Out-of-Task Training for Dialog State Tracking Models

Michael Heck, Christian Geishauser, Hsien-Chin Lin, Nurul Lubis, Marco Moresi, Carel van Niekerk, Milica Gašić
Heinrich Heine University Düsseldorf, Germany
{heckmi,geishaus,linh,lubis,moresi,niekerk,gasic}@hhu.de

Abstract

Dialog state tracking (DST) suffers from severe data sparsity. While many natural language processing (NLP) tasks benefit from transfer learning and multi-task learning, in dialog these methods are limited by the amount of available data and by the specificity of dialog applications. In this work, we successfully utilize non-dialog data from unrelated NLP tasks to train dialog state trackers. This opens the door to the abundance of unrelated NLP corpora to mitigate the data sparsity issue inherent to DST.

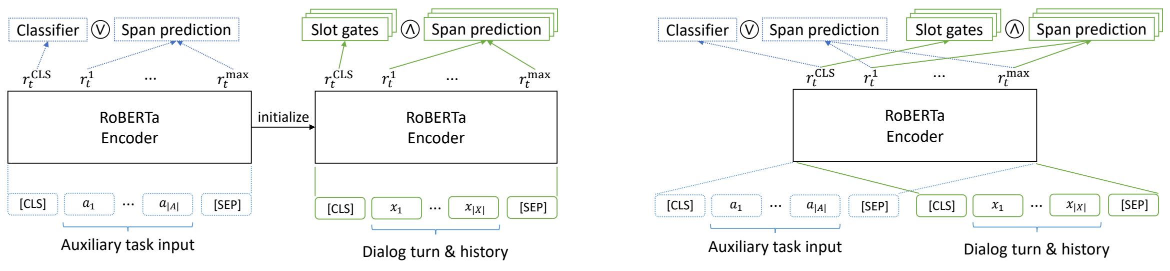
1 Introduction

The role of the dialog state tracker in a task-oriented dialog system is to summarise the history of the conversation so far and extract the user goal. Dialog state tracking (DST) suffers extraordinarily from data sparsity. Collecting data for DST is expensive and time consuming. Typically, conversations are either staged or collected in a Wizard-of-Oz style setup and annotated by hand, severely inhibiting data collection. The enormous number of possible *dialog states* further exacerbates this. Even if we combined all data that commercial assistants generate, there will still be realistic but unobserved dialog states.

Unsupervised learning, transfer learning and multi-task learning (MTL) (Caruana, 1997) in general help mitigate data sparsity. Unsupervised learning relies on predicting inherent characteristics of the data. Transfer learning exploits knowledge learned on related problems to generalize to new tasks. MTL optimizes towards solving multiple tasks at once for synergy effects. Typically, utilized datasets have related domains and tasks share objectives. In other words, these strategies are typically used to address the problem of adaptation.

Recent approaches to adaptation in NLP tasks rely on contextual models. The methods above have been applied to improve generalization across *related* tasks and datasets. For instance, Phang et al. (2018), Wang et al. (2019) and Pruksachatkun et al. (2020) facilitate transfer learning by intermediate task fine-tuning (ITFT) on tasks that are related to the target task. Peng et al. (2020) and Liu et al. (2019a) jointly optimize transformer based models (Vaswani et al., 2017) towards multiple related tasks and/or domains. The latter apply MTL to pre-training, rather than fine-tuning. Gururangan et al. (2020) report improvements by continuing unsupervised pre-training for domain/task adaptation. Raffel et al. (2019) and Keskar et al. (2019) propose model architectures that handle diverse tasks with a unified mechanism.

Natural language understanding (NLU) and dialog state tracking (DST) benefit from joint modeling via multitask learning, as this utilizes dialog data more efficiently (Rastogi et al., 2018). Recent approaches view DST as a generative problem (Wu et al., 2019; Ren et al., 2019) or as a reading comprehension problem (Gao et al., 2019; Chao and Lane, 2019) utilizing contextual models. In the latter, span prediction or sequence tagging extract relevant information from the input directly. These mechanisms utilize training data more efficiently than early approaches that relied on exhaustive classification given a list of known concepts (Mrkšić et al., 2016; Liu and Lane, 2017; Zhong et al., 2018). Pre-training on multiple dialog datasets has been proposed to support subsequent fine-tuning towards specific dialog modeling tasks such as DST (Wu et al., 2020). Synergies between DST subtasks can be exploited via



(a) Intermediate task fine-tuning (ITFT)

(b) Multi-task learning (MTL)

Figure 1: Schematics of our proposed out-of-task training schemes. Blue dotted components are used for training on the auxiliary tasks only. Green solid components are used for DST training only.

MTL (Rastogi et al., 2019). Better generalization across slots is attempted via knowledge transfer and zero-shot learning (Rastogi et al., 2020).

All of the above approaches are suitable to better utilize available dialog data, but the general issue of data sparsity persists. There likely will never be enough task-specific data to train dialog models to their full potential. Instead of resorting to the limited quantities of such data, we propose to utilize *non-dialog data* from *unrelated tasks* for the training of DST models. For this we explore two strategies: (1) in a sequential transfer learning approach, we first train a model to solve an unrelated task, followed by training towards solving DST; (2) we use MTL to jointly optimize towards DST and an unrelated task. We call our overall approach *out-of-task training* for DST.

With our methods, we achieve new state-of-the-art performance on all four target datasets. We show that even small amounts of auxiliary task data are beneficial to support model training, especially with MTL, which particularly improves performance on difficult tasks. Our positive experimental results open the door to the abundance of unrelated NLP corpora defined over a wide range of non-dialog tasks to mitigate the issue of data sparsity in DST.

2 Out-of-Task Training for DST

2.1 Dialog State Tracking

The task of DST is to extract meaning and intent from the user input, and to keep and update this information over the continuation of a dialog (Young et al., 2010). A restaurant recommender needs to know user preferences such as price, location, etc. These concepts are defined by an ontology in terms of domains (e.g., restaurant), slots (e.g., price range), and values (e.g. expensive). We utilize TripPy, our publicly available DST model with state-of-the-art performance on a range of datasets.[1] The details of this model are described in Heck et al. (2020). We briefly describe the aspects relevant to this work.

TripPy encodes the current dialog context using a transformer model. First, the model determines at each turn whether any of the known domain-slot pairs is present. This is done via slot gates, which either predict that a slot can be filled via a copy mechanism or that it takes a special value (none, dontcare, or true/false). There are three copy mechanisms in TripPy; span prediction and two types of memory lookup. Slot gates and span prediction are realized as classification heads on top of the contextual encoder. The model we use in this work is a modification of the original, as we use RoBERTa (Liu et al., 2019b) as encoder instead of BERT (Devlin et al., 2018). We motivate this by the fact that BERT's distinction of segments has little applicability in dialog. When approaching DST as a reading comprehension task, system and user utterance may take on both the roles of query and response. The overall performance of this DST model depends on the individual performance of contextual encoder, slot gates and span prediction, i.e., any of these parts could potentially benefit from out-of-task training.

2.2 Auxiliary Tasks

We consider two types of auxiliary tasks unrelated to DST. The first category encompasses sentence and sentence-pair level classification tasks that aim at discovering linguistic phenomena. We resort to the

[1]Our code is available at `https://gitlab.cs.uni-duesseldorf.de/general/dsml/trippy-public`.

datasets used by the GLUE benchmark (Wang et al., 2018), which cover various NLP problems; (1) MNLI, QNLI, RTE and WNLI for natural language inference (e.g., entailment detection); (2) CoLA for linguistic acceptability classification; (3) SST-2 for sentiment classification; (4) MRPC and QQP for paraphrase detection[2]. We use SQuAD2.0 (Rajpurkar et al., 2018), a question-answering dataset, as representative of token-level classification tasks such as span prediction. SQuaD consists of questions, where the answer to every question is an extractable sequence of text, i.e., a span found in an accompanying paragraph. We refer to the original papers for further details regarding the datasets.

We employ the following training constraints: (1) The auxiliary task can either be a classification problem or a span prediction problem, and (2) only one auxiliary task at a time can be used. The latter allows us to clearly identify the effect of particular auxiliary tasks.

2.3 Intermediate Task Fine-tuning (ITFT)

Our ITFT scheme trains the same model successively on two unrelated tasks, i.e,. the auxiliary task, followed by the DST task. Figure 1a is a depiction of the model architecture. The encoder is either followed by task specific classification heads for DST or a task specific classification head for the auxiliary task depending on the training phase. Both phases of fine-tuning follow the procedure as described by Devlin et al. (2018). The intention of ITFT is to steer the encoder's parameters into a favorable direction so that subsequent fine-tuning finds a better local optimum.

2.4 Multi-task Learning (MTL)

With MTL, we train the same model simultaneously on two unrelated tasks. Figure 1b is a depiction of the model architecture. The strategy is formally outlined in Algorithm 1. For each step s that is to be trained on DST, we also train one additional step on the auxiliary task. In other words, the training alternates between auxiliary task and target task on the level of steps. We share one optimizer for both tasks and perform two successive updates (lines 9 and 12 in Algorithm 1), one for each batch b. The number of these double steps is determined by $s_{\mathtt{max}}$, the maximum number of steps for the target task, so as to not overpower the main task. A hyperparameter $e_{\mathtt{MTL}}$ determines the last epoch for which MTL is applied. If $e_{\mathtt{MTL}} < e_{\mathtt{max}}$ (maximum number of epochs), then we fine-tune only on DST for the remainder.

Algorithm 1: MTL

```
1  $m \leftarrow$ pre-trained RoBERTa
2  $B_{\mathtt{aux}} \leftarrow$ auxiliary task batches
3  $B_{\mathtt{DST}} \leftarrow$ target task batches
4  $e_{\mathtt{MTL}} \leftarrow$ last epoch $e$ to do MTL
5  for $e \leftarrow 1$ to $e_{max}$ do
6      for $s \leftarrow 1$ to $s_{max}$ do
7          if $e \leq e_{MTL}$ then
8              $b^s_{\mathtt{aux}} \leftarrow$ next($B_{\mathtt{aux}}$)
9              $m$.update($b^s_{\mathtt{aux}}$)
10             if $b^s_{aux}$ is last then
11                 reset($B_{\mathtt{aux}}$)
12         $m$.update(next($B_{\mathtt{DST}}$))
13     reset($B_{\mathtt{DST}}$)
```

3 Experiments

Datasets We conduct our evaluation on four dialog datasets. MultiWOZ 2.1 (Eric et al., 2019) is the most challenging, containing over 10k dialogs defined over 5 domains with 30 domain-slot pairs. WOZ 2.0 (Wen et al., 2016) is a single-domain benchmark. sim-M and sim-R (Shah et al., 2018) are single-domain datasets that are challenging due to their high out-of-vocabulary (OOV) rate in some slots.

Scoring We use joint goal accuracy (JGA) on the evaluation sets as the primary measure to compare individual models. JGA is the ratio of dialog turns for which all slots have been filled with the correct value according to the ground truth. We report average JGA over 5 tests each with different seeds.

Training Details As encoder we use *RoBERTa-base*. Training for the first phase of ITFT follows Devlin et al. (2018).[3] For target task training and MTL, the maximum input sequence length is 180 tokens after Byte-pair encoding (Sennrich et al., 2015). We use Adam optimizer (Kingma and Ba, 2014) with joint cross entropy and back-propagate through the entire network. The initial learning rate (LR) is $1e^{-4}$. We conduct training with a warm-up proportion of 10% and linear LR decay. Weight decay is set to 0.01.

[2]We do not consider STS-B, the semantic textual similarity benchmark due to it being defined as regression problem.

[3]For the first phase of SQuaD2.0 ITFT, the maximum input sequence length is 384, the initial LR is $5e^{-5}$, and training is for two epochs. For the first phase of ITFT on GLUE tasks, the initial LR is $2e^{-5}$ and training is for three epochs.

Auxiliary	Available	sim-M		sim-R		WOZ 2.0		MultiWOZ 2.1		average diff.	
Task	Samples[1]	ITFT	MTL	ITFT	MTL	ITFT	MTL	ITFT	MTL	ITFT	MTL
- (Baseline)	2/6/3/57k	88.8	88.8	89.1	89.1	92.1	92.1	56.2	56.2	-	-
SQuaD2.0	130k	91.0*	**92.1****	89.4	90.2**	**92.9***	92.4	56.2	56.9**	0.8	1.4
MRPC	3.7k	89.7	90.7*	89.8*	90.1*	92.7*	**93.1****	55.5	57.1**	0.4	1.2
QNLI	108k	89.6	90.9*	89.9**	89.7*	92.2	92.8*	**56.3**	57.1**	0.5	1.1
SST-2	67k	**91.2****	90.2	**90.4****	89.7	92.2	93.0*	56.0	**57.2****	0.9	1.0
QQP	364k	88.5	90.5*	88.8	89.8*	91.6	93.0*	56.0	57.0**	-0.3	1.0
CoLA	8.5k	90.9**	90.1*	89.7	89.8*	92.6	92.1	56.2	57.1**	0.8	0.7
RTE	2.5k	89.2	89.7	89.5	89.9*	92.3	92.4	55.7	57.1**	0.1	0.7
WNLI	634	89.5	89.4	89.9**	89.4	92.4	92.3	56.2	**57.2****	0.5	0.5
MNLI	393k	89.2	88.3	89.2	**90.3****	92.0	92.1	**56.3**	**57.2****	0.1	0.4
average	-	89.9	90.2	89.6	89.9	92.3	92.6	56.1	57.1	0.4	0.9

Table 1: Performance comparison of out-of-task training methods and utilized tasks. Bold indicates best performance per dataset and training method. ** and * indicate statistically significant improvements over the baseline with $p < 0.05$ and $p < 0.1$, respectively. [1]the maximum use of auxiliary task samples for MTL is $e_{\texttt{max}}$ times the number of samples for the target task (see Section 2.4).

During training we use a dropout (Srivastava et al., 2014) rate of 30% on the RoBERTa output, and 10% on the classification heads. We use early stopping based on the JGA of the development set. $e_{\texttt{max}} = 10$, and $e_{\texttt{MTL}} = 7$. We do not use slot value dropout (Xu and Sarikaya, 2014) except for sim-M.

3.1 Results

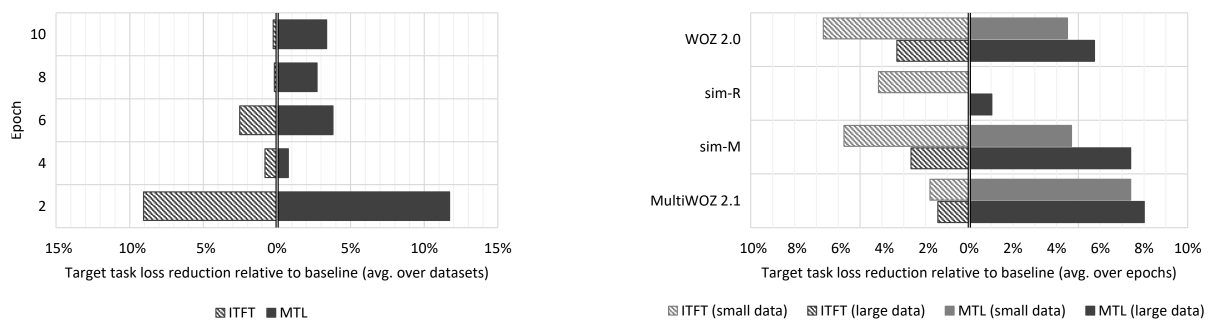
Table 1 lists our out-of-task training results on all four DST datasets, compared to a baseline that does not use any auxiliary task. It can be seen that additional training on an unrelated task produces considerably better models in almost every tested combination. With one exception, the average JGA of all models trained with IFTF or MTL is always higher than the respective baseline performance.

ITFT vs. MTL MTL shows a better performance than ITFT 3 out of 4 times. On every single DST dataset, it is preferrable to use multi-task learning rather than sequential fine-tuning. On average, MTL improves the performance of DST by almost 1% absolute across all datasets, while ITFT improves the average performance by 0.4%. In only one case (QQP), ITFT harms performance consistently. In stark contrast, MTL successfully utilizes QQP to consistently improve performance for all DST tasks. Figure 2a shows that both methods benefit early target task training. However, only MTL, which revisits out-of-task data during training, maintains a positive effect throughout all epochs.

Potential impact of target task difficulty It is reasonable to assume that target tasks do not benefit equally from out-of-task training, depending on the baseline model's initial capacities. WOZ 2.0 can be considered to be the easiest task to solve. sim-R and sim-M both feature slots with high OOV rates, and sim-M contains extremely limited amounts of data. MultiWOZ 2.1 is most challenging. Table 1 shows that more difficult tasks tend to benefit more from MTL than easier tasks. This is also true for ITFT except for MultiWOZ.

Potential impact of data amount The amount of target task data and potential improvement via out-of-task training seems uncorrelated. The amount of available and utilized auxiliary task data likewise does not seem to be decisive. Even the smallest of the datasets (WNLI, RTE, MRPC, CoLA), can be utilized successfully to significantly improve DST. However, we did observe a correlation between the auxiliary task data size and the performance of the training methods. Figure 2b shows that MTL tends to benefit from larger out-of-task datasets, while ITFT performs better on small datasets. MTL only sees a subset of all samples of the auxiliary task if the target task is small, yet clearly outperforms ITFT, which always sees all out-of-task training samples and might therefore suffer from adverse effects of over-training on an unrelated task.

Potential impact of auxiliary task type Table 1 is not indicative of trends regarding the usefulness of particular auxiliary task types. We did observe that span prediction (SQuaD), sentiment classification (SST-2) and linguistic acceptability classification (CoLA) led to more consistent improvements than



(a) ITFT vs. MTL

(b) Impact of data amount

Figure 2: (a) Both methods help early training. In contrast to ITFT, MTL maintains a positive effect on target task training throughout. (b) Loss reductions are averaged over small (WNLI, RTE, MRPC, CoLA; 4k samples on avg.) and large (SST-2, QNLI, SQuaD2.0, QQP, MNLI; 212k samples on avg.) out-of-task datasets. MTL benefits from more data, while ITFT performs better with small datasets.

Auxiliary Task	ITFT						MTL					
	All Slots			High OOV Slots			All Slots			High OOV Slots		
	SA	SGA	SPA	SA	SGA	SPA	SA	SGA	SPA	SA	SGA	SPA
- (Baseline)	96.5	97.6	91.5	91.4	95.3	75.7	96.5	97.6	91.5	91.4	95.3	75.7
Avg. All Tasks	96.8	97.6	92.2	92.0	95.5	77.9	96.9	97.6	92.2	92.2	95.4	77.8
Avg. GLUE Tasks	96.8	97.6	92.2	92.0	95.6	77.8	96.8	97.6	92.1	92.0	95.4	77.4
SQuaD	96.9	97.6	92.5	92.5	95.3	78.6	97.3	97.8	93.1	93.3	95.8	80.4

Table 2: Average slot (SA), slot gate (SGA) and span prediction accuracy (SPA) after ITFT or MTL.

NLI-type tasks and paraphrase detection (MRPC, QQP). The latter two types lead to significantly lower improvements with ITFT, while MTL can benefit from all task types comparably well.

DST training effects SQuaD is the only auxiliary task that utilizes the token-level representations of RoBERTa for prediction, while all other tasks (which are GLUE tasks) solely rely on the sequence-level representation. Table 2 shows that fine-tuning on either task category leads to similar DST performance improvements in terms of average slot gate accuracy. While slot gates expect sequence representations as input, span prediction relies on token representations. As can be seen, out-of-task training with SQuaD leads to larger improvements on span prediction than GLUE tasks. The "movie" and "restaurant" slots in sim-M and sim-R show very high OOV rates (100% and 40%). SQuaD proved most helpful to improve accuracies of these particularly difficult slots. Overall, both task categories proved beneficial for improving DST performance.

4 Conclusion

We investigated auxiliary out-of-task training for DST and found that model training benefits most from joint optimization, compared to sequential training. Even though auxiliary tasks and target task are domain and task mismatched, our training schemes consistently improve target task performance, regardless of task types or data amounts. We reach state-of-the-art results with considerable improvements on all target datasets. We showed that out-of-task training is suitable to overcome data sparsity issues. In future work we pursue the direction of scaling up to do joint out-of-task training on multiple unrelated auxiliary tasks. We would also like to investigate iterative approaches to out-of-task training, where new data is added to the training during the course of a model's lifetime, rather than training from scratch.

Acknowledgements

M. Heck, N. Lubis and C. van Niekerk are supported by funding provided by the Alexander von Humboldt Foundation in the framework of the Sofja Kovalevskaja Award endowed by the Federal Ministry of Education and Research, while C. Geishauser, H-C. Lin and M. Moresi are supported by funds from the European Research Council (ERC) provided under the Horizon 2020 research and innovation programme (Grant agreement No. STG2018_804636). Computing resources were provided by Google Cloud.

References

Rich Caruana. 1997. Multitask learning. *Machine learning*, 28(1):41–75.

Guan-Lin Chao and Ian Lane. 2019. BERT-DST: Scalable end-to-end dialogue state tracking with bidirectional encoder representations from transformer. *arXiv preprint arXiv:1907.03040*.

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*.

Mihail Eric, Rahul Goel, Shachi Paul, Abhishek Sethi, Sanchit Agarwal, Shuyag Gao, and Dilek Hakkani-Tür. 2019. MultiWOZ 2.1: Multi-domain dialogue state corrections and state tracking baselines. *arXiv preprint arXiv:1907.01669*.

Shuyang Gao, Abhishek Sethi, Sanchit Aggarwal, Tagyoung Chung, and Dilek Hakkani-Tür. 2019. Dialog state tracking: A neural reading comprehension approach. *arXiv preprint arXiv:1908.01946*.

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A Smith. 2020. Don't stop pretraining: Adapt language models to domains and tasks. *arXiv preprint arXiv:2004.10964*.

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*.

Nitish Shirish Keskar, Bryan McCann, Caiming Xiong, and Richard Socher. 2019. Unifying question answering, text classification, and regression via span extraction. *arXiv preprint arXiv:1904.09286*.

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

Bing Liu and Ian Lane. 2017. An end-to-end trainable neural network model with belief tracking for task-oriented dialog. *arXiv preprint arXiv:1708.05956*.

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019a. Multi-task deep neural networks for natural language understanding. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4487–4496.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019b. RoBERTa: A robustly optimized BERT pretraining approach. *arXiv preprint arXiv:1907.11692*.

Nikola Mrkšić, Diarmuid O Séaghdha, Tsung-Hsien Wen, Blaise Thomson, and Steve Young. 2016. Neural belief tracker: Data-driven dialogue state tracking. *arXiv preprint arXiv:1606.03777*.

Yifan Peng, Qingyu Chen, and Zhiyong Lu. 2020. An empirical study of multi-task learning on BERT for biomedical text mining. *arXiv preprint arXiv:2005.02799*.

Jason Phang, Thibault Févry, and Samuel R Bowman. 2018. Sentence encoders on STILTs: Supplementary training on intermediate labeled-data tasks. *arXiv preprint arXiv:1811.01088*.

Yada Pruksachatkun, Jason Phang, Haokun Liu, Phu Mon Htut, Xiaoyi Zhang, Richard Yuanzhe Pang, Clara Vania, Katharina Kann, and Samuel R Bowman. 2020. Intermediate-task transfer learning with pretrained models for natural language understanding: When and why does it work? *arXiv preprint arXiv:2005.00628*.

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*.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for SQuAD. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 784–789.

Abhinav Rastogi, Raghav Gupta, and Dilek Hakkani-Tur. 2018. Multi-task learning for joint language understanding and dialogue state tracking. In *Proceedings of the 19th Annual SIGdial Meeting on Discourse and Dialogue*, pages 376–384.

Abhinav Rastogi, Xiaoxue Zang, Srinivas Sunkara, Raghav Gupta, and Pranav Khaitan. 2019. Towards scalable multi-domain conversational agents: The schema-guided dialogue dataset. *arXiv preprint arXiv:1909.05855*.

Abhinav Rastogi, Xiaoxue Zang, Srinivas Sunkara, Raghav Gupta, and Pranav Khaitan. 2020. Schema-guided dialogue state tracking task at DSTC8. *arXiv preprint arXiv:2002.01359*.

Liliang Ren, Jianmo Ni, and Julian McAuley. 2019. Scalable and accurate dialogue state tracking via hierarchical sequence generation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1876–1885.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*.

Pararth Shah, Dilek Hakkani-Tür, Gokhan Tür, Abhinav Rastogi, Ankur Bapna, Neha Nayak, and Larry Heck. 2018. Building a conversational agent overnight with dialogue self-play. *arXiv preprint arXiv:1801.04871*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355.

Alex Wang, Jan Hula, Patrick Xia, Raghavendra Pappagari, R Thomas McCoy, Roma Patel, Najoung Kim, Ian Tenney, Yinghui Huang, Katherin Yu, et al. 2019. Can you tell me how to get past Sesame Street? sentence-level pretraining beyond language modeling. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4465–4476.

Tsung-Hsien Wen, David Vandyke, Nikola Mrkšić, Milica Gašić, Lina M Rojas-Barahona, Pei-Hao Su, Stefan Ultes, and Steve Young. 2016. A network-based end-to-end trainable task-oriented dialogue system. *arXiv preprint arXiv:1604.04562*.

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. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 808–819.

Chien-Sheng Wu, Steven Hoi, Richard Socher, and Caiming Xiong. 2020. ToD-BERT: Pre-trained natural language understanding for task-oriented dialogues. *arXiv preprint arXiv:2004.06871*.

Puyang Xu and Ruhi Sarikaya. 2014. Targeted feature dropout for robust slot filling in natural language understanding. In *Fifteenth Annual Conference of the International Speech Communication Association*.

Steve Young, Milica Gašić, Simon Keizer, François Mairesse, Jost Schatzmann, Blaise Thomson, and Kai Yu. 2010. The hidden information state model: A practical framework for POMDP-based spoken dialogue management. *Computer Speech & Language*, 24(2):150–174.

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

Appendix A. Out-of-Task Training for DST Using BERT

Table 3 summarizes experimental results when using BERT instead of RoBERTa as encoder in TripPy. Even though BERT benefits less from the out-of-task training, the same tendencies as for RoBERTa are observed. One notable difference is the poor performance of SST-2 for BERT. TripPy with BERT uses segment ID 0 for the current user utterance and segment ID 1 for the system utterance plus dialog history, while TripPy for RoBERTa does not distinguish between multiple segments in the input (Devlin et al., 2018; Liu et al., 2019b). Out-of-task training on SST-2 might negatively affect TripPy with BERT, because this data consists of single segments instead of pairs. However, the nature of the task (see Table 4) seems to be relevant, as CoLa - another a single segment classification problem - does not result in such poor performance using BERT. It is noteworthy that the improvements using RoBERTa over BERT for SST-2 are also above average in the official GLUE benchmark leaderboard[4] (while being on average for CoLA), which might indicate a generally higher aptitude of RoBERTa for learning from SST-2.

Auxiliary	Available	sim-M		sim-R		WOZ 2.0		MultiWOZ 2.1		average diff.	
Task	Samples[1]	ITFT	MTL	ITFT	MTL	ITFT	MTL	ITFT	MTL	ITFT	MTL
- (Baseline)	2/6/3/57k	84.4	84.4	88.1	88.1	91.6	91.6	**55.9**	55.9	-	-
SQuaD2.0	130k	84.7	**86.8***	88.2	**88.9***	91.2	**92.5**	55.8	56.2**	0.0	1.1
MRPC	3.7k	85.2*	86.6*	88.5*	88.1	91.3	91.8	55.5	56.4**	0.1	0.7
QNLI	108k	84.8	85.0	88.6	88.3	91.3	92.1	**55.9**	56.6**	0.2	0.5
SST-2	67k	83.9	82.9	88.3	88.7*	91.6	91.3	**55.9**	56.0	-0.1	-0.3
QQP	364k	83.6	83.7	88.8*	88.1	91.2	91.4	**55.9**	56.1	-0.1	-0.1
CoLA	8.5k	85.2*	85.2	**89.0****	88.5	91.3	91.5	55.6	56.4**	0.3	0.4
RTE	2.5k	84.7	85.2*	88.3	88.6	**92.0**	91.6	55.7	**56.7****	0.2	0.5
WNLI	634	84.0	84.3	88.1	88.4	91.3	92.2	55.3	56.3*	-0.3	0.3
MNLI	393k	**86.2****	84.4	88.8*	87.9	91.0	91.4	55.2	56.4**	0.3	0.0
average	-	84.7	84.9	88.5	88.4	91.4	91.8	55.6	56.3	0.1	0.4

Table 3: Performance comparison of out-of-task training methods and utilized tasks when using BERT as encoder. Bold indicates best performance per dataset and training method. ** and * indicate statistically significant improvements over the baseline with $p < 0.05$ and $p < 0.1$, respectively. [1]the maximum use of auxiliary task samples for MTL is $e_{\texttt{max}}$ times the number of samples for the target task (see Section 2.4).

Appendix B. Description of Auxiliary Tasks

Task	Type	Cl.	Input	Task description
CoLA	Classification	2	single	Predicting a sequence's linguistic acceptability for a sequence
SST-2	Classification	2	single	Predicting a sequence's positive or negative sentiment
MRPC	Classification	2	pair	Predicting semantic equivalence of potential paraphrases
QQP	Classification	2	pair	Predicting semantic equivalence of potential paraphrases
MNLI	Classification	3	pair	Predicting if a hypothesis for a premise is neutral, contradiction or entailment
QNLI	Classification	2	pair	Predicting if a text snippet contains the answer given a question
RTE	Classification	2	pair	Predicting if a sentence pair constitutes entailment
WNLI	Classification	2	pair	Predicting if statement 2 is true or false, given statement 1
SQuaD	Span prediction	-	pair	Predicting start and end of answer span given a text and a question

Table 4: Overview of auxiliary tasks that were utilized for out-of-task training for DST. *Cl.* denotes the number of target classes for a task. *Input* is either a single sequence or a sequence pair.

[4]`https://gluebenchmark.com/leaderboard`