# Generative Zero-Shot Prompt Learning for Cross-Domain Slot Filling with Inverse Prompting

**Xuefeng Li**[1]*, **Liwen Wang**[1]*, **Guanting Dong**[1]*,
**Keqing He**[2], **Jinzheng Zhao**[3], **Hao Lei**[1], **Jiachi Liu**[1], **Weiran Xu**[1]
[1]Beijing University of Posts and Telecommunications, Beijing, China
[2]Meituan Group, Beijing, China
[3]School of Computer Science and Electronic Engineering, University of Surrey, UK
{lixuefeng,w_liwen,dongguanting,leihao,ljc1997}@bupt.edu.cn
kqin@bupt.cn, j.zhao@surrey.ac.uk, xuweiran@bupt.edu.cn

## Abstract

Zero-shot cross-domain slot filling aims to transfer knowledge from the labeled source domain to the unlabeled target domain. Existing models either encode slot descriptions and examples or design handcrafted question templates using heuristic rules, suffering from poor generalization capability or robustness. In this paper, we propose a generative zero-shot prompt learning framework for cross-domain slot filling, both improving generalization and robustness than previous work. Besides, we introduce a novel inverse prompting strategy to distinguish different slot types to avoid the multiple prediction problem, and an efficient prompt tuning strategy to boost higher performance by only training fewer prompt parameters. Experiments and analysis demonstrate the effectiveness of our proposed framework, especially huge improvements (+13.44% F1) on the unseen slots.[1]

## 1 Introduction

Slot filling in a task-oriented dialogue system aims to extract task-related information like *hotel_name*, *hotel_address* from user queries, which is widely applied to existing intelligent conversation applications (Tulshan and Dhage, 2019; Zhang et al., 2020). Traditional supervised methods (Zhang and Wang, 2016; Goo et al., 2018; Qin et al., 2019; Wu et al., 2020; He et al., 2020a,b) have shown remarkable performance, but they still rely on large-scale labeled data. Lack of generalization to new domains hinder its further application to practical industrial scenarios.

In this work, we focus on zero-shot cross-domain slot filling which transfers knowledge from the source domain $D_S$ to the target domain $D_T$ without requiring any labeled training data of $D_T$. Conventional approaches (Bapna et al., 2017; Shah et al., 2019; He et al., 2020c; Wang et al., 2021) formulate slot filling as a sequence labeling task and use meta-information such as slot descriptions and slot examples to capture the semantic relationship between slot types and input tokens. However, these models only learn a surface mapping of the slot types between $D_S$ and $D_T$ and get poor performance on unseen slots in the target domain (Wang et al., 2021). Further, (Lee and Jha, 2019; Mehri and Eskenazi, 2021; Du et al., 2021; Yu et al., 2021) propose a machine reading comprehension (MRC) framework for slot filling to enhance the semantic interaction between slot types and slot values. They firstly construct many well-designed question templates based on slot schema or slot examples, then train an MRC model (Rajpurkar et al., 2018a) to predict corresponding slot values for a given slot type question. But they rely on handcrafted question templates using heuristic rules and pre-defined ontologies, which suffers from poor model robustness. Besides, employing additional pre-training on large-scale external MRC datasets is also time-consuming and prohibitively expensive.

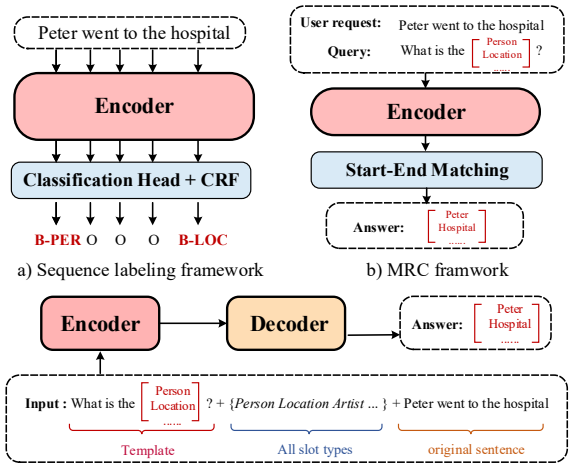


Figure 1: Illustration of different frameworks for zero-shot slot filling.

*The first three authors contribute equally. Weiran Xu is the corresponding author.

[1]Our source code is available at: https://github.com/LiXuefeng2020ai/GZPL



*Findings of the Association for Computational Linguistics: ACL 2023*, pages 825–834
July 9-14, 2023 

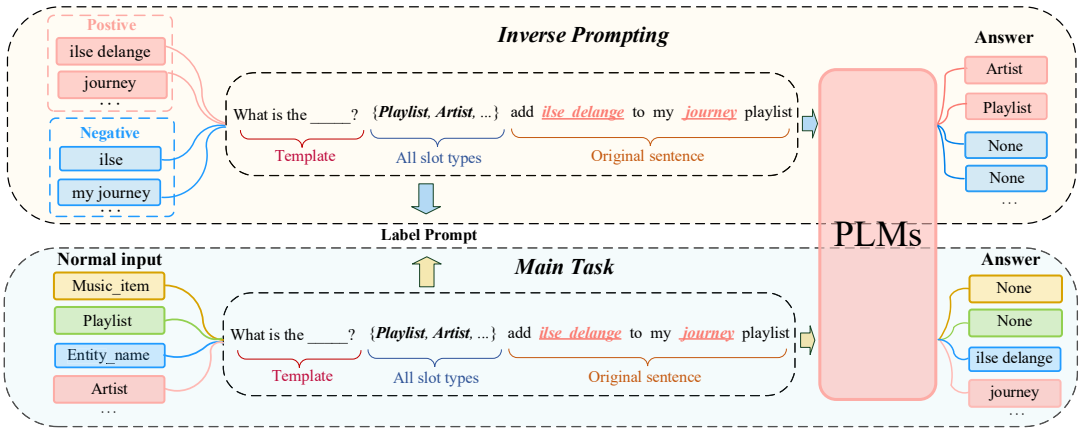


Figure 2: The overall architecture of our proposed GZPL framework with inverse prompting.

To solve the above issues, in this paper, we propose a **G**enerative **Z**ero-shot **P**rompt **L**earning (**GZPL**) framework for cross-domain slot filling. Instead of transforming the slot filling task into sequence labeling or MRC, we formulate it as a language generation task (see Fig 1). Specifically, we concat the question of each slot type, names of all slot types, the input query together to construct the input sequence and take the related slot values as output sequence. The converted text-to-text format has two benefits for zero-shot slot filling: (1) Compared to sequence labeling, our formulation enriches deep semantic interaction between slot types and slot values via pre-trained language models (Raffel et al., 2020), which helps recognize unseen slots only existing in the target domain. We find it significantly improves unseen slot F1 by 13.44% compared to the previous state-of-the-art (SOTA) model (see Section 4.2). The result proves the strong generalization capability to new domains of our proposed framework. (2) Compared to MRC, our framework reduces the complexity of creating well-designed question templates and is more robust to different templates (see Section 4.2). Besides, we concat the names of all slot types into the input sequence to construct direct connections between different slot types, while MRC makes independent predictions for each slot type. Along with our proposed framework, we present an inverse prompting strategy to distinguish different slot types for a given entity to avoid the multiple prediction problem (He et al., 2020d) where the model possibly predicts multiple slot types for one entity span. Different from the above formulation, we take each slot value as input and corresponding slot type as output to build a mapping from entity tokens to entity types. In this way, we force the model to learn explicit distinctions of different types. Inspired by recent parameter-efficient tuning work (Li and Liang, 2021a; Lester et al., 2021), we also introduce an efficient prompt tuning strategy to boost higher performance by training fewer prompt parameters instead of the whole PLM.

Our contributions are three-fold: (1) We propose a simple but strong generative zero-shot prompt learning framework for cross-domain slot filling, which has better generalization capability and robustness than previous work. (2) We present a novel inverse prompting strategy to distinguish different slot types to avoid the multiple prediction problem. Besides, we introduce an efficient prompt tuning strategy to boost higher performance only training fewer prompt parameters. (3) Experiments and analysis demonstrate the effectiveness of our proposed framework, especially for good generalization to unseen slots (F1 $+13.44\% \uparrow$), strong robustness to different templates ($\Delta$ F1 $+10.23\% \uparrow$), parameter efficiency (10x fewer parameters).

## 2 Methodology

Our model is shown in Fig 2. In our framework, we first construct several simple template sentences for the model input, where each sentence includes a slot type question, all slot types and the original query. Then we use a PLM to generate the corresponding slot values. Along with the main task formulation, we perform an inverse-prompting task to warm up the parameters to strengthen the relationship between entities and slot types.

### 2.1 Problem Definition

Given a user input sentence containing $n$ words $\mathbf{X}_{input} = \{x_1, x_2, ..., x_n\}$ and slot type sets $\mathbf{S} = \{s_1, s_2, ..., s_m\}$, the slot filling task aims to find all the entities in $\mathbf{X}_{input}$. For zero-shot setting in our paper, we train models using labeled data from the

source domain and make predictions in the target domain.

### 2.2 Generative Zero-shot Prompt Learning Framework

We customize the entire task using a generative zero-shot prompt learning framework. Specifically, we concat the question of each slot type, names of all slot types, the input query together to construct the input sequence and take the related slot values as output sequence. We formulate it as follows:

$what\ is\ the$ **slot_type** ? **{all slot types}** $x_1\ x_2\ ...\ x_n$

where **slot_type** represents the queried slot type, **{all slot types}** represents all slot types across all domains. For slot types that do not exist in the input, we set the answer to special token "none". For each original input query, we construct QA pairs as the same number of slot types[2].

**Label Prompt Construction** We do not focus on the question template construction as the previous works Du et al. (2021); Yu et al. (2021). Instead, we simply set up the simplest question form of "$what\ is\ the$ ? " to highlight the simplicity and effectiveness of our proposed framework. It is worth noting that we also include slot names from all domains in the prompt. The main purpose of this setting is to enhance the interaction between different slot types, so that the model can find the best answer from the original text.

**Inverse Prompting** Previous MRC works suffer from the multiple prediction problem (He et al., 2020d) where the model possibly predicts multiple slot types for one entity span. To solve such conflict, we design an invert prompting task to warm up the model parameters first. We inverse the original QA pair, that is, set the question to the entities and the answer to the corresponding slot types. This task enables the model to distinguish different slot types for slot entities. In this way, deep semantic relationships between slot types are learned, and the model will learn stronger entity-slot relations. We both train the main task and the inverse task in the same auto-regressive way. Experiments show that first using the inverse task for pre-training then the main task gets the best performance.

In addition, since the result of the main task could be "none", we additionally use a negative sampling strategy here to ensure the consistency of the two tasks. We just randomly sample different spans in sentences, and set the corresponding answers to "none". This strategy can also improve the anti-noise ability of the model and improve the robustness of the framework. In our experiments, we set the ratio of positive and negative samples to 1:1.

**Training and Inference** During training, we try two different training strategies: fine-tuning and prefix-tuning (Li and Liang, 2021b). In the fine-tuning mode, we first use the inverse task to warm up the model parameters, and then perform the main task. All the PLM parameters are finetuned. For prefix-tuning, the parameters of the pre-trained model are fixed during training, and only the parameters of the new added prefix embeddings are trained. Specifically, we add a trainable prefix embedding matrix in each attention layer of the PLM [3]. This method requires 10x fewer trainable parameters and is more parameter-efficient.

During the inference, we only perform the main task. We query for all slot types, and the model directly generates the corresponding slot entities. Compared with the previous method (Yu et al., 2021), our model will not need additional span matching mechanism, so it will be more concise and intuitive. To ensure task consistency with MRC-based models, we add a post-processing step: if multiple slot types predict the same entity span, we choose the answer with the highest generation probability of the first word.

## 3 Settings

### 3.1 Datasets

SNIPS (Coucke et al., 2018) is a public spoken language understanding dataset consisting of crowd-sourced user utterances with 39 slots across 7 domains. It has around 2000 training instances per domain. To simulate the cross-domain scenarios, we follow Liu et al. (2020) to split the dataset, which selects one domain as the target domain and the other six domains as the source domains each time.

### 3.2 Baselines

Sequence Tagging Models: **Concept Tagger (CT)** proposed by (Bapna et al., 2017), which utilizes slot descriptions to boost the performance on detecting unseen slots. **Robust Zero-shot Tagger**

[2]Appendix A shows more details about input and output formats. Appendix B gives the analysis of the inverse-prompting task.

[3]Please see more details in the original prefix-tuning work (Li and Liang, 2021b).

| Training Setting | Sequence tagging-based models | | | | | MRC-based models | | Our models | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Domain ↓ ~ Model → | CT | RZT | Coach | CZSL | PCLC | QASF | RCSF* | GZPL(ft) | GZPL(pt) | GZPL*(ft) | GZPL*(pt) |
| AddToPlaylist | 38.82 | 42.77 | 50.90 | 53.89 | 59.24 | 59.29 | **68.70** | 57.52 | **59.34** | 59.83 | 61.64 |
| BookRestaurant | 27.54 | 30.68 | 34.01 | 34.06 | 41.36 | 43.13 | **63.49** | 57.50 | **63.77** | 61.23 | 62.93 |
| GetWeather | 46.45 | 50.28 | 50.47 | 52.04 | 54.21 | 59.02 | **65.36** | **64.90** | 64.20 | 62.58 | 64.97 |
| PlayMusic | 32.86 | 33.12 | 32.01 | 34.59 | 34.95 | 33.62 | 53.51 | 54.35 | **56.78** | 62.73 | **66.42** |
| RateBook | 14.54 | 16.43 | 22.06 | 31.53 | 29.31 | 33.34 | 36.51 | 31.86 | **38.88** | 45.88 | **47.53** |
| SearchCreativeWork | 39.79 | 44.45 | 46.65 | 50.61 | 53.51 | 59.90 | 69.22 | 66.97 | **71.96** | 71.30 | **72.88** |
| SearchScreeningEvent | 13.83 | 12.25 | 25.63 | 30.05 | 27.17 | 22.83 | 33.54 | 44.80 | **49.83** | 48.26 | **51.42** |
| Average F1 | 30.55 | 32.85 | 37.39 | 40.99 | 42.82 | 44.45 | 55.76 | 53.99 | **57.82** | 58.82 | **61.07** |

Table 1: Slot F1-scores (%) on SNIPS for different target domains under zero-shot settings. ft and pt stands for fine-tuning and prefix-tuning respectively. * indicates the backbone model is a large version of pre-trained model.

**(RZT)** proposed by (Shah et al., 2019), which is based on CT and leverages both slot descriptions and examples to improve the robustness of zero-shot slot filling. **Coarse-to-fine Approach (Coach)** proposed by (Liu et al., 2020), which contains coarse-grained BIO 3-way classification and a fine-grained slot type prediction. In this model, slot descriptions are used in the second stage to help recognize unseen slots, and template regularization is applied to further improve the slot filling performance of similar or the same slot types. **Contrastive Zero-Shot Learning with Adversarial Attack (CZSL-Adv)** proposed by (He et al., 2020c), which is based on Coach and utilizes contrastive learning and adversarial attacks to improve the performance and robustness of the framework. **Prototypical Contrastive Learning and Label Confusion (PCLC)** (Wang et al., 2021), which proposes a method to dynamically refine slot prototypes' representations based on Coach framework and obtains an improved performance.

MRC-based Models: **QA-driven Slot Filling Framework (QASF)**. Contrary to previous methods, Du et al. (2021) introduced MRC-based framework and leveraged the PLMs to solve the problem. **Reading Comprehension for Slot Filling (RCSF)** (Yu et al., 2021), which takes a new perspective on cross-domain slot filling by formulating it as a machine reading comprehension (MRC) problem, which transforms slot names into well-designed queries to improve the detection performance of domain-specific slots.

### 3.3 Implementation Details

We use T5-base[4] as the backbone in our experiments. Model parameters are optimized using the AdamW optimizer (Kingma and Ba, 2014) with a learning rate 5e-05. We set the batch size to 8 and use early stop with a patience 10 to ensure the stability of the model. The prefix length is set to 5 and the dropout rate is set to 0.1. Since RCSF uses the BERT-Large[5] model, we use T5-large[6] model to match the number of parameters of the model used in RCSF. The number of parameters of T5-base[7], T5-large and prefix parameters are 2.2 billion, 7.7 billion, and 20 million, respectively. For all experiments, we train and test our model on 3090 GPU and use f1-score as the evaluation metric. During the training process, we only do prefix-tuning on T5-base, we fix the parameters of T5-base and only fine-tune the parameters of prefix embeddings. We take the average F1 scores of three experiments as our final result.

## 4 Experiments

### 4.1 Main Results

Results show that our proposed framework GZPL significantly outperforms SOTAs. Our base model GZPL(pt) outperforms PCLC by 15.00% and QASF by 13.37% respectively. We don't directly compare our model with RCSF because it uses two unfair settings: using BERT-large as backbone and pre-training it on the QA dataset SQuAD2.0 (Rajpurkar et al., 2018b). Nevertheless, our base model still outperforms RCSF by 2.06%. We adopt another setting to compare with RCSF, that is, change the backbone model to T5-large to ensure that the model size is consistent. We can see GZPL*(pt) with T5-large outperforms RCSF by 6.31%. Besides, we also find using prefix-tuning is better than traditional fine-tuning, which proves prefix-tuning has better knowledge transferability.[8]

### 4.2 Analysis

**Generalization Analysis** Following Wang et al. (2021), if a slot does not exist in the remaining six

[4]T5 is a transformer-based pre-training language model, whose pre-training tasks include text-to-text formulation. We select it as our pre-training model for the consistency between the pre-training tasks and the downstream slot-QA tasks.

[5]https://huggingface.co/deepset/bert-large-uncased-whole-word-masking-squad2

[6]https://huggingface.co/t5-large

[7]https://huggingface.co/t5-base

[8]GZPL without special annotations represent using prefix-tuning unless otherwise noted in the following section.

| | CT | RZT | Coach | PCLC | RCSF | GZPL |
|---|---|---|---|---|---|---|
| seen | 37.23 | 40.99 | 46.22 | 51.69 | **75.96** | 66.49 |
| unseen | 3.38 | 2.19 | 9.31 | 17.38 | 26.21 | **39.65** |

Table 2: Average F1 scores on seen and unseen slots across all target domains.

| $\Delta$ F1 | del "what" | del "what is" | del "what is the" |
|---|---|---|---|
| GZPL | 2.4$\downarrow$ | 3.0$\downarrow$ | 7.2$\downarrow$ |
| RCSF | 12.8$\downarrow$ | 14.1$\downarrow$ | 19.8$\downarrow$ |

Table 3: Average F1 score drop across all domains after the template changes. The smaller number indicates the better effect.

| | 20 (1%) | 50 (2.5%) | 100 (5%) | 200 (10%) | 500 (25%) | 2000 (100%) |
|---|---|---|---|---|---|---|
| RCSF | 0.4 | 0.9 | 2.8 | 9.8 | 17.2 | 55.8 |
| GZPL | 6.2 | 23.7 | 36.5 | 41.5 | 48.2 | 57.8 |

Table 4: Averaged F1-scores (%) over all target domains on SNIPS under the few-shot settings on source domains.

| | GZPL | w/o LP | w/o RP | w/o (LP & RP ) |
|---|---|---|---|---|
| Average F1 | 57.82 | 55.47 | 54.72 | 53.13 |

Table 5: Ablation studies. LP and RP stands for label prompt and inverse prompt, respectively.

source domains, it will be categorized into the "unseen slot" part, otherwise "seen slot". The results are shown in Table 2. We can see that our method outperforms previous methods by a large margin on unseen slots, while performs slightly worse than RCSF on seen slots. Our model focuses more on the generalizable knowledge transfer rather than overfitting on the seen slots in source domains, so it has stronger generalization ability than the previous methods.

**Robustness Analysis** To verify the robustness of our framework, we change the original template "what is the ?" as RCSF. We still use the complete template during training, but delete some tokens of the template during testing, and the results are shown in Table 3. Our model drops slightly by average 4.2% when the template changes, while RCSF drops significantly by 15.6%. This demonstrates that our model is more robust to different input templates.

**Effectiveness Analysis** To further explore the effectiveness of the GZPL under low resource scenarios, we conduct several low-resource settings on source domains, which means only 20, 50, 100, 200 and 500 samples in source domain are used during training stage. As SOTA model (RCSF) does not show results of few-shot experiments, we evaluate RCSF using its open source code. As shown in Table 4, the per formance of our model is much better than that of RCSF under low resource conditions. Besides, with only 100 samples (5%), our model maintains 63.13% performance compared to the results using complete source domain data. While using 500 samples (25%), 82.08% performance can be maintained. This demonstrates our approach is more data-efficient than other slot filling models.

**Ablation Studies** To better prove the effectiveness of the label prompt strategy and the inverse-prompt task, we conduct ablation experiments on these two components. Table 5 illustrates the results of ablation, where "w/o" denotes the model performance without specific module. As we can see, the model will have a slight performance drop (-2.35%) if the slot types in template are removed and the performance of the model will degrade significantly (-3.5%) without the inverse-prompt task. Besides, it is observed that when removing both the label-prompt and inverse-prompt jointly, the performance of the model will drop drastically (-4.69%). This suggests that both of them play an important role in improving the performance.

## 5 Conclusion

In this paper, we introduce a generative prompt learning framework for zero-shot cross-domain slot filling. Based on this, we introduce the label prompt strategy and the inverse prompting to improve the generalization capability and robustness of the framework. Another prefix-tuning mechanism is performed to boost model training efficiency. The exhaustive experimental results show the effectiveness of our methods, and the qualitative analysis inspire new insight into related area. Generally, our framework can be applied to more complex situations, such as nested NER, discontinuous/multiple slots, which we leave to future work. Another interesting direction is to improve the inference efficiency, like concat all the slot questions together and get final results.

## 6 Acknowledgements

This work was partially supported by National Key R&D Program of China No. 2019YFF0303300 and Subject II No. 2019YFF0303302, DOCOMO Beijing Communications Laboratories Co., Ltd, MoE-CMCC "Artifical Intelligence" Project No. MCM20190701.

## References


Ankur Bapna, Gokhan Tur, Dilek Hakkani-Tur, and Larry Heck. 2017. Towards zero-shot frame semantic parsing for domain scaling. *arXiv preprint arXiv:1707.02363*.

Alice Coucke, Alaa Saade, Adrien Ball, Théodore Bluche, Alexandre Caulier, David Leroy, Clément Doumouro, Thibault Gisselbrecht, Francesco Caltagirone, Thibaut Lavril, et al. 2018. Snips voice platform: an embedded spoken language understanding system for private-by-design voice interfaces. *arXiv e-prints*, pages arXiv–1805.

Xinya Du, Luheng He, Qi Li, Dian Yu, Panupong Pasupat, and Yuan Zhang. 2021. QA-driven zero-shot slot filling with weak supervision pretraining. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 654–664, Online. Association for Computational Linguistics.

Chih-Wen Goo, Guang Gao, Yun-Kai Hsu, Chih-Li Huo, Tsung-Chieh Chen, Keng-Wei Hsu, and Yun-Nung Chen. 2018. Slot-gated modeling for joint slot filling and intent prediction. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 753–757, New Orleans, Louisiana. Association for Computational Linguistics.

Keqing He, Shuyu Lei, Yushu Yang, Huixing Jiang, and Zhongyuan Wang. 2020a. Syntactic graph convolutional network for spoken language understanding. In *COLING*.

Keqing He, Yuanmeng Yan, and Weiran Xu. 2020b. Learning to tag OOV tokens by integrating contextual representation and background knowledge. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 619–624, Online. Association for Computational Linguistics.

Keqing He, Jinchao Zhang, Yuanmeng Yan, Weiran Xu, Cheng Niu, and Jie Zhou. 2020c. Contrastive zero-shot learning for cross-domain slot filling with adversarial attack. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 1461–1467, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Keqing He, Jinchao Zhang, Yuanmeng Yan, Weiran Xu, Cheng Niu, and Jie Zhou. 2020d. Contrastive zero-shot learning for cross-domain slot filling with adversarial attack. In *COLING*.

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

Sungjin Lee and Rahul Jha. 2019. Zero-shot adaptive transfer for conversational language understanding. In *AAAI*.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Xiang Lisa Li and Percy Liang. 2021a. 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.

Xiang Lisa Li and Percy Liang. 2021b. 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.

Zihan Liu, Genta Indra Winata, Peng Xu, and Pascale Fung. 2020. Coach: A coarse-to-fine approach for cross-domain slot filling. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 19–25.

Shikib Mehri and Maxine Eskenazi. 2021. Gensf: Simultaneous adaptation of generative pre-trained models and slot filling. In *Proceedings of the 22nd Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 489–498.

Libo Qin, Wanxiang Che, Yangming Li, Haoyang Wen, and Ting Liu. 2019. A stack-propagation framework with token-level intent detection for spoken language understanding. 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 2078–2087, Hong Kong, China. Association for Computational Linguistics.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018a. 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, Melbourne, Australia. Association for Computational Linguistics.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018b. 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.

Darsh Shah, Raghav Gupta, Amir Fayazi, and Dilek Hakkani-Tur. 2019. Robust zero-shot cross-domain slot filling with example values. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5484–5490.

Amrita S. Tulshan and Sudhir N. Dhage. 2019. Survey on virtual assistant: Google assistant, siri, cortana, alexa. *Communications in Computer and Information Science*.

Liwen Wang, Xuefeng Li, Jiachi Liu, Keqing He, Yuanmeng Yan, and Weiran Xu. 2021. Bridge to target domain by prototypical contrastive learning and label confusion: Re-explore zero-shot learning for slot filling. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 9474–9480.

Di Wu, Liang Ding, Fan Lu, and Jian Xie. 2020. Slotrefine: A fast non-autoregressive model for joint intent detection and slot filling. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1932–1937.

Mengshi Yu, Jian Liu, Yufeng Chen, Jinan Xu, and Yujie Zhang. 2021. Cross-domain slot filling as machine reading comprehension. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21, Montreal, QC, Canada*, pages 19–26.

Xiaodong Zhang and Houfeng Wang. 2016. A joint model of intent determination and slot filling for spoken language understanding. In *IJCAI*.

Zheng Zhang, Ryuichi Takanobu, Qi Zhu, MinLie Huang, and XiaoYan Zhu. 2020. Recent advances and challenges in task-oriented dialog systems. *Science China Technological Sciences*, 63(10):2011–2027.

## A Details about the input and output formats

Table 6 shows an example of how to perform slot filling tasks for a user query under our settings. As shown in the table, since we already know the slot type information for the domain the data belongs to, we will customize the unique questions for each slot type according to our template and the model then generate the answers for each question. The answer can be one or more spans in the original sentence, or be the special token "none". It is worth noting that when a slot type corresponds to multiple slot entities, the answer will be separated by commas. However, this situation hardly exists in the Snips dataset, so it is rare to have multiple spans as answers when testing.

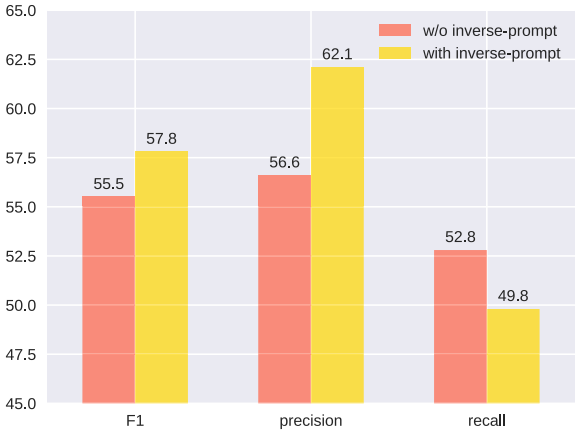


Figure 3: Impact of the proposed inverse-prompt task on F1, precision and recall scores.

## B Analysis of the Inverse-prompting Task

To further explore whether our auxiliary task alleviates the problem of repeated generation, we verify its effect through the following two metrics: precision and recall score. We use these metrics based on our recognition that repeated generation will result in more entities being predicted. On the one hand, this will improve the recall score, and on the other hand, it will hurt the accuracy of the model prediction. The experimental results are shown in Figure 3. As can be seen from the figure, after adding this inverse-prompt task, the recall-score of the model decreased by 3%, while the precision-score increased by 5.5%, which also increased the overall f1-score by 2.4%. We also conducted a case study on the output of the model, and the results are shown in Table 7. After the tasks are added, the repeated generation of the model is significantly reduced. These results above illustrate that the proposed task enables the model to learn deep relationships between slot types, thereby reducing the problem of repeated generation.

## C Limitations and Future Work

The current work does achieve better performance than previous methods, but processing only one slot type at a time also reduces the efficiency of the model. In the future, we will explore how to maximize model efficiency. It would be an interesting challenge to generate answers for all the slots at once without degrading the effect of the model. Also, we will also try to apply our framework to more scenarios, such as NER and other tasks to explore the adaptability of the proposed method.

| Domain | SearchCreativeWork |
|---|---|
| slot types in this domain | object type, object name |
| all_alot_types | artist, playlist....object type, object name.... |
| query | play the game sugarfoot |
| input1 | what is the object type ? artist, playlist....object type, object name.... play the game sugarfoot |
| output1 | game |
| input2 | what is the object name ? artist, playlist....object type, object name.... play the game sugarfoot |
| output2 | sugarfoot |

Table 6: An example showing the details of the input and output formats under our settings.

| Case Study | Data |
|---|---|
| Query | add ilse delange to my journey playlist |
| Answer | music_item→none; playlist_owner→none; entity_name→none; playlist→journey; artist→ilse delange |
| w/o Inverse Prompting | **music_item→ilse delange;** playlist_owner→none; entity_name→none; playlist→journey; artist→ilse delange |
| w Inverse Prompting | **music_item→none;** playlist_owner→none; entity_name→none; playlist→journey; artist→ilse delange |

Table 7: The case study of GZPL w/o Inverse Prompting

## ACL 2023 Responsible NLP Checklist

### A For every submission:

☑ A1. Did you describe the limitations of your work?
*Section 4(line 288 293) and Appendix E.*

☑ A2. Did you discuss any potential risks of your work?
*Section 2.2. The description in Inverse Prompt(line157-160).*

☑ A3. Do the abstract and introduction summarize the paper's main claims?
*Abstract(Line 7 16) Introduction(Line105 119)*

☒ A4. Have you used AI writing assistants when working on this paper?
*Left blank.*

### B ☒ Did you use or create scientific artifacts?

*Left blank.*

☐ B1. Did you cite the creators of artifacts you used?
*No response.*

☐ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
*No response.*

☐ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
*No response.*

☐ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
*No response.*

☐ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
*No response.*

☐ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
*No response.*

### C ☑ Did you run computational experiments?

*Left blank.*

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
*Appendix B(Implementation details)*

*The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.*

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
*Section 3.1 and Appendix B.*

☑ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
*Appendix B(line 472 474)*

☑ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
*Appendix B*

**D ☒ Did you use human annotators (e.g., crowdworkers) or research with human participants?**

*Left blank.*

☐ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
*No response.*

☐ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
*No response.*

☐ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
*No response.*

☐ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
*No response.*

☐ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
*No response.*