Synergistic Augmentation: Enhancing Cross-Domain Zero-Shot Slot Filling with Small Model-Assisted Large Language Models

Weizhen Li[†], **Junbao Huang**[†], **Peijie Huang**[*], **Yuhong Xu, Jiekun Fan**
College of Mathematics and Informatics, South China Agricultural University, China
lwz1311620816@163.com, gumbouh@163.com, pjhuang@scau.edu.cn,
xuyuhong@scau.edu.cn, jiekunfan@163.com

Abstract

In real-world scenarios, cross-domain slot filling in spoken language understanding remains a significant challenge due to data scarcity. Previous works exhibit limited generalization ability in the target domain, demonstrating effective knowledge transfer only on seen slots while performing poorly on unseen slots. Although large language models (LLMs) can alleviate this issue to some extent, they underperform on seen slots compared to small models. To address these challenges, we introduce a novel framework that harnesses the power of a small model to augment the inferential capabilities of LLMs without additional training. Initially, we utilize target domain samples synthesized by LLMs as pre-calculated demonstrations, which are curated and chosen using confidence metrics derived from a small model. We further extract slot predictions from the small model to fully exploit its robust learning of familiar slots. Finally, during the inference process for test inputs, we integrate these demonstrations and slot prediction insights as references to enhance the slot filling performance of LLMs. Experiments on a slot filling dataset and a NER dataset including eight cross-domain settings show our framework achieves the best results. Our codes are publicly available at https://github.com/SIGSDSscau/SLSF.

1 Introduction

Slot filling is a core component in dialogue systems that aids in understanding user needs, directly impacting task completion rates and user experience. The task involves extracting and categorizing slot entities (such as artist or playlist) from user utterances. Previous slot filling studies (Kurata et al., 2016; Wang et al., 2018; Qin et al., 2019) use a single-domain training and evaluation learning paradigm. Thanks to the rapid advancements in deep learning and the abundant labeled training data, the paradigm gains significant momentum. However, there remain issues of time-consuming efforts and difficulties in obtaining labeled data. As a result, researchers start exploring cross-domain slot filling (CDSF) to transfer annotation-rich knowledge of multiple source domains to annotation-scarce target domains, which aims to improve cross-domain performance.

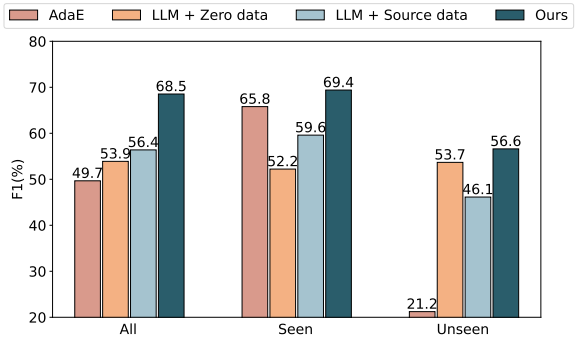


Figure 1: The average performance of different slots across seven different domains on SNIPS using four methods. "All", "Seen", and "Unseen" respectively represent all slots, slots that appear in both the source and target domains, and slots that only appear in the target domain. "Zero data" means there are no examples to select, while "Source data" indicates selecting examples from the source domain.

Traditional methods (Lee and Jha, 2019; Shah et al., 2019; He et al., 2020; Shi et al., 2023; Liu et al., 2022b) involve training a model in the source domain and achieving good performance in the target domain. However, they can only learn features of unseen slots from a limited number of similar seen slots, which leads to low unseen slot performance, as shown by AdaE in Figure 1. With the increasing capabilities of LLMs (Sarkar et al., 2023; Imran et al., 2024; Li et al., 2024), we test their performance on different slots in CDSF under various settings, as shown in Figure 1. Although LLMs perform exceptionally well on unseen slots, their performance on seen slots still falls short compared

[†] Equal contribution.
[*] Corresponding author.

Findings of the Association for Computational Linguistics: ACL 2025, pages 21299–21312

to small models.

In this paper, we aim to explore how small models assist LLMs in integrating their own strengths and enhancing LLMs' overall performance without additional training. Specifically, we do not use source domain samples as demonstrations. As shown in Figure 1, while the presence of slots in these samples similar to the target domain can enhance LLMs' performance, there are some irrelevant noisy slots or text that degrade performance on unseen slots. So we utilize samples generated by LLMs as demonstrations that are similar to the target domain. However, LLMs suffer from hallucination issues, leading to inevitable noisy synthetic samples. Therefore, we proposed a data curation mechanism to effectively alleviate the harmful synthetic sample problem. Our approach is inspired by the principles of learning theory that show the state changes of samples during training, offering crucial insights into the sample's value (Arpit et al., 2017; Arora et al., 2019; Li et al., 2020). A synthetic sample that is dynamically mispredicted or has ambiguous predictions could potentially harm a model. With a curation model trained on the source data, we analyze the learning evolution of synthetic samples and curate these by evaluating their evolution of confidence. Based on this, we select suitable demonstrations from the curated dataset for each test input. To further leverage the robust learning of familiar slot types by the small model, we provide the small model's predictions to LLMs as references. Finally, by combining the selected high-quality synthetic demonstrations, small model's prediction and test input, we input them into LLMs to obtain more accurate final prediction results.

Overall, our main contributions can be summarized as follows: (1) To the best of our knowledge, we are the first to explore completing CDSF using LLMs' inference without training it. (2) We propose a method that leverages LLMs to synthesize samples as demonstrations of In-context learning, enhancing its understanding of slots in the target domain. (3) We propose a novel framework that leverages small models to augment the slot filling capability of LLMs. In our framework, we explore different strategies to curate synthesized samples, select samples with relatively high confidence as demonstrations for LLMs learning and provide slot prediction as reference from small models, leveraging the small model's effective learning of familiar slots. (4) Experiments demonstrate that the performance of our proposed framework improves significantly on different slots and eight cross-domain settings in CDSF compared to baselines.

2 Related Work

Cross Domain Slot Filling. Previous approaches (Bapna et al., 2017; Shah et al., 2019; He et al., 2020; Wang et al., 2021) utilize meta-information such as slot descriptions and slot instances to capture the semantic relationship between slot types and the user query. However, these models only learn surface mappings of slot types between different domains and perform poorly on unseen slots in the target domain. To alleviate this problem, recent approaches (Zhang and Zhang, 2023; Shi et al., 2023) adopt a contrastive learning framework to learn a generalized feature representation. Combining learnable prompts and slot descriptions (Luo and Liu, 2023) and proposing a prompt-based hierarchical pipeline with three innovations (Wei et al., 2024) achieves more accurate cross-domain performance. Nevertheless, the generalization capability of these models is limited due to a lack of learning in the target domain slots.

Aimed at enhancing this ability, some methods (Du et al., 2021; Liu et al., 2022b) treat slot filling tasks as machine reading comprehension tasks to enhance the semantic interaction between slots and user queries, and further pre-train models on large-scale external MRC datasets. Other methods (Yan et al., 2022; Li et al., 2023) frame the task as a language generation task to leverage the rich pre-training knowledge of large-parameter generative models. Both of these methods introduce general knowledge to better answer slot values corresponding to certain slots lacking knowledge. However, their performance on unseen slots still falls short.

Moreover, recent studies (Imran et al., 2024; Sarkar et al., 2023; Li et al., 2024) of LLMs show strong generalization in zero-shot and few-shot scenarios for different tasks. So we explore the performance of LLMs on our task, surpassing previous methods on unseen slots. However, LLMs still fall short in overall performance on CDSF (Wei et al., 2024). Therefore, we leverage small models to supplement the performance deficiencies of LLMs.

Demonstrations in In-context Learning. Some works explore generating various demonstrations in ICL (Lyu et al., 2023; Xie et al., 2024). Xie et al. (2024) use LLMs to make predictions on the unlabeled NER corpus as demonstrations. However, CDSF has a more challenging zero-shot setting,

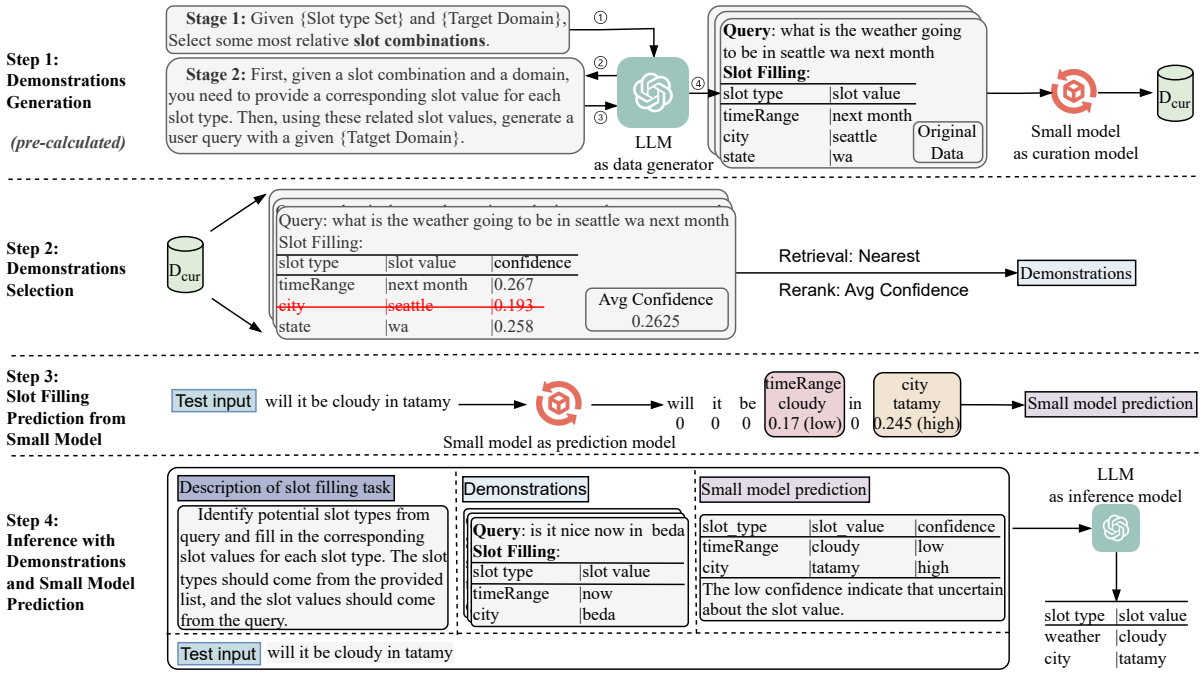


Figure 2: The overview of our proposed framework for cross-domain zero-shot slot filling with LLMs.

where there are no samples from the target domain. Therefore, we first use LLMs to generate samples of target domains as a demonstration corpus.

Previous methods randomly select examples from training data (Brown et al., 2020; Lewkowycz et al., 2022), which may hinder the potential of LLMs. So retrieval-based method (Mishra et al., 2022; Luo et al., 2023; Scarlatos and Lan, 2024) is intensively investigated. Recent works (Wu et al., 2023; Peng et al., 2024; Xie et al., 2024) adopt a select-then-rank framework, ranking the demonstrations selected by the k nearest neighbors (Nearest, (Liu et al., 2022a)) method based on different reordering techniques. Xie et al. (2024) use self-consistency (SC, (Wang et al., 2023)) scores to rerank the selective samples. This approach involves using unlabeled corpora for multiple invocations of LLMs, consuming more resources and it is difficult to apply to our task where lack target domain data. Furthermore, small models can be used to evaluate the quality of sample labeling (Seedat et al., 2024). So we utilize small models to get label confidence of synthesized samples and use it to select suitable samples as demonstrations.

3 Method

As shown in Figure 2, our framework is divided into four steps. **In the first step**, we use LLMs to generate target domain samples and use a small model to curate a high-quality synthetic dataset as a demonstration corpus. The corpus can be pre-calculated and saved for LLMs' inference. **Then**, we retrieve some similar demonstrations using Nearest method for each test input, then select more suitable demonstrations from these based on Average Confidence. **Next step**, we obtain the slot-filling predictions for the test input from the small model to provide reference information for LLMs. **In the final step**, we combine suitable demonstrations, slot-filling predictions of a small model, and test input, and input them into LLMs to infer the final result. The last three steps are iterated to generate slot filling results for each test input.

3.1 Step 1: Demonstrations Generation

3.1.1 Original Data Generation

Directly selecting candidate slots and synthesizing samples in a prompt makes it difficult to control the generation of diverse synthetic data. Therefore, we decompose the task of generating data into two-stage subtasks to alleviate this issue. We first select multiple slot combinations as seeds, and then use them for data synthesis to ensure the balance and diversity of data categories. The complete template is shown as Appendix C.1.

Stage 1: Slot Combinations Selection. During this stage, our aim is to select diverse and relevant slot combinations as seeds for generating samples. Utilizing all slots or random slots from the target domain for data synthesis poses challenges for

LLMs in understanding the relationships among different slots, resulting in the generation of semantically confused, low-quality samples. Therefore, we construct an in-context prompt to select more relevant slot combinations by combining some examples from the source data.

Stage 2: Data Synthesis. For ease of annotation and to generate high-accuracy labels, we create a Chain of Thought (COT) prompt template. We use natural language to describe the task process and provide examples for LLMs to learn the relationships between certain slots and their values. The template involves initially generating values for each slot within the slot combination, which are subsequently used to create synthetic samples. Finally, we annotate these using the IOB2 format, where B- marks the beginning of a slot, I- indicates a continuation of a slot, and O denotes non-slot words. For example, generating the slot value pair "city: big delta" and the sample "weather for big delta", the annotation would be "O O B-city I-city".

3.1.2 Data Curation

The two-stage data generation approach combines examples from the source domain and COT prompts to generate samples, making the syntactic structure, language style, and quality of the synthesized samples closer to the target distribution. However, LLMs exhibit hallucination phenomena, leading to deviation from the real distribution. To improve the relevance of this distribution to the target domain, we establish a data curation mechanism to eliminate potentially mislabeled tokens.

Throughout the training process of the small model, we save some different checkpoints and gain insights into the evolution of token predictions from them. Some tokens are predicted accurately, while others present challenges, potentially due to erroneous labels, resulting in incorrect predictions. Hence, we utilize these checkpoints as the foundation of our curation mechanism.

Learning Evolution. Here we demonstrate how to observe the learning evolution of each token. To ensure higher accuracy of each synthesized sample during the learning state, we choose a small model and train it on the source data as our curation model. The model progresses through different checkpoints, forming a collection: $\mathcal{T} = \{t_1, t_2, \ldots, t_n\}$, where t_n represents the model at the n-th checkpoint. We indicate the predicted logits for slot type y of slot value v in sample x by using the n-th checkpoint reflecting the learning state:

$$L_n(x, v)_y = \frac{1}{S} \sum_{i=1}^{S} [t_n(x, v_i)] y_i, \tag{1}$$

where S denotes the token number of slot value. Our objective is to use the average slot type logits of corresponding slot value within each token across the curation model's n checkpoints to assess the quality of the synthesized samples.

Curation Metrics. To enhance the evaluation of the quality of synthesized samples, inspired by (Kwon et al., 2020; Seedat et al., 2022, 2024), we employ **Confidence** for sample (x, y, v) to serve as data curation metrics. For the set of checkpoints $\mathcal{T}$, the confidence is defined as the following marginal:

$$\bar{P}_{\mathcal{T}}(x, y, v) = \frac{1}{n} \sum_{i=1}^{n} L_i(x, v)_y. \tag{2}$$

Curation Strategies. Based on the confidence, we define a threshold α for filtering and a **Average Confidence**: $Avg(x, y) = \frac{1}{J} \sum_{j=1}^{J} \bar{P}_{\mathcal{T}}(x, y, v^j)$ to evaluate all labels confidence of a test input, which J denotes the slot value number of input x. We investigate the following three strategies for curating the original data. (1) **Value-level filtering**, which drops the predicted slot value v if $\bar{P}_{\mathcal{T}}(x, y, v) < \alpha$. (2) **Value-level replace**, which replace the slot type of v with the type predicted by the small model if $\bar{P}_{\mathcal{T}}(x, y, v) < \alpha$. (3) **Sentence-level filtering**, which drops the synthesized sample x if $Avg(x, y) < \alpha$. After filtering, we obtain the curated data as D_{cur}.

3.2 Step 2: Demonstrations Selection

To dynamically select more relevant and accurate examples, we design a method based on semantics and **Average Confidence** for selecting demonstrations. When a test input arrives, we retrieve K samples from D_{cur} using Nearest method and rerank these by recalculating $Avg(x, y)$ after the filter. Finally, we select the top-M samples using these rerank-scores as the demonstrations $\mathcal{C}_{rank}$:

$$\mathcal{I} = \underset{j \in \{1, \ldots, K\}}{\text{argTopM}} Avg(x_j, y_j), \tag{3}$$

$$\mathcal{C}_{rank} = \{(x_q, y_q) \mid q \in \mathcal{I}\}. \tag{4}$$

3.3 Step 3: Slot Filling Prediction from Small Model

To further utilize small models for robust learning of familiar to the target domain, we input test inputs

into the best checkpoint from the curation model to obtain predicted slot value pairs. Then, we use Equation (2) to compute the confidence scores for each predicted slot value pair. In diverse domains with varying lengths of slot label lists, the standards for high-confidence labels differ. LLMs face difficulties in comprehending the reliability signified by the confidence levels of slot-value pairs. By setting a dynamic confidence threshold β in different target domains, when the confidence is below this threshold, it indicates that the prediction reliability is low. Otherwise, it is more reliable. By converting confidence scores into natural language descriptions and organizing it into a slot value pair table as shown in Figure 2, LLMs use more proficient natural language-based confidence assessments to obtain more accurate slot predictions.

3.4 Step 4: Slot Filling with Demonstration and Small Model Prediction

For each test input, we first select appropriate synthetic samples as demonstrations. Subsequently, the test inputs are inputted into a small model to obtain a reference table containing slot filling predictions and corresponding confidences. Lastly, we combine these with the task description and test input to create a powerful template. We feed it into LLMs to produce a final slot filling prediction:

$$\hat{y} = \text{LLM}(Desc; \mathcal{C}_{rank}; D; x), \tag{5}$$

where $Desc$ denotes a natural language description of the slot filling task and D denotes the table of small model prediction results. The complete prompt template for slot filling is provided in the Appendix C.2.

4 Experiment

4.1 Setup

Datasets and Settings. Following previous works (Shi et al., 2023; Li et al., 2023), we conduct extensive experiments on **SNIPS** (Coucke et al., 2018): It defines 39 different slot types spanning 7 domains with each domain having its slot types and around 2000 annotated samples. For each domain, we select it as the target domain and other domains as the source domain. So we obtain seven cross-domain experimental settings. The source domain samples are used to train a small model. We set aside 500 samples in the target domain for validation to select the best model, which is used to test the remaining samples. We also test the remaining samples with LLMs.

Baselines. We employ two types of baselines to compare with our method:

- Model scale. Small model training with source data (**AdaE** (Shi et al., 2023) and **RCSF** (Liu et al., 2022b)); LLM (**ICL with different data**).
- Retrieval methods. **Random** baseline randomly selects in-context examples for each testing sample. **BM25** (Robertson and Zaragoza, 2009) uses BM25 to calculate the word-overlap similarity between samples and test input, and select the high similarity samples as demonstrations. **Nearest** (Liu et al., 2022a) uses the nearest neighbors of a given test sample as the corresponding in-context examples. **Nearest + SC ranking** (Xie et al., 2024) first retrieve K nearest neighbors and select samples with the top-k sample-level SC scores as demonstrations.

Implementation Details. We employ ChatGPT-3.5-turbo API as the sample generator and inference model with a temperature coefficient of 0. We used Nearest retrieval to obtain four examples from the source domain as demonstrations for sample generation. In each experimental setup, we synthesize 2000 samples. The settings of other LLMs are shown in the Appendix A.2. For robust learning state acquisition, we employ AdaE (Shi et al., 2023) as the small model and choose three checkpoints every ten epochs. We set α and β to $(\lambda + (1/labels))$, which $labels$ denotes the label number of different target domains. We set both λ_α and λ_β to 0.1. We use the Nearest method to retrieve 100 candidates for each input, and then rank 5 candidates using Avg Confidence. Each domain is run three times.

4.2 Main Results

The main results are given in Table 1. Regarding this, the observations and analyses are as follow:
(1) **Compared to traditional small models trained with supervised learning,** LLMs still exhibit limitations in CDSF. Although LLMs outperform the RCSF baseline in few-shot scenarios, the performance improvement remains marginal.
(2) **Compared to using source domain data as ICL examples,** leveraging generated data significantly improves the performance of LLMs across three retrieval methods: "Random" (8.19%), "BM25" (7.98%), and "Nearest" (8.09%). These

Model↓Domain→	ATP	GW	BR	PM	RB	SCW	SSE	Avg. F1
Small Model (Traning with Source Data)								
RCSF (Liu et al., 2022b)	54.35	63.49	65.36	53.51	36.51	69.20	33.50	55.76
AdaE (Shi et al., 2023)	61.13	42.35	69.87	36.24	33.25	70.81	34.06	49.67
LLM (ICL with Zero Data)								
ChatGPT	37.36 $_{0.21}$	63.93 $_{0.34}$	63.00 $_{1.24}$	67.47 $_{0.57}$	52.51 $_{0.31}$	52.21 $_{0.49}$	40.87 $_{0.41}$	53.91
LLM (ICL with Source Data)								
Random	53.44 $_{0.93}$	53.18 $_{0.75}$	67.29 $_{0.38}$	70.66 $_{1.31}$	48.74 $_{1.51}$	58.51 $_{0.60}$	31.16 $_{0.48}$	54.71
BM25 (Robertson and Zaragoza, 2009)	54.75 $_{0.45}$	55.67 $_{0.77}$	67.11 $_{0.81}$	71.61 $_{0.38}$	48.16 $_{0.33}$	59.53 $_{0.78}$	36.72 $_{0.56}$	56.22
Nearest (Liu et al., 2022a)	54.31 $_{0.09}$	56.20 $_{1.48}$	69.13 $_{0.92}$	70.60 $_{0.41}$	50.85 $_{0.57}$	60.24 $_{0.84}$	33.41 $_{0.40}$	56.39
LLM (ICL with Curated Data)								
Random	55.63 $_{1.31}$	60.31 $_{0.76}$	68.84 $_{0.94}$	71.28 $_{0.84}$	68.15 $_{0.69}$	68.36 $_{0.61}$	47.74 $_{1.37}$	62.90
BM25 (Robertson and Zaragoza, 2009)	56.62 $_{0.59}$	62.23 $_{0.84}$	71.63 $_{0.52}$	73.19 $_{0.85}$	69.44 $_{1.04}$	68.11 $_{0.34}$	48.21 $_{0.75}$	64.20
Nearest (Liu et al., 2022a)	55.81 $_{0.84}$	62.44 $_{0.11}$	69.62 $_{0.52}$	75.95 $_{0.69}$	70.75 $_{0.17}$	68.62 $_{0.26}$	48.17 $_{1.25}$	64.48
Nearest + SC ranking (Xie et al., 2024)	57.31 $_{0.38}$	61.29 $_{0.19}$	70.48 $_{0.22}$	73.31 $_{0.89}$	67.73 $_{0.25}$	69.52 $_{0.78}$	47.19 $_{0.31}$	63.83
Nearest + Ours	56.09 $_{0.70}$	64.13 $_{0.51}$	72.24 $_{0.68}$	**77.56** $_{0.59}$	**73.82** $_{0.46}$	68.78 $_{0.96}$	50.44 $_{0.67}$	66.15
LLM (ICL with Curated Data and SMP)								
Random	68.22 $_{0.51}$	66.78 $_{0.57}$	76.33 $_{0.94}$	72.65 $_{1.46}$	65.72 $_{1.31}$	71.04 $_{0.85}$	47.28 $_{0.18}$	66.86
BM25 (Robertson and Zaragoza, 2009)	67.24 $_{0.37}$	66.62 $_{1.63}$	77.23 $_{0.55}$	70.67 $_{0.83}$	65.58 $_{0.85}$	71.40 $_{0.61}$	48.93 $_{0.44}$	66.81
Nearest (Liu et al., 2022a)	70.95 $_{1.36}$	67.07 $_{1.09}$	**77.79** $_{0.54}$	70.70 $_{0.69}$	65.58 $_{0.18}$	71.02 $_{1.35}$	49.57 $_{0.75}$	67.53
Nearest + SC ranking (Xie et al., 2024)	71.17 $_{0.98}$	62.63 $_{0.63}$	73.21 $_{1.24}$	69.60 $_{0.43}$	57.90 $_{1.29}$	70.98 $_{0.51}$	47.74 $_{0.84}$	63.68
Nearest + Ours	**71.99** $_{0.52}$	**69.47** $_{1.07}$	76.72 $_{0.68}$	73.08 $_{0.54}$	66.38 $_{1.10}$	**71.48** $_{0.27}$	**50.68** $_{0.27}$	**68.54**

Table 1: Slot filling performance (F1-scores) of different models for different target domains on SNIPS. "ATP", "BR", "GW", "PM", "RB", "SCW" and "SSE" denote AddToPlaylist, BookRestaurant, GetWeather, PlayMusic, RateBook, SearchCreativeWork and SearchScreeningEvent, respectively. "Avg." denotes average, and "SMP" refers to Small Model Prediction. Best results are highlighted in **bold**. Right subscript numbers are standard deviations.

results demonstrate the effectiveness of incorporating high-quality synthetic data into ICL for LLMs. (3) **Compared to other retrieval methods,** our approach consistently achieves superior performance across various settings. Further analysis reveals that the "SC ranking" method exhibits a performance decline compared to its predecessor, which we attribute to two potential issues in the synthetic data: text distribution shift and erroneous labeling. These issues may degrade model consistency. In contrast, our method effectively mitigates these challenges by using small models to obtain highly confident synthetic examples, enabling the retrieval of more correct examples for LLMs.

(4) **The performance is further improved when leveraging predictions from small models.** However, a closer analysis reveals performance degradation in the "PM" and "RB" domains. This can be attributed to the relatively lower performance of small models in these domains, which may misguide the slot predictions of LLMs. In contrast, performance improvements are observed in most other domains, demonstrating the effectiveness of the small model prediction approach.

4.3 Further Ablation Studies

In Table 1, we present the main ablation studies about the effectiveness of synthetic data as examples, our rerank method and SMP. To validate the robustness of other components of our framework, we conduct additional ablation studies.

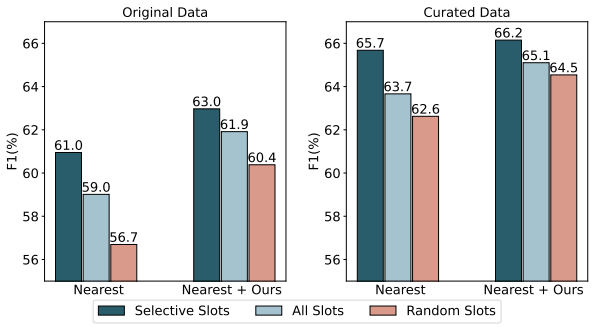


Figure 3: The performance comparison of different slot combination selection strategies.

Different Slot Combination Strategies. To validate the effectiveness of our slot combination selection method, we investigate two alternative approaches for data synthesis: (1) All Slots: directly selecting all slot types, and (2) Random Slots: randomly selecting multiple slot types as combination. As shown in Figure 3, both methods exhibit performance degradation. Specifically, the Random Slots performs worse, due to the loss of relevant slot types as combinations, such as "playlist" and "playlist owner.". This results in difficulty generating samples to learn the relevance of slots. Our method uses LLMs to select relevant slot combinations, which can effectively alleviate this issue.

Different Data Curation Strategies. Our per-

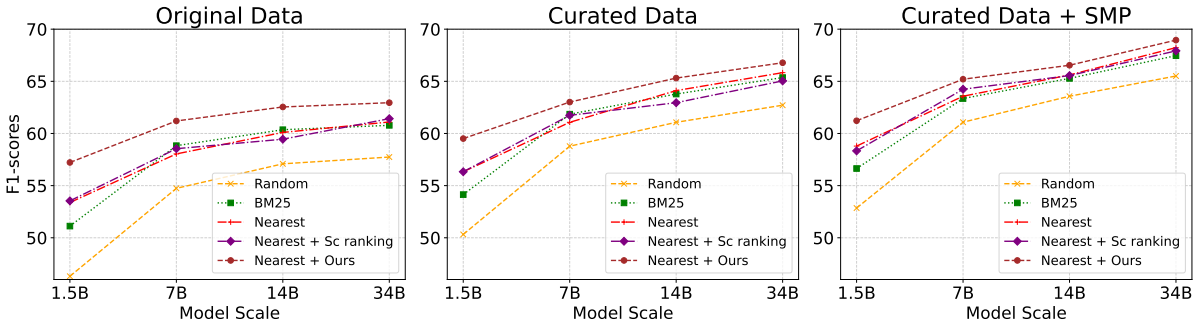


Figure 4: The average performance of seven target domains on SNIPS across different model scale settings.

Data Curation Strategy	F1
LLM (ICL with Original Data)	62.97
w/ Value-level replace	65.31
w/ Sentence-level filtering	65.63
w/ Value-level filtering	**66.15**

Table 2: Average F1-scores about different data curation strategies.

formance comparison of the three different data curation strategies is shown in the Table 2. All three methods achieve performance improvements. Specifically, the Value-level replace method show limited improvements, likely due to incorrect slot type replacements. Meanwhile, the coarse-grained Sentence-level filtering approach may discard too much useful information. In contrast, the token-level filtering method effectively retain more valuable information without introducing additional noise, leading to superior performance.

Different LLMs Performance. Figure 4 presents the average performance evaluated using various LLMs ranging in size from 1.5B to 34B, with full model details in the Appendix A.2. The results demonstrate that all ICL methods achieve better performance as the model size increases, whereas random ICL method tends to produce unstable results. Notably, our approach consistently outperforms previous methods. Specifically, for the Original Data setting, our method achieves the most significant performance improvement compared to other methods. This is attributed to the limited number of effective examples in this setting, where our approach demonstrates superior capability in retrieving useful examples. Additionally, as the experimental settings provide more information and the model scale increases, the performance gains become more pronounced, indicating that larger-scale models can better leverage the diverse and effective information provided by our framework.

Model↓Setting→	All	Seen	Unseen
Small Model (Traning with Source Data)			
AdaE	49.67	65.80	21.23
LLM (ICL with Zero Data)			
ChatGPT	53.90	52.22	53.68
LLM (ICL with Source Data)			
Nearest	56.39	59.60	46.14
LLM (ICL with Curated Data)			
Nearest + SC ranking	63.83	61.25	53.18
Nearest + Ours	66.15	62.36	58.36
LLM (ICL with Curated Data and SMP)			
Nearest + SC ranking	63.68	66.29	50.77
Nearest + Ours	**68.54**	**69.40**	56.61
- Unseen Slots Prediction	67.73	69.86	53.20
- Seen Slots Prediction	66.75	62.03	**60.15**

Table 3: Average F1-scores on seen and unseen slots.

4.4 Further Analysis

4.4.1 Slot Generalization

In the cross-domain scenario, since the model's limited exposure to the knowledge of unseen slots, it encounters challenges when attempting to fill them. The meanings of seen slots may vary across different domains, leading to misclassification. We divide the dataset into seen and unseen groups and evaluate them separately. From Table 3, our method has high performance on different slots. Specifically, ChatGPT performs high on SNIPS for unseen slots. Our methods surpasses it by 4.68% and 2.93%. Otherwise, our method surpasses AdaE by 3.6% in seen slots. Although SMP adversely affects the performance of certain slot predictions, leading to a decline in the performance of unseen slots, the substantial improvements in seen slots result in an overall performance gain. However, exclusive reliance on seen slot predictions results in suboptimal performance for unseen slots. We posit that this limitation arises from the LLM's inher-

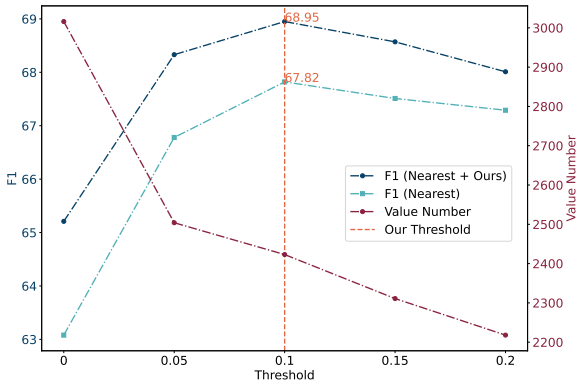


Figure 5: The value number, and model performance on different thresholds on SNIPS.

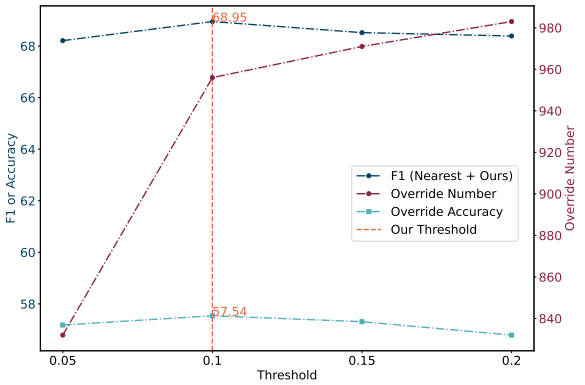


Figure 6: The override accuracy, override quantity, and model performance on different thresholds on SNIPS.

ent classification bias, where tokens from unseen slots are systematically misassigned to seen categories. Our method overcomes this limitation by leveraging AdaE's capability in accurately identifying unseen slots, which significantly improves the model's overall robustness. In summary, our approach is better able to combine the strengths of small model and LLMs in various slots and significantly compensate for the shortcomings.

4.4.2 Impact of Different Thresholds

In this section, we conduct studies on different thresholds in our framework. We adjust the λ of $(\lambda + (1/labels))$ to fine-tune thresholds. The analysis is in the Appendix B and is as follows:

Impact of Value-level filtering Threshold. Figure 5 presents the impact of different λ_α values on the quantity of slot values in D_{cur} and the performance of the model. We observe that as the threshold increases up to 0.1, the model performance improve significantly. The improvement is accompanied by a sharp decrease in the quantity of slot value pairs. Beyond the threshold of 0.1, although the quantity continues to decrease, the model performance keeps declining. This implies that filtering out low-quality slot value pairs with appropriate confidence threshold is necessary to improve LLMs' performance.

Impact of High or low Confidence Threshold. We investigate the impact of varying the threshold λ_β on three key aspects as illustrated in Figure 6. We observe that as the threshold increases up to 0.1, both the accuracy of overridden predictions and the model performance improve significantly. The improvement is accompanied by a sharp increase in the quantity of overridden instances. Beyond the threshold of 0.1, although the quantity of overridden instances continues to rise, the accuracy of overridden declines, leading to a slight degradation in model performance. This suggests that attempting to override high-confidence slot predictions negatively impact model performance.

Model	F1
(Wu et al., 2022)	75.06
AdaE (Shi et al., 2023)	75.29
ours (w/o SMP)	75.82
ours (w/o Curated Data)	78.12
ours	**79.54**

Table 4: F1-scores on cross-domain NER dataset.

4.4.3 Case Study

Figure 7 presents a case study, where "Gold label" denotes the ground truth annotation for the test input. The baseline model "With source data" misclassifies "elise and alma" as "party size description." In contrast, the "With Original Data" approach correctly predicts the slot due to learning from relevant slot value pair in the demonstrations. However, it still makes an incorrect prediction influenced by the noisy slot-value pair "time range | sunday". Ours "With Curated Data" mitigates the impact of such erroneous slot-value pairs and corrects the prediction. Finally, by incorporating SMP, LLMs recognizes that "brasserie" is semantically closer to "restaurant" but does not belong to the "restaurant name" type, thus predicting the correct slot as "restaurant type."

4.5 Results on Cross Domain NER

Following AdaE (Shi et al., 2023), we evaluate our framework in a cross-domain NER setting. The statistical information of the NER setting is shown in

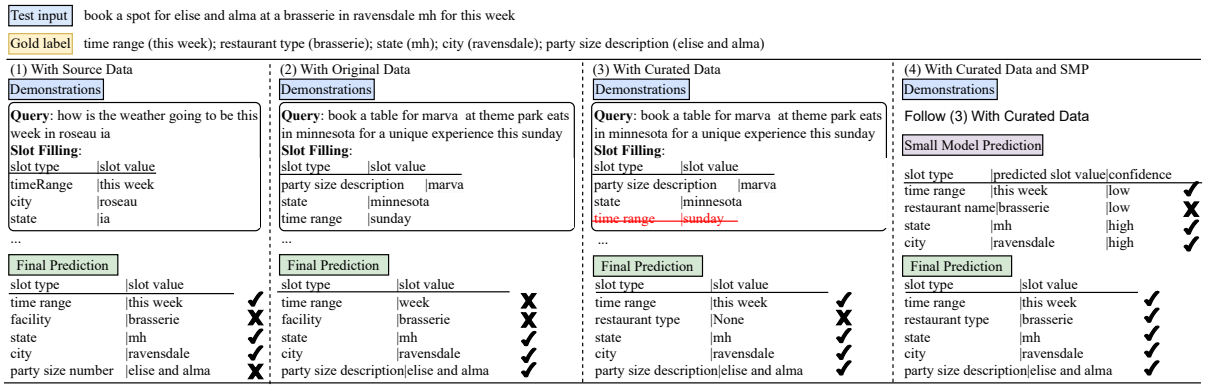


Figure 7: An example of four type of different information on the "GetWeather" target domain.

Appendix A.1. As shown in Table 4, our approach demonstrates strong adaptability to this scenario, outperforming previous competitive baselines. Furthermore, the integration of curated data and small model predictions contributes to performance improvements.

5 Conclusion

To fully leverage the complementary strengths of small models and LLMs, we propose a training-free, small model-assisted framework for cross-domain zero-shot slot filling with LLMs. Our framework significantly improves performance across eight corss-domain settings on two datasets. Comprehensive experimental analysis demonstrates the effectiveness and robustness of our framework.

6 Acknowledgements

This work was supported by the National Natural Science Foundation of China (62306119 and 71472068) and the Science and Technology Projects in Guangzhou (2025A04J3436).

Limitations

This work focuses on exploring the cross-domain zero-shot slot filling or NER task. The investigation of this paradigm on other IE tasks has not been studied yet. We explore the commonly-used self-consistency method to compare with our approach. Additionally, there may exist alternative methods to evaluate the quality of synthetic samples, which could be investigated in future work.

References

Sanjeev Arora, Simon S. Du, Wei Hu, Zhiyuan Li, and Ruosong Wang. 2019. Fine-grained analysis of optimization and generalization for overparameterized two-layer neural networks. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 322–332.

Devansh Arpit, Stanislaw Jastrzebski, Nicolas Ballas, David Krueger, Emmanuel Bengio, Maxinder S. Kanwal, Tegan Maharaj, Asja Fischer, Aaron C. Courville, Yoshua Bengio, and Simon Lacoste-Julien. 2017. A closer look at memorization in deep networks. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 233–242.

Ankur Bapna, Gokhan Tür, Dilek Hakkani-Tür, and Larry Heck. 2017. Towards Zero-Shot Frame Semantic Parsing for Domain Scaling. In *Proc. Interspeech 2017*, pages 2476–2480.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. *Preprint*, arXiv:2005.14165.

Alice Coucke, Alaa Saade, Adrien Ball, Théodore Bluche, Alexandre Caulier, David Leroy, Clément Doumouro, Thibault Gisselbrecht, Francesco Caltagirone, Thibaut Lavril, Maël Primet, and Joseph Dureau. 2018. Snips voice platform: an embedded spoken language understanding system for private-by-design voice interfaces. *CoRR*, abs/1805.10190.

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.

Keqing He, Jinchao Zhang, Yuanmeng Yan, Weiran Xu, Cheng Niu, and Jie Zhou. 2020. 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).

Mia Mohammad Imran, Preetha Chatterjee, and Kostadin Damevski. 2024. Uncovering the causes of emotions in software developer communication using zero-shot llms. In *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, ICSE '24, New York, NY, USA. Association for Computing Machinery.

Gakuto Kurata, Bing Xiang, Bowen Zhou, and Mo Yu. 2016. Leveraging sentence-level information with encoder LSTM for semantic slot filling. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 2077–2083.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*.

Yongchan Kwon, Joong-Ho Won, Beomjoon Kim, and Myunghee Cho Paik. 2020. Uncertainty quantification using bayesian neural networks in classification: Application to biomedical image segmentation. *Comput. Stat. Data Anal.*, 142.

Sungjin Lee and Rahul Jha. 2019. Zero-shot adaptive transfer for conversational language understanding. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pages 6642–6649.

Aitor Lewkowycz, Anders Johan Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Venkatesh Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, Yuhuai Wu, Behnam Neyshabur, Guy Gur-Ari, and Vedant Misra. 2022. Solving quantitative reasoning problems with language models. In *Advances in Neural Information Processing Systems*.

Xuefeng Li, Liwen Wang, Guanting Dong, Keqing He, Jinzheng Zhao, Hao Lei, Jiachi Liu, and Weiran Xu. 2023. Generative zero-shot prompt learning for cross-domain slot filling with inverse prompting. In *Findings of the Association for Computational Linguistics: ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 825–834.

Yuanzhi Li, Tengyu Ma, and Hongyang R. Zhang. 2020. Learning over-parametrized two-layer neural networks beyond NTK. In *Conference on Learning Theory, COLT 2020, 9-12 July 2020, Virtual Event [Graz, Austria]*, volume 125 of *Proceedings of Machine Learning Research*, pages 2613–2682.

Zhenyu Li, Sunqi Fan, Yu Gu, Xiuxing Li, Zhichao Duan, Bowen Dong, Ning Liu, and Jianyong Wang. 2024. Flexkbqa: A flexible llm-powered framework for few-shot knowledge base question answering. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(17):18608–18616.

Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. 2022a. What makes good in-context examples for GPT-3? In *Proceedings of Deep Learning Inside Out (DeeLIO 2022): The 3rd Workshop on Knowledge Extraction and Integration for Deep Learning Architectures*, pages 100–114, Dublin, Ireland and Online.

Jian Liu, Mengshi Yu, Yufeng Chen, and Jinan Xu. 2022b. Cross-domain slot filling as machine reading comprehension: A new perspective. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 30:673–685.

Man Luo, Xin Xu, Zhuyun Dai, Panupong Pasupat, Mehran Kazemi, Chitta Baral, Vaiva Imbrasaite, and Vincent Y Zhao. 2023. Dr.icl: Demonstration-retrieved in-context learning. *Preprint*, arXiv:2305.14128.

Qiaoyang Luo and Lingqiao Liu. 2023. Zero-Shot Slot Filling with Slot-Prefix Prompting and Attention Relationship Descriptor. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(11):13344–13352.

Xinxi Lyu, Sewon Min, Iz Beltagy, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2023. Z-ICL: Zero-shot in-context learning with pseudo-demonstrations. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2304–2317, Toronto, Canada.

Bhavana Dalvi Mishra, Oyvind Tafjord, and Peter Clark. 2022. Towards teachable reasoning systems: Using a dynamic memory of user feedback for continual system improvement. *Preprint*, arXiv:2204.13074.

Keqin Peng, Liang Ding, Yancheng Yuan, Xuebo Liu, Min Zhang, Yuanxin Ouyang, and Dacheng Tao. 2024. Revisiting demonstration selection strategies in in-context learning. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 9090–9101.

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.

Stephen E. Robertson and Hugo Zaragoza. 2009. The probabilistic relevance framework: BM25 and beyond. *Found. Trends Inf. Retr.*, 3(4):333–389.

Souvika Sarkar, Dongji Feng, and Shubhra Kanti Karmaker Santu. 2023. Zero-shot multi-label topic inference with sentence encoders and LLMs. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 16218–16233, Singapore.

Alexander Scarlatos and Andrew Lan. 2024. Reticl: Sequential retrieval of in-context examples with reinforcement learning. *Preprint*, arXiv:2305.14502.

Nabeel Seedat, Jonathan Crabbé, Ioana Bica, and Mihaela van der Schaar. 2022. Data-iq: Characterizing subgroups with heterogeneous outcomes in tabular data. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Nabeel Seedat, Nicolas Huynh, Boris van Breugel, and Mihaela van der Schaar. 2024. Curated LLM: synergy of llms and data curation for tabular augmentation in low-data regimes. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*.

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, Florence, Italy.

Yuanjun Shi, Linzhi Wu, and Minglai Shao. 2023. Adaptive end-to-end metric learning for zero-shot cross-domain slot filling. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 6291–6301, Singapore.

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, Online and Punta Cana, Dominican Republic.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*.

Yu Wang, Yilin Shen, and Hongxia Jin. 2018. A bi-model based RNN semantic frame parsing model for intent detection and slot filling. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 2 (Short Papers)*, pages 309–314.

Xiao Wei, Yuhang Li, Yuke Si, Longbiao Wang, Xiaobao Wang, and Jianwu Dang. 2024. A prompt-based hierarchical pipeline for cross-domain slot filling. *IEEE ACM Trans. Audio Speech Lang. Process.*, 32:3061–3075.

Linzhi Wu, Pengjun Xie, Jie Zhou, Meishan Zhang, Ma Chunping, Guangwei Xu, and Min Zhang. 2022. Robust self-augmentation for named entity recognition with meta reweighting. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4049–4060, Seattle, United States.

Zhiyong Wu, Yaoxiang Wang, Jiacheng Ye, and Lingpeng Kong. 2023. Self-adaptive in-context learning: An information compression perspective for in-context example selection and ordering. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 1423–1436.

Tingyu Xie, Qi Li, Yan Zhang, Zuozhu Liu, and Hongwei Wang. 2024. Self-improving for zero-shot named entity recognition with large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 2: Short Papers)*, pages 583–593, Mexico City, Mexico.

Yang Yan, Junda Ye, Zhongbao Zhang, and Liwen Wang. 2022. AISFG: Abundant information slot filling generator. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4180–4187, Seattle, United States.

Junwen Zhang and Yin Zhang. 2023. Hierarchicalcontrast: A coarse-to-fine contrastive learning framework for cross-domain zero-shot slot filling. In *Findings of the Association for Computational Linguistics: EMNLP 2023, Singapore, December 6-10, 2023*, pages 14483–14503.

A Other Experiment Settings

A.1 Setting of NER Task

Following AdaE (Shi et al., 2023), we conduct cross-domain evaluations on the CBS SciTech News test set, using all data of CoNLL-2003 as the source-domain dataset. All other experimental

Split↓ Dataset→	CoNLL-2003	CBS SciTech News
Train	15.0k	-
Dev	3.5k	-
Test	5.6k	2.0k

Table 5: The statistical information of the NER task

settings remain consistent with the main experiments. The statistical information of the NER task is shown in Table 5.

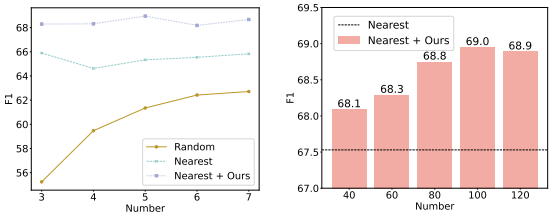
A.2 Other LLMs Setting

To investigate the generalizability of our framework, we utilize other LLMs to generate data and inference for all test inputs, such as Qwen1.5-34B-Chat-AWQ[1], Qwen1.5-14B-Chat[2], Qwen2.5-7B-Instruct[3] and Qwen2-1.5B-Instruct[4]. For these models, we maintain a similar experimental setup to ChatGPT. We use the same prompt template as Figure 2 and the same temperature coefficient of 0 to maintain the stability of the output. For the decoding strategy parameters, we set the top-k to 5 and top-p to 0.9. Additionally, we utilize vllm (Kwon et al., 2023) to reduce the memory footprint and accelerate inference. Therefore, we can generate data and inference on two RTX 3090.

B Impact of Other Thresholds

Impact of ICL Example Quantities. We gradually increase the number of in-context examples (denoted as k) from 3 to 7. The results are presented in Figure 8(a). As shown, increasing the number of in-context examples within a certain range further improves model performance with our method. This suggests that additional examples can effectively leverage the potential of LLMs. However, providing an excessive number of examples may degrade performance, due to the introduction of noise in LLMs' generation process. Notably, our method consistently outperforms other baselines across different values of k, demonstrating its robustness and generalizability.

Impact of Condidate Example Quantities. As mentioned earlier, our method consists of two modules: Nearest selection and Avg Confidence reranking. The selection module reduces the search space of in-context examples to accelerate the overall process. Therefore, we investigate the impact



(a) In-context examples (b) Condidate examples

Figure 8: The impact of two types of example quantities on model performance.

of the number of candidates selected by the Nearest module. The results in Figure 8(b) illustrate the performance with five in-context examples using the Qwen1.5-34B model. We observe that our method consistently outperforms the Nearest baseline, and increasing the number of candidates further augments performance. However, when the number of candidates exceeds 120, model performance declines, likely due to the introduction of noisy samples from an large candidate pool.

C Prompts

C.1 Data Generation Prompts

We show the prompts used to generate data in Table 6. Multiple slot type combinations are chosen according to the target domain and its associated label set in the first stage. These combinations are then utilized to create prompt templates for the second stage, facilitating the generation of varied synthetic samples. In the **Slot Combinations Selection**, we introduce two conditions within the prompts: "Include only slot combinations that appear together in a user query" and "Provide reasoning for your selection". Additionally, "Ensure all slots have been used" is introduced to ensure combination diversity. Finally, we construct the final prompt by combining some examples selected from the source data to fully harness the few-shot capabilities of LLMs. In the **Data Synthesis**, we create a Chain of Thought (COT) prompt in conjunction with target domain details to generate samples in two steps. We use natural language to describe the task process and provide examples for LLMs to learn the relationships between certain slots and their values. The prompt involves initially generating values for each slot within the slot combinations, which are subsequently used to create synthetic samples. To prevent the inclusion of any extraneous or irrelevant slot values in the synthesized sample, we add

[1]https://huggingface.co/Qwen/Qwen1.5-32B-Chat-AWQ
[2]https://huggingface.co/Qwen/Qwen1.5-14B-Chat
[3]https://huggingface.co/Qwen/Qwen2.5-7B-Instruct
[4]https://huggingface.co/Qwen/Qwen2-1.5B-Instruct

Stage 1: Slot Combinations Selection

Task Description:
I would like you to help me select some slot combinations from a set of slot types.
Conditions:
1. The values corresponding to the slots in the combination can appear in the same user query.
2. The user query only includes the selected slots, avoid including other slots.
3. Provide the reasoning process along with the slot combinations.
In-Context Examples:
(domain): AddToPlaylist
(slot type set): music_item; playlist_owner; entity_name; playlist; artist
1. (slot types selected): playlist_owner; entity_name; playlist
1. (inference): "add sugarolly days to my list your favorite slaughterhouse" only include selected slots.
2. (slot types selected): music_item
2. (inference): "please add this this tune to the playlist" only include selected slots.
Input:
Now, given a new domain: PlayMusic. and given a new list of slot types: Slot Type Set of PlayMusic.
Please select 10 slot combinations and their corresponding reasoning processes.

Stage 2: Data Sythesis

Task Description:
First, given a selected set of slot types and a domain, you need to provide a corresponding slot value for each slot type. Then, using these related slot values, generate a user query with a given domain.
Conditions:
1. The user query cannot contain words corresponding to slots other than the selected slot.
2. Generate samples as diverse as possible. In-Context Examples:
case1:
(domain): AddToPlaylist
(slot type combination) :artist;music_item;playlist
Fisrt, (type</res>value): artist</res>kj 52;music_item</res>track;playlist</res>te quiero
Then, (generate sample): add a kj 52 track to the te quiero playlist
Input:
Now, given a new domain: PlayMusic. and given a new slot types: A Slot Combinations.
Please help me generate 20 samples of the domain using these slot types.

Table 6: Two-stage data generation prompt.

the condition "The sentence can only contain slot combinations of slot values." to the prompt. In the end, we obtain a set of generated samples for the target domain.

C.2 Slot filling prompt

The overall prompt template, as shown in Table 7, primarily consists of three components: task description, demonstrations, and small model prediction results.

Task description
When I provide you a list and a query, please answer with the following format:
A table containing two columns with the column headers as (slot type, slot value).
You need to identify potential slot types from the query and fill in the corresponding slot values for each slot type from the query.
The slot types should come from the provided list, and the slot values should come from the query.
A token only has a slot type. For example,
Demonstrations
Given a query: add the song perfect to my wedding playlist.
answer:
slot type	real slot value
———	———————
playlist owner	my
playlist	wedding

Given a query: include the dance track in the party mix playlist:
answer:
slot type	real slot value
———————	———————————
entity name	dance track
playlist	party mix

... (Omit some examples)
Small model prediction results
slot type	predicted slot value	reliability
———	———————————	———
playlist	stress relief	high
The low reliability indicate that uncertain about the slot type.
Output
Now, I give you a list: ['music_item', 'playlist_owner', 'entity_name', 'playlist', 'artist'],
and a new query: add camille to the this is lady antebellum playlist.
Please answer:

Table 7: Slot filling prompt.