A Study on Prompt-based Few-Shot Learning Methods for Belief State Tracking in Task-oriented Dialog Systems

Debjoy Saha , Bishal Santra , Pawan Goyal
Indian Institute of Technology, Kharagpur, India
sahadebjoy10@iitkgp.ac.in, bsantraigi@gmail.com, pawang@cse.iitkgp.ac.in

Abstract

We tackle the Dialogue Belief State Tracking *(DST)* problem of task-oriented conversational systems. Recent approaches to this problem leveraging Transformer-based models have yielded great results. However, training these models is expensive, both in terms of computational resources and time. Additionally, collecting high quality annotated dialogue datasets remains a challenge for researchers because of the extensive annotation requirement. Driven by the recent success of pre-trained language models and prompt-based learning, we explore prompt-based few-shot learning for Dialogue Belief State Tracking. We formulate the DST problem as a 2-stage prompt-based language modelling task and train language models for both tasks and present a comprehensive empirical analysis on their separate and joint performance. We demonstrate the potential of prompt-based methods in few-shot learning for DST and provides directions for future improvement.

1 Introduction

Dialogue Belief State Tracking is a central problem for task-based conversational systems. The Belief State maintains a distribution of states across different dialogue turns summarising the conversation state at any point by extracting the intent from the user and system inputs. The belief state is used by the system to take appropriate actions at each turn until the conversation is concluded and the user goal is achieved.

Recent State-of-the-art models that tackle the Belief State Tracking problem are generally based on large language models (Hosseini-Asl et al., 2020; Heck et al., 2020; Wu et al., 2019). Their training usually involves huge computation and data requirements, one or both of which might be unavailable. The development of models like BERT (Devlin et al., 2018) and GPT-2 (Radford et al., 2019) has also inspired advances in the use of pre-trained language models (*PLMs*) for low-resource few-shot learning for dialog generation (Zhao et al., 2020). The recent paradigm of *prompt-based learning* (Brown et al., 2020) equips PLMs with constructive task-dependent prompts to simplify language generation for downstream tasks. This method has shown great results on few-shot and zero-shot learning tasks such as classification (Gao et al., 2021; Schick and Schütze, 2021; Han et al., 2021) and text generation (Schick and Schütze, 2020; Li and Liang, 2021). Relatively fewer attempts have been made towards few-shot learning for the DST task of dialog systems. Dingliwal et al. (2021) presents a few-shot meta-learning approach to DST. Peng et al. (2020); Madotto et al. (2021) show few-shot DST performance on just single domain subsets of data. Madotto and Liu (2020) shows few-shot training results on the speech *ACT* (Active Intent) identification task of task-oriented datasets. However, none of the papers present baselines on the end-to-end multi-domain, multi-slot belief-state tracking.

In this paper, we make the first step towards evaluating prompt-based few-shot learning for the end-to-end dialogue to belief state prediction task. Specifically, we formulate the DST task as a two stage language generation problem and provide few-shot performance using pretrained language models like GPT-2, BERT and T5. Our analysis questions the viability of tackling DST in a prompt-based few-shot setting.

2 Proposed Method

A prompt-based DST pipeline: We present a top-level overview of the belief-state prediction pipeline in Figure 1. The first part of the pipeline, the **domain prediction system**, predicts a set of domains, and the second part of the pipeline, **slot prediction system**, assigns values to each of the slots belonging to these domains and constructs the entire belief-state.
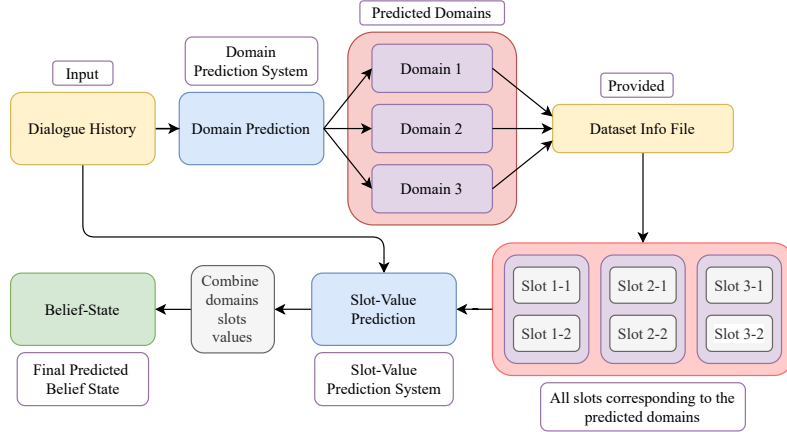
Figure 1: The Dialogue History to Belief State Pipeline

2.1 Domain Prediction

In this part of the pipeline, the task is to predict one or multiple domains from the input text comprising the dialogue history and the prompt. We adopt two different approaches for this.

Using Masked Language Modelling: We extend the dialogue history by the appropriate prompt, containing up to 4 masks, which corresponds to the maximum number of domains that are referred to for any example. For each mask, we select the domain with the highest language modelling score.

Masked Prompt Design: The key challenge in designing the prompt is to have a phrase that can be applied to all the domains normally, thus achieving minimal zero-shot perplexity and maximum few-shot learnability. We adopt the prompt "Excited to see the *[MASK]*." and its multi-mask variants, based on their zero-shot performances, among other similar prompts, like this 2-mask variant "Thank you for the information on the *[MASK]* and *[MASK]*". We also restrict the vocab for domain prediction using MLM to help the model to avoid wrong predictions.

Masked Language Modelling Inference: An important limitation of MLM-based domain prediction is the inability to know the number of domains at run-time, which prevents us from directly using the appropriate prompt for generation. To address this issue, we propose **Weighted Grouped Token Scores (WGS)**. In this method, the model is run four times, once with each prompt to get a set of domain predictions D_k (each containing k prompts) for $k \in [1, 4]$, and the averaged scores are normalised by a weighting factor w_k, to minimise the effects induced by changing the number of masks. The score can be expressed by

the equation: $S_k = \frac{1}{k \times w_k} \times \sum_{d \in D_k} q(d)$, where $q(d)$ is the softmax score for domain $d \in D_k$. The final set of predicted tokens comes out as $\hat{D} = argmax_k(S_k)$. The weights w_k are learnt using a **Genetic Algorithm**[1], over the training set. It was observed that the genetic algorithm assigned almost the same weights to 1, 2, and 3-domain predictions (around 0.35) and a significantly higher weight (0.8) to the 4-domain predictions, indicating that the trained model over-incentivizes 4-domain predictions. However, these weights can be expected to change based on the dataset used.

Using Causal Language Modelling: Using causal language modelling can help remove the issues arising during masked-language model inference, as the generation process is unconstrained and can predict variable number of tokens. We extend the dialogue history by the appropriate prompt, and train the model to predict the appropriate domain string containing all the domains. An issue with making inferences from models trained using datasets whose majority of samples are for 1 or 2 domains was that the output was frequently cut-short to two or fewer domains. To predict longer sequences, we adopt *Unlikelihood Training* for *EOS* (End-of-Sentence)-tokens, similar to Welleck et al. (2019).

Causal Prompt Design: We select a *QA*-style prompt *"What are the mentioned domains?"* as, (a) Prompts similar to the one considered are used in the demo examples provided in Brown et al. (2020) for GPT-2, and (b) *QA*-style prompts for GPT-2 showed better zero-shot performance as

---
[1]Genetic algorithm (Man et al., 1996) is a type of evolutionary computer algorithm that can be used to determine good solutions to optimisation problems.

compared to *continuation*-prompts.

Evaluation metrics: For evaluating domain predictions, we used the metric **Full Accuracy** *(FA)*. For ground-truth set of domains D_{gold} and predicted set of domains D_{pred}, the accuracy is given by the equation: $FA = (D_{pred} == D_{gold})$.

2.2 Slot-Value Predictions

In this sub-problem, we generate values for each slot which is a part of the identified domains. Since these predictions need not necessarily conform to a predetermined template, we just adopt the causal language modelling approach here.

Causal LM Training: This task is formulated similar to the causal LM based domain prediction, the only difference being that we use slot-specific prompts. For example, for *hotel-name*, the prompt can be *"What is the name of the hotel?"*.

Evaluation metrics: In addition to accuracy, we define a flexible accuracy measure which allows small mistakes in the generated outputs, for example, capitalisation or punctuation.

3 Experimental Setup

Dataset: We use the **MultiWOZ-2.2** dataset (Zang et al., 2020), a large-scale, multi-domain dialogue dataset of human-human conversations. Dialogues span over eight domains (restaurant, train, attraction, hotel, taxi, hospital, police, bus) and over 61 domain-slot pairs (*hotel-name*, *hotel-type*, *train-arriveby* and so on). We sample ideally distributed datasets for both the problems.
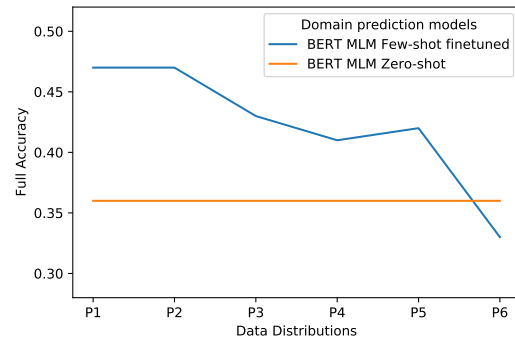
Implementation Details: For masked domain-prediction, we used pretrained masked language model BERT (Devlin et al., 2018). We use GPT-2 (Brown et al., 2020) for CLM-based domain prediction. For the slot-value prediction task, we use GPT-2 (Radford et al., 2019), GPT-neo (Brown et al., 2020) and T5 (Raffel et al., 2019). A detailed description of the hyper-parameters and computational resources are provided in the appendix.
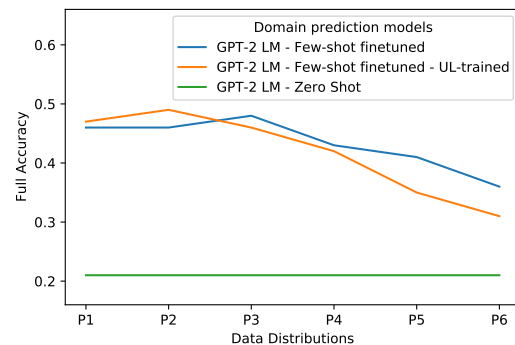
4 Results and Analysis

4.1 Domain Prediction

The **BERT-MLM** model was trained on datasets containing **128** training examples with different data distributions, containing different proportions of 1, 2, 3 and 4-domain data points. We have shown results for various sample distributions in Figure 2a, the proportion of 3 and 4-domain data-points in
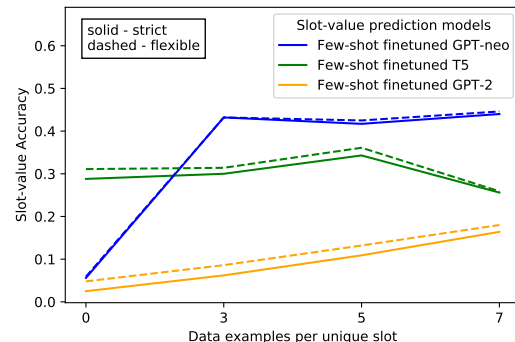
the training data is increased from left to right (P1 to P6).

(a) Domain prediction full accuracy variation with data distribution for BERT-MLM

(b) Domain prediction full accuracy variation with data distribution for GPT2-LM

(c) Slot-value prediction accuracy comparison of GPT-neo, GPT2 and T5 with dataset size

Figure 2: Results from the two stages in our pipeline.

The proportion of test-data having just 1 or 2 ground-truth domains was found to be much higher. Consequently, the full accuracy (Figure 2a) metric shows a gradual decline as the proportion of such data reduces in the training dataset. The overall best performance was obtained as a full accuracy of **0.47** with the data proportions 0.35, 0.35, 0.15 and 0.15 for 1, 2, 3, and 4-domain data respectively.

The **GPT2-LM** model was run on datasets with the same data distributions and dataset size (=128) as BERT-MLM. We present the accuracy metric

full accuracy in Figure 2b for models trained (a) without and (b) with the unlikelihood loss component. The unlikelihood-trained models slightly outperform in a less-domain data setting only to get much worse as the proportion of data with large number of ground-truth domains increases. The reason for this is the over-prediction of domains in the latter case. The best metrics are obtained for the same data distribution as full accuracy of **0.49** for the unlikelihood-trained model.
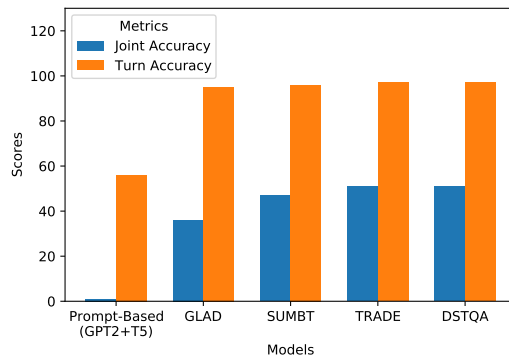
4.2 Slot-value Prediction

The slot-value prediction models were trained on datasets containing a fixed amount of data samples per-slot. The prediction accuracy is displayed in Figure 2c (with the 0-data-per-slot label referring to the zero-shot accuracy). GPT-neo demonstrates the best performance, reaching up to **0.44 accuracy** with 7 data-points per sample (which amounts to a total data-size of 125), followed by T5 with **0.34 accuracy** with 5 data-points per sample. More detailed results are presented in the appendix.
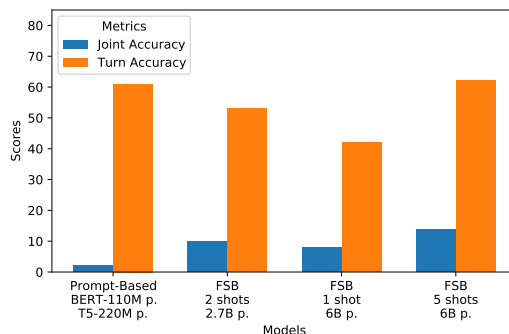
4.3 Full Belief-state Prediction

For getting the final predictions, we use both the best performing domain-prediction models (BERT-MLM and GPT2-LM trained using unlikelihood loss) trained on the 128-sized dataset and the T5-based slot-value prediction model, trained using a dataset containing 5 data points per unique slot giving a total of 80 training samples. T5 is preferred over GPT-neo for slot-prediction, as GPT-neo gives low combined performance due to over-prediction of values (predicting the same value for a set of similar slots) and was observed to generate less slots on average than the T5 model. The final metrics come out as **Joint Accuracy = 1%** and **Slot Accuracy = 56%**

Comparison with few-shot baselines: The only model that presents results on few-shot belief state tracking for the MultiWoz dataset is the Few-Shot Bot (Madotto et al., 2021), which uses language models with knowledge retrievers for performing few-shot inference across multiple tasks. They evaluate their model on a single-domain subset of the MultiWoz dataset. We can see that our prompt-based approach, despite being trained on models having fewer parameters (330M v/s 6B) and on just 128 and 80 data points respectively, gets a lower joint accuracy (2%) but a similar slot accuracy (61%) as the best FSB baselines (Figure 3b).

Comparison with finetuned models: We compare with the end-to-end finetuned state-of-the-art models, GLAD (Zhong et al., 2018), SUMBT (Lee et al., 2019), TRADE (Wu et al., 2019) and DSTQA (Zhou and Small, 2019). From Figure 3a, can get a sense of the large difference between these and this few-shot approach.

(a) End-to-end trained models

(b) Few-Shot Baselines on 1-domain subset

Figure 3: Comparison with existing models

5 Conclusion and Future Directions

This work highlights the difficulties in applying current prompt-based methods for dialog state tracking or, in general, complex multi-step tasks. We find several reasons for this. First is the *propagation of error* through the pipeline. A possible improvement can be to iteratively update the belief-state in each dialogue turn, similar to (Madotto et al., 2021). In this method, provided a proper correction mechanism, errors can be rectified in future dialogue turns. The second reason is the absence of appropriate prompts for domain prediction. This can benefit from the advent of ideas such as soft-prompting (Qin and Eisner, 2021). Lastly, prompt-based models have a dependency on large pretrained models, which can lead to high inference delays. Our experiments show that although prompt-based learning has shown promising per-

formance for few-shot classification problems, its application to more complex tasks is still an open problem and needs further research.

References

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Saket Dingliwal, Bill Gao, Sanchit Agarwal, Chien-Wei Lin, Tagyoung Chung, and Dilek Hakkani-Tur. 2021. Few shot dialogue state tracking using meta-learning. *arXiv preprint arXiv:2101.06779*.

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

Xu Han, Weilin Zhao, Ning Ding, Zhiyuan Liu, and Maosong Sun. 2021. Ptr: Prompt tuning with rules for text classification. *arXiv preprint arXiv:2105.11259*.

Michael Heck, Carel van Niekerk, Nurul Lubis, Christian Geishauser, Hsien-Chin Lin, Marco Moresi, and Milica Gašić. 2020. Trippy: A triple copy strategy for value independent neural dialog state tracking. *arXiv preprint arXiv:2005.02877*.

Ehsan Hosseini-Asl, Bryan McCann, Chien-Sheng Wu, Semih Yavuz, and Richard Socher. 2020. A simple language model for task-oriented dialogue. *arXiv preprint arXiv:2005.00796*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Hwaran Lee, Jinsik Lee, and Tae-Yoon Kim. 2019. SUMBT: Slot-utterance matching for universal and scalable belief tracking. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5478–5483, Florence, Italy. Association for Computational Linguistics.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages

4582–4597, Online. Association for Computational Linguistics.

Andrea Madotto, Zhaojiang Lin, Genta Indra Winata, and Pascale Fung. 2021. Few-shot bot: Prompt-based learning for dialogue systems. *arXiv preprint arXiv:2110.08118*.

Andrea Madotto and Zihan Liu. 2020. Language models as few-shot learner for task-oriented dialogue systems. *ArXiv*, abs/2008.06239.

K.F. Man, K.S. Tang, and S. Kwong. 1996. Genetic algorithms: concepts and applications [in engineering design]. *IEEE Transactions on Industrial Electronics*, 43(5):519–534.

Baolin Peng, Chunyuan Li, Jinchao Li, Shahin Shayandeh, Lars Lidén, and Jianfeng Gao. 2020. Soloist: Few-shot task-oriented dialog with a single pre-trained auto-regressive model. *ArXiv*, abs/2005.05298.

Guanghui Qin and Jason Eisner. 2021. Learning how to ask: Querying lms with mixtures of soft prompts. *arXiv preprint arXiv:2104.06599*.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*.

Timo Schick and Hinrich Schütze. 2020. Few-shot text generation with pattern-exploiting training. *arXiv preprint arXiv:2012.11926*.

Timo Schick and Hinrich Schütze. 2021. Exploiting cloze-questions for few-shot text classification and natural language inference. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 255–269, Online. Association for Computational Linguistics.

Sean Welleck, Ilia Kulikov, Stephen Roller, Emily Dinan, Kyunghyun Cho, and Jason Weston. 2019. Neural text generation with unlikelihood training. *arXiv preprint arXiv:1908.04319*.

Chien-Sheng Wu, Andrea Madotto, Ehsan Hosseini-Asl, Caiming Xiong, Richard Socher, and Pascale Fung. 2019. Transferable multi-domain state generator for task-oriented dialogue systems. *arXiv preprint arXiv:1905.08743*.

Xiaoxue Zang, Abhinav Rastogi, Srinivas Sunkara, Raghav Gupta, Jianguo Zhang, and Jindong Chen. 2020. Multiwoz 2.2: A dialogue dataset with additional annotation corrections and state tracking baselines. *arXiv preprint arXiv:2007.12720*.

Xueliang Zhao, Wei Wu, Chongyang Tao, Can Xu, Dongyan Zhao, and Rui Yan. 2020. Low-resource knowledge-grounded dialogue generation. In *International Conference on Learning Representations*.

Victor Zhong, Caiming Xiong, and Richard Socher. 2018. Global-locally self-attentive dialogue state tracker. *arXiv preprint arXiv:1805.09655*.

Li Zhou and Kevin Small. 2019. Multi-domain dialogue state tracking as dynamic knowledge graph enhanced question answering. *arXiv preprint arXiv:1911.06192*.

A Appendix

A.1 Prompts and formatted texts

Here, we present all the prospective prompts ranked by their zero-shot performance for the domain-prediction task, where DH is the dialogue-history.

- DH Excited to see the $[MASK]$
- DH I am looking forward to see the $[MASK]$.
- DH Are any more details required pertaining to the $[MASK]$
- Can you help me out about a $[MASK]$ DH
- DH Any other questions in regard to the $[MASK]$
- I need some information about a $[MASK]$. DH
- DH So, we talked about the $[MASK]$ right?
- DH Thank you for helping me get all the information regarding the $[MASK]$.
- I need some assistance in regards to finding a $[MASK]$ DH
- DH Thank you for the information on the $[MASK]$.
- DH So we are settled about the $[MASK]$ right?
- I would need a $[MASK]$. DH
- DH I would need a $[MASK]$
- DH Services present are $[MASK]$.

We provide examples of prompt-formatted inputs that are used during inference in Figures 4 and 5. For slot-value prediction, we use the slot-dependant prompts as displayed in 6.

A.2 Detailed results and further analysis

Here, we provide the detailed results from each experiment in tables 2, 3, 4, 5.

Domain prediction with a known number of domains to predict: One additional subject of interest is the full accuracy for BERT-MLM based domain prediction, when the number of domains to predict is known beforehand, thus, the prompt choice is predetermined (Figure 7). These results are significant because of the following reasons -

Figure 4: Variants of the adopted domain-prediction prompt concatenated to the Dialogue History (DH)

Figure 5: Formatted Dialogue History (DH) with QA-prompt and answer for slot-value prediction (Y)

Figure 6: Slot-to-Prompt mapping examples

Figure 7: Domain prediction full accuracy variation with data distribution given the number of domains is known during inference for BERT-MLM

- With a pre known prompt, we get a full accuracy of upto 0.75, about 30% more than the actual full accuracy, which goes on to show

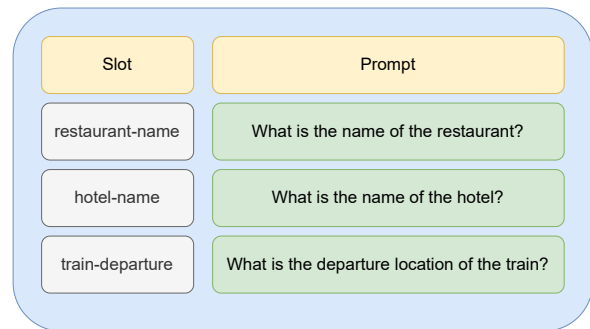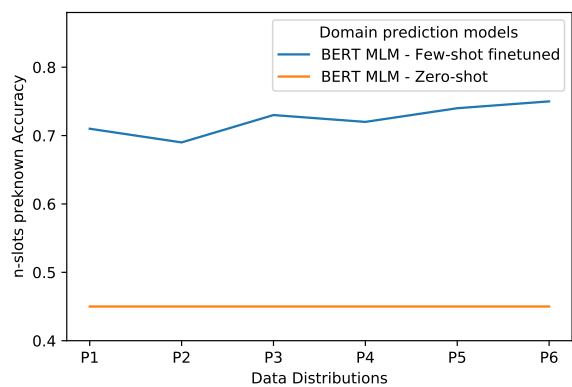| Task	Model	Loss Function	Best Metrics across training runs	
			Full Accuracy	Partial Accuracy
Domain	BERT MLM	NLL Loss	0.47	0.70
Prediction	GPT2 LM	NLL Loss	0.48	0.68
System (DP)	GPT2 LM	Unlikelihood Loss	0.49	0.69
			Accuracy	Flexible Accuracy
Slot-value	GPT2 LM		0.164	0.180
Prediction	T5 LM	NLL Loss	0.343	0.361
System (SvP)	GPT-Neo LM		0.440	0.446
			Turn Accuracy	Joint Accuracy
Combined	GPT2 LM (DP)		**0.56**	**0.01**
System-1	with	N.A.	**(Multi-domain)**	**(Multi-domain)**
(End-to-End)	T5 LM (SvP)		0.58	0.02
			(Single-domain)	(Single-domain)
Combined	BERT MLM (DP)		0.51	0.00
System-2	with	N.A	(Multi-domain)	(Multi-domain)
(End-to-End)	T5 LM (SvP)		**0.61**	**0.02**
			(Single-domain)	**(Single-domain)**

Table 1: Best performing models across all sub-parts of the DST task: For the combined system, we prefer GPT2-LM domain prediction with T5-LM slot-value prediction. We observe that GPT-neo gives low final performance due to over-prediction of values. For the single-domain prediction case, we prefer BERT-MLM domain prediction, as we can constrain the prediction to only one domain using the appropriate prompt.

the potential of MLM-based domain predictions with proper output sampling.

- With an increasing proportion of data having up to 3 or 4 ground truth domains in the training-set, it shows an **increase in full accuracy**, as these data samples provide more data (in terms of masks to predict) per data sample and over-prediction of domains is not an issue given the pre-specified number of masks to include in the prompt.

Comparisons of domain-prediction model performances with dataset size: In this section, we compare the performances for BERT-MLM and GPT-2 LM models for a range of dataset sizes, maintaining constant data distribution. Since there are no existing baselines for the multi-domain prediction task, we design a naïve **Keyword-based Classifier**. Here, we identify a few keywords manually from the dialogues that frequently come in conjunction with each domain. While predicting, we include a domain if any of the corresponding keywords appear in the dialogue history.

According to Figure 8, among the LM-based models, we can see that the GPT-2 accuracy is quite low when trained on smaller datasets, but rapidly increases with the dataset size. This is likely because of the less-constrained nature of generation adopted in GPT-2 LM.

The keyword-based classifier *(Keyword Clf)* obtains a higher accuracy than the LM-based predictors. However, they make some unavoidable mistakes if the selected keywords are absent or are present in a different context. In general, these keyword-based approaches are neither scalable nor

Figure 8: Full accuracy comparison of BERT-MLM and GPT-2 LM for different dataset sizes

very robust, and hence are not preferable.

Slot-value prediction for categorical slots: For slot-value prediction using categorical slots, the possible values that can be predicted are known beforehand (For example, any question involving a day of the week). In that case, it suffices to obtain the predictions by generating just one token and comparing with the first token for each of the possible slots, instead of generating the entire string.

Assuming X to be the dialogue history and $F_S(X)$, the dialogue history concatenated with the prompt corresponding to slot S, which is $p = T(S)$. Also assume $I : V \mapsto v_0$ to be the mapping from value to its first token and $J : v_0 \mapsto V$ as the inverse mapping. Given set of all possible slot-values $Vals$, the probability of the predicted slot to be V is given as follows -

$$P(V|X,S) = \frac{P_M(Y_0 = I(V) \mid F_S(X))}{\sum_{V' \in Vals} P_M(Y_0 = I(V') \mid F_S(X))}$$

Dialogue History

USER: I need a train going to Cambridge that will depart after 10:15 from broxbourne.
SYS: I have train TR5678 that would suit you.
USER: Could you just tell me when that one departs?
SYS: Train TR5678 departs Broxbourne at 10:32 and arrives at Cambridge at 11:32.
USER: Great can you get me a booking for 3 people?

Belief State

train:
"train-arriveby": "11:32"
"train-bookpeople": "3"
"train-departure": "broxbourne"
"train-destination": "cambridge"
"train-leaveat": "10:32"

Relevant Domains

train

Relevant Slots

train-arriveby, train-bookpeople
train-departure, train-destination,
train-leaveat

Predicted Belief State

```
"train": {
    "train-arriveby": "11:32",
    "train-departure": "Broxbourne",
    "train-day": "tuesday",
    "train-bookpeople": "3",
    "train-leaveat": "10:32",
    "train-destination": "Cambridge"
}
```
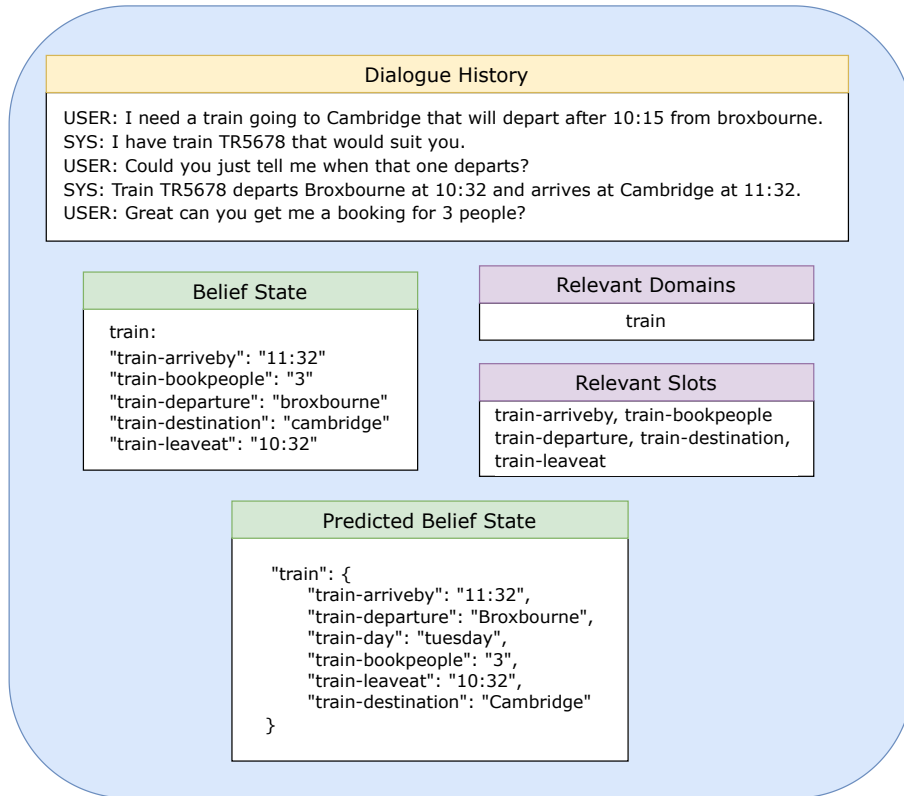
Figure 9: Example of MutiWOZ dialogue with ground-truth and predicted belief-states

Where P_M is the language modelling score for some token. One key thing to note is that, this method only works when all the categorical slots have different first tokens, so the mappings are unique.

Implementation Details: For masked domain-prediction, we used pretrained masked language model **BERT** (Devlin et al., 2018). It was finetuned using Adam optimizer with a learning rate of 1e-7. The maximum number of epochs was capped to 20 and the batch-size was set to 8. Inference predictions were obtained with a single forward pass only. We use **GPT-2** (Brown et al., 2020) for CLM-based domain prediction. GPT-2 was finetuned using Adam(Kingma and Ba, 2014) optimizer with a learning rate of 1e-7. The model was trained for 50 epochs, with a batch-size of 2. For less amount of data and a fixed number of epochs, we found that a batch-size of 2 gave better performance than a batch-size of 4 due to the more frequent weight updates in the former case. During inference, we used beam search decoding with beam size 5.

For the slot-value prediction task, we use **GPT-2** (Radford et al., 2019), **GPT-neo** (Brown et al., 2020) and **T5** (Raffel et al., 2019). T5 was trained using Adam optimizer using a learning rate 1e-5

and the number of epochs was capped to 30. The GPT models were trained using the same hyper-parameters as used previously. During inference, we used beam search decoding with beam size 5. The pretrained model configs are given below -

- **BERT**: bert-base-uncased, 12-layer, 768-hidden, 12-heads, 110M parameters. Trained on lower-cased English text.
- **GPT-2**: gpt2, 12-layer, 768-hidden, 12-heads, 117M parameters. Trained by OpenAI on a very large corpus of English data.
- **T5**: t5-base, 220M parameters with 12-layers, 768-hidden-state, 3072 feed-forward hidden-state, 12-heads, Trained on English text: the Colossal Clean Crawled Corpus (C4).
- **GPT-neo**: gpt-neo-125M, 12-layer, 768-hidden, 12-heads, 125M parameters. Trained on the Pile, a large scale curated dataset created by EleutherAI.

Hardware specification and computational cost: The models were trained on a single Tesla P100 12 GB GPU for about 30 minutes to 3 hours, depending on the model. Full training pipeline would require 4-6 hours.

K-domain data = 128 * pk				BERT MLM		
1	2	3	4	Full	Partial	Pre-known
0	0	0	0	0.36	0.60	0.45
0.4	0.3	0.2	0.1	**0.47**	**0.70**	0.71
0.35	0.35	0.15	0.15	**0.47**	**0.70**	0.69
0.25	0.25	0.25	0.25	0.43	0.68	0.73
0.2	0.2	0.3	0.3	0.41	0.67	0.72
0.15	0.15	0.35	0.35	0.42	0.67	0.74
0.1	0.2	0.3	0.4	0.33	0.63	0.75

Table 2: Full and partial accuracy metrics for BERT-MLM domain prediction

K-domain data = 128 * pk				GPT2 LM		GPT2 LM UL	
1	2	3	4	Full	Partial	Full	Partial
0	0	0	0	0.21	0.28	-	-
0.4	0.3	0.2	0.1	0.46	0.66	0.47	0.67
0.35	0.35	0.15	0.15	0.46	0.66	**0.49**	**0.69**
0.25	0.25	0.25	0.25	**0.48**	**0.68**	0.46	0.67
0.2	0.2	0.3	0.3	0.43	0.64	0.42	0.65
0.15	0.15	0.35	0.35	0.41	0.61	0.35	0.60
0.1	0.2	0.3	0.4	0.36	0.59	0.31	0.57

Table 3: Full and partial accuracy metrics for GPT-2 LM domain prediction (UL = Unlikelihood Training)

Data-Size	GPT2 LM UL		BERT MLM	
	Full Acc	Partial Acc	Full Acc	Partial Acc
16	0.20	0.28	0.33	0.60
32	0.20	0.29	0.30	0.59
64	0.28	0.45	0.37	0.64
128	**0.49**	**0.69**	0.47	0.70
256	0.44	0.65	**0.47**	**0.73**

Table 4: Variation of accuracy metrics for GPT-2 LM and BERT MLM for domain prediction with dataset size

Model	Trained	Data-per-slot (Total)	Strict Acc	Flexible Acc
GPT2	Zero-shot	0	0.025	0.048
	Few-shot	3 (54)	0.062	0.086
	Few-shot	5 (80)	0.109	0.132
	Few-shot	7 (125)	**0.164**	**0.180**
T5	Zero-shot	0	0.288	0.311
	Few-shot	3 (54)	0.300	0.314
	Few-shot	5 (80)	**0.343**	**0.361**
	Few-shot	7 (125)	0.256	0.259
GPT-neo	Zero-shot	0	0.056	0.059
	Few-shot	3 (54)	0.432	0.432
	Few-shot	5 (80)	0.417	0.425
	Few-shot	7 (125)	**0.440**	**0.446**

Table 5: Accuracy metrics for different training configurations of GPT2, GPT-neo and T5 for slot-value prediction.