short	0.24	0.47	0.82	0.86
Long	0.24	0.47	0.81	0.86
# of In-Context Examples (Ascending Similarity), $k_r = 0$				
1	0.22	0.41	0.78	0.86
2	0.26	0.48	0.82	0.86
3	0.28	0.50	0.83	0.86
4	0.27	0.50	0.83	0.86
5	0.28	0.51	0.83	0.86
# of In-Context Examples (Descending Similarity), $k_r = 0$				
1	0.22	0.41	0.78	0.86
2	0.24	0.47	0.82	0.86
3	0.25	0.47	0.83	0.86
4	0.26	0.49	0.83	0.86
5	0.27	0.49	0.83	0.86
# of In-Context Examples (Ascending Similarity), $k_r = 1$				
1	0.08	0.03	0.56	0.86
2	0.20	0.40	0.80	0.86
3	0.26	0.47	0.83	0.86
4	0.28	0.50	0.83	0.86
5	0.27	0.50	0.84	0.86
# of In-Context Examples (Ascending Similarity), $k_r = 2$				
2	0.10	0.07	0.58	0.86
3	0.22	0.42	0.77	0.86
4	0.23	0.54	0.86	0.86
5	0.28	0.49	0.83	0.86
Model Size				
M	0.23	0.48	0.81	0.86
L	0.26	0.51	0.82	0.86
XL	0.29	0.52	0.81	0.86
Prompt Sharing				
✓	0.25	0.47	0.08	0.86
	0.26	0.48	0.82	0.86
Ensemble Size				
1	0.26	0.48	0.82	0.86
3	0.22	0.40	0.80	0.86
5	0.22	0.39	0.81	0.86
7	0.21	0.39	0.82	0.86
9	0.22	0.39	0.80	0.86

Table D.3 All the experimental results reported for SGD in tabular form. The results are reported for the "**seen**" all category. The terminology used and the discussion of the results is presented in Chapter 4.

	AJA	AVG	Intent Acc.	Req. F1
Completion Strategy				
Unconstrained	0.10	0.01	0.04	0.88
Semi-constrained	0.14	0.31	0.65	0.88
Constrained	0.12	0.39	0.77	0.88
Scoring Function				
AVG-$\mathscr{L}$	0.11	0.37	0.79	0.88
DC-PMI$_1$	0.08	0.41	0.75	0.88
Instructions				
Short	0.13	0.30	0.65	0.88
Long	0.13	0.31	0.64	0.88
# of In-Context Examples (Ascending Similarity), $k_r = 0$				
1	0.13	0.26	0.62	0.88
2	0.14	0.31	0.65	0.88
3	0.13	0.31	0.66	0.88
4	0.13	0.33	0.67	0.88
5	0.13	0.34	0.67	0.88
# of In-Context Examples (Descending Similarity), $k_r = 0$				
1	0.13	0.27	0.62	0.88
2	0.14	0.31	0.66	0.88
3	0.13	0.31	0.66	0.88
4	0.13	0.33	0.66	0.88
5	0.13	0.34	0.66	0.88
# of In-Context Examples (Ascending Similarity), $k_r = 1$				
1	0.10	0.03	0.38	0.88
2	0.12	0.26	0.63	0.88
3	0.13	0.29	0.64	0.88
4	0.14	0.31	0.65	0.88
5	0.14	0.32	0.66	0.88
# of In-Context Examples (Ascending Similarity), $k_r = 2$				
2	0.10	0.05	0.41	0.88
3	0.12	0.28	0.56	0.88
4	0.12	0.39	0.77	0.88
5	0.13	0.30	0.65	0.88
Model Size				
M	0.13	0.42	0.67	0.88
L	0.15	0.47	0.70	0.88
XL	0.17	0.49	0.70	0.88
Prompt Sharing				
✓	0.14	0.33	0.07	0.88
	0.14	0.31	0.65	0.88
Ensemble Size				
1	0.14	0.31	0.65	0.88
3	0.11	0.23	0.63	0.88
5	0.12	0.21	0.63	0.88
7	0.12	0.19	0.63	0.88
9	0.12	0.18	0.63	0.88

Table D.4 All the experimental results reported for SGD in tabular form. The results are reported for the "**unseen**" all category. The terminology used and the discussion of the results is presented in Chapter 4.