kNN-ICL: Compositional Task-Oriented Parsing Generalization with Nearest Neighbor In-Context Learning

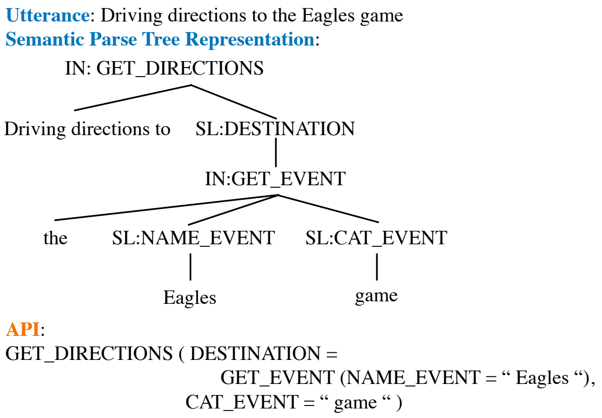
Anonymous ACL submission

Abstract

Task-Oriented Parsing (TOP) enables conversational assistants to interpret user commands expressed in natural language, transforming them into structured outputs that combine elements of both natural language and intent/slot tags. Recently, *Large Language Models (LLMs)* have achieved impressive performance in synthesizing computer programs based on a natural-language prompt, mitigating the gap between natural language and structured programs. Our paper focuses on harnessing the capabilities of LLMs for semantic parsing tasks, addressing the following three key research questions: 1) *How can LLMs be effectively utilized for semantic parsing tasks?* 2) *What defines an effective prompt?* and 3) *How can LLM overcome the length constraint and streamline prompt design by including all examples as prompts?* We introduce k Nearest Neighbor In-Context Learning (kNN-ICL), which simplifies prompt engineering by allowing it to be built on top of any design strategy while providing access to all demo examples. Extensive experiments show that: 1) Simple ICL without kNN search can achieve a comparable performance with strong supervised models on the TOP tasks, and 2) kNN-ICL significantly improves the comprehension of complex requests by seamlessly integrating ICL with a nearest-neighbor approach. Notably, this enhancement is achieved without the need for additional data or specialized prompts.

1 Introduction

Task-Oriented Parsing (TOP), which aims to translate the natural-language commands from users into specific actions, such as booking a restaurant, is an essential component in conversational systems (Chen et al., 2020; Budzianowski et al., 2018; Wu et al., 2023). Over the past few years, a wide range of methods have been developed, ranging from rule-based methods (Zelle and Mooney, 1996; Dong and Lapata, 2018) to neural program synthesis methods (Shin et al., 2019; Mansimov and Zhang, 2022; Drozdov et al., 2022).



```
GET_DIRECTIONS ( DESTINATION =
                    GET_EVENT (NAME_EVENT = " Eagles "),
              CAT_EVENT = " game " )
```

Figure 1: Reduction of an utterance from Topv2 with semantic parse tree to Python style API.

The recent advancements in TOP involve formulating the task as a sequence-to-sequence problem, relying on a wealth of labeled data. This process typically entails feeding user utterances into Transformer models (Liu et al., 2019; Devlin et al., 2018; Raffel et al., 2020) and designing specialized Transformer architectures to generate structured outputs that combine natural language with intent and slot tags like "IN:" and "SL:". Some alternative approaches attempt to reframe TOP as conventional tasks, such as canonical paraphrasing (Shin et al., 2021) and abstractive question answering (Zhao et al., 2022). These approaches aim to reduce the model's burden by eliminating the need to rationalize non-linguistic labels that were not part of pre-training. However, in real-world scenarios, obtaining high-quality training examples with expert annotations can be challenging. Additionally, API documentation, which contains valuable information, often remains underutilized within the fine-tuning paradigm.

On the contrary, *Large Language Models (LLMs)* excel in challenging few-shot scenarios, where they only need a few examples to generate the desired output and can also understand extensive API documentation. In this paper, we delve into an examination and analysis of the performance of ICL in the context of TOP. Besides, following a comprehensive review of prompt design strategies, we introduce kNN-ICL, which offers adaptable integration with any prompt design strategy, further augmenting model performance.

We conducted experiments using three representative models: GPT-Neox-20B (Black et al., 2022), CodeGen-16B-Multi (Shin et al., 2019), and Codex (Chen et al., 2021) code-davinci-002 to assess the performance of LLMs in TOP. Our research revolves around three key questions:

Question 1: How can we effectively leverage LLMs for TOP tasks? We transform TOP into a code generation task, mapping semantic parse trees to Python code (referred to as 'API' hereafter) to align with LLMs' output format, as shown in Figure 1.

Question 2: What constitutes proper prompt design for TOP using LLMs? We analyze ICL performance under various prompt design strategies, including factors: API documentation and exemplar selection methods. Our findings include: API documentation benefits the more powerful CODEX model but can be distracting for the less-capable CODEGEN and GPT-NEOX models. Unsupervised demo selection proves most effective with ICL, thanks to its flexibility in capturing semantically similar examples holistically.

Question 3: How can we overcome the LLM's length constraint and simplify prompt design by including all examples as prompts? Instead of solely focusing on LLM performance through complex prompt design strategies, we draw inspiration from the retrieval language model approach (Khandelwal et al., 2019, 2021) and introduce kNN-ICL. kNN-ICL enables LLMs to access all available demo exemplars during inference, harnessing synergy between the copy mechanism and the labeling task. We tackle two key challenges. First, how to infuse TOP knowledge into LLMs, considering their limited domain expertise due to the scarcity of labeled data? We address this by including a few demos as kNN-ICL prompts, guiding LLMs to follow the desired output pattern. Second, how to retrieve the nearest neighbor when there's a representation discrepancy between the kNN-ICL datastore and the LLM prompt? To address this challenge, instead of using the hidden state h_t directly from the LLM as the current step representation, we utilize a combination of the target utterance and previously generated words from the LLM at time step t. This ensures consistency in representation between the kNN-ICL datastore and LLMs, offering flexibility in adapting to various prompt design strategies.

In summary, our contributions can be distilled into three key aspects: 1). We examine prompt design strategies for LLMs in the context of TOP by framing TOP as a code generation task. We conclude that similarity-based demo selection is effective for both black-box and open-source models, with stronger LLMs benefiting more from documentation; 2). We introduce kNN-ICL to address the length constraints of LLMs, offering flexibility to integrate with any prompt design strategy; and 3). Extensive experimental results showcase that kNN-ICL can enhance the generation of better-nested API structures by leveraging guidance from nearest neighbors.

2 Methodology

In this section, we outline the methodology employed in this study. Our approach consists of two key components: (1) Prompt Design for Semantic Parsing (Section 2.3): We begin by crafting an effective prompt for TOP. This involves varying two crucial elements within the prompt: API documentation and exemplar selection strategies. (2) Integration with kNN-ICL (Section 2.4): we then integrate all the exemplars to LLM in kNN-ICL. This integration allows us to harness the collective knowledge from all exemplars within the demo pool, enhancing the generation of the semantic parse API.

2.1 Preliminaries

In-Context Learning (ICL), which involves no parameter updates or fine-tuning for downstream tasks, has demonstrated competitiveness across multiple *Natural Language Understanding (NLU)* and *Natural Language Generation (NLG)* tasks. Typically, prompts are designed as task instructions, often comprising a few exemplars paired with input-output examples retrieved from a demo pool. However, due to the limited input length of LLMs, the number of exemplars is constrained.

More recently, the concept of retrieval language models (Khandelwal et al., 2019; Borgeaud et al., 2022) has emerged. In this paradigm, language models have access to extensive corpora and em-

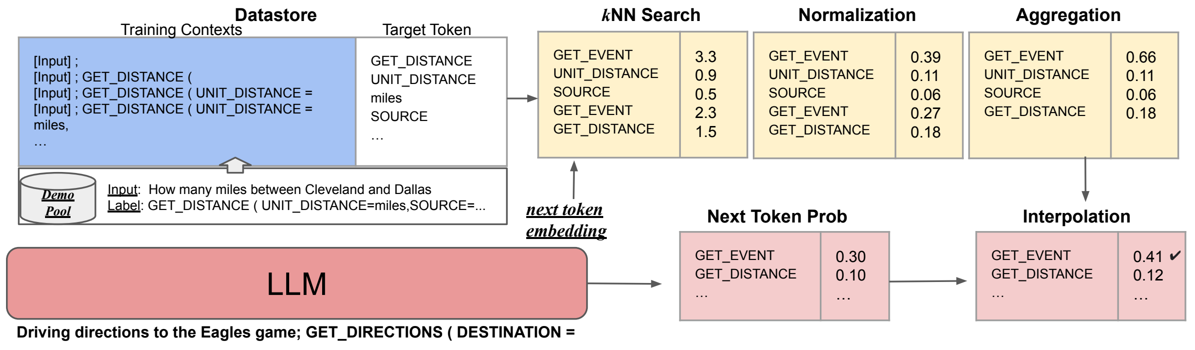


Figure 2: An illustration of kNN-ICL at decoding time step t. The input to LLM is the user utterance "*Driving directions to the Eagles game*" combined with the previously generated output "*GET_DIRECTIONS (DESTINATION =*". The target token "*GET_EVENT*" is generated as output.

ploy explicit memory mechanisms for text generation. This approach has proven to enhance the capabilities of language models significantly.

2.2 Reduction to Code Generation

We utilize pre-trained code style knowledge, as found in models like CODEX, CODEGEN, and GPT-NEOX, trained on extensive code style corpora. We transform the API representations into Python code style, mapping the root node's intent name from the TOP semantic parse tree to the outermost Python function name. The tree branches serve as variable-value pairs within the Python function structure, enabling effective use of pre-trained code style knowledge.

2.3 Prompt Design for Semantic Parsing

The fundamental prompt design for TOP consists of three core elements: "`[API documentation]+[exemplars]+[target utterance]`." The target utterance represents the expected prediction by the LLMs, based on the provided prompt. We will delve into the first two components in the subsequent discussion.

2.3.1 API Documentation

In each domain, the API name, which includes intent and slot names from the TOP dataset, is represented symbolically, like *GET_LOCATION*. The API document or description offers natural language explanations of the domain service, aiding pre-trained LLMs in comprehension.

2.3.2 Demo Selection

We explore three exemplar selection strategies.
Random Selection Exemplars are randomly drawn from the demo pool to create the prompt.
Unsupervised Selection with SentenceBERT Utilizing SentenceBERT, we generate sentence-level embeddings for both the target utterance and all exemplar utterances. We rank the cosine similarity scores for all demos in the pool and select the top-k most similar exemplars. In experiments, we set the number of exemplars (m) to 10 for all three models.
Supervised Selection with Paraphrasing In contrast to SentenceBERT, we train a more precise classifier to rank utterances based on the similarity at the outermost intent level. We frame the similarity ranking as a paraphrasing task. To construct pairwise data for the paraphrase model, we label `[a, b]` as `True` if `a` and `b` share the same intent name from a parent node of the semantic parse tree, where `b` is sampled from the demo pool. For each utterance `a`, the ratio of constructed `True` pairs to `False` pairs is 1 to 5. During inference, we use the probability of predicting the label `True` as the ranking score.

2.4 k-Nearest Neighbour In-Context Learning

LLMs face a limitation in fully utilizing the exemplar information from the demo pool, making the demo selection strategy a critical factor influencing generation quality for ICL. Here, we introduce a comprehensive model, kNN-ICL, designed to enable ICL to consider all demo exemplars from the demo pool, irrespective of the length constraint imposed by LLMs.

An Overview of kNN-ICL is shown in Figure 2: kNN-ICL leverages both LLMs' predictions and k nearest neighbor search to generate the final prediction. This involves probability distribution over the LLMs' vocabulary during the decoding process and the retrieval of the top-k nearest neighbors from the datastore.

A challenge for kNN-ICL arises from the need to copy a span from the target utterance as a slot value, a requirement of the TOP task. Unfortunately, traditional kNN-LM (Khandelwal et al., 2019) or kNN-MT (Khandelwal et al., 2020) methods do not naturally support this. Furthermore, the constrained size of the demonstration pool may not encompass all potential slot values during the inference process. To tackle these challenges, we undertake the following steps:

LLM prompt kNN-ICL bridges the information formality gap between LLM and the datastore. ICL has demonstrated promising copying ability from target utterances. Hence, we prompt the LLMs with a combination of exemplars and the target utterance to compensate for the limited datastore records, ensuring that the slot value is grounded in the target utterance.

Datastore Creation We create a datastore offline, comprising multiple key-value pairs. The key represents the contextualized representation of the input sentence encoded by LLM, and the value corresponds to the subsequent token of the input sentence. Given a set of training contexts denoted as C and a target represented as T, the formulation of the datastore D is as follows:

$$D \overset{\text{def}}{=} (\text{K}, \text{V}) = \{(f(c_i), t_i) | \\ \forall c_i \in C, \forall t_i \in T, (c_i, t_i) \in (C, T)\} \tag{1}$$

For each example in the training set, we store multiple datastore records containing only tokens corresponding to the semantic parsing labels as targets.

Similarity Search Another challenge arises because the hidden state produced by LLM at time step t resides in a different representation space than the datastore. To address this, at time step t when generating token y_t, we calculate the current hidden state representation as a combination of the target utterance s and the previously generated API tokens $y_{1...i-1}$. This alignment ensures compatibility with the datastore representation space. Given the limited size of the datastore in TOP, the distribution of retrieved k nearest neighbors tends to be skewed. To mitigate overfitting to the most similar retrieval, we introduce a temperature parameter $Temp$ to flatten the distribution.

$$p_{kNN}(y_t|c, y_{1:t-1}) \propto \\ \sum_{(k_j,v_j)\in N} 1_{y_t=v_j} \exp\left(\frac{-Dis(k_j, f(c, y_{1:t-1}))}{Temp}\right) \tag{2}$$

Interpolation The decoding process involves interpolating the k nearest neighbor from the datastore with the language model distribution. To enable the model to accurately predict slot values by copying spans from the target utterance, we employ the full vocabulary from LLM rather than restricting it to the intersection between LLM vocabulary and the k nearest neighbor vocabulary.

$$p(y_t|x, y_{1:t-1}) = \lambda p_{kNN}(y_t|c, y_{1:t-1}) \\ + (1-\lambda)p_{lm}(y_t|x, y_{1:t-1}) \tag{3}$$

kNN-ICL vs. kNN-MT kNN-ICL is a generalization of kNN-MT for conditional generation tasks using In-Context Learning (ICL), seamlessly integrating with LLMs. It introduces some significant differences compared to kNN-MT. First, the LLM conditions not only on the source sequence but also on the provided prompt, which typically includes a few selected demonstrations. This enables the LLM to align more closely with the pattern of the target demonstration. Second, while the datastore in kNN-MT is constructed solely from the source and target mappings, the LLM produces representations that are conditioned on the entire prompt and target. This results in a discrepancy when selecting the k nearest neighbors. To address this, we rely on representations derived from both the source and the previously generated tokens in the target to ensure consistency during similarity search.

3 Experiments

In this section, we address the following key research questions through our experiments: (1) What constitutes an effective prompt design to enhance the performance of LLMs? (2) How does ICL compare to state-of-the-art supervised models on TOP tasks? (3) What impact does kNN-ICL, which incorporates all available demo examples from the demo pool through interpolation, have on prediction quality?

3.1 Dataset, LLMs, and Evaluation

We conduct experiments on the TOPv2 dataset (Chen et al., 2020), an extension of the TOP (Gupta et al., 2018) dataset designed for voice assistants. TOPv2 encompasses 8 domains, including 6 new additions, providing a diverse range of task-oriented semantic parsing examples. Due to the resource-intensive nature of running LLMs, we select 4 representative domains for our testing: Navigation, Reminder, Alarm, and

Exemplars	Doc	GPT-NeoX	CodeGen	Codex
Random	✗	2.02	3.54	19.20
Unsupervised	✗	**6.23**	**10.27**	36.03
Supervised	✗	4.04	9.60	**37.54**
Random	✓	1.68	3.37	26.43
Unsupervised	✓	4.04	7.91	39.23
Supervised	✓	**4.04**	**9.43**	**41.25**

Table 1: Exact Match results for TOPv2 dataset on Navigation domain using LLMs with varying prompt components.

Weather. These domains were chosen based on their complexity, with Navigation and Reminder representing more intricate domains with complex nested semantic parses and a larger number of intent slot names, while Alarm and Weather offer relatively simpler examples with fewer complexities in terms of intent and slot names.

For our experiments, we employ GPT-Neox-20B (Black et al., 2022), CodeGen-16B-Multi (Shin et al., 2019), and Codex (Chen et al., 2021) (`code-davinci-002`) to assess the performance of LLMs in ICL. GPT-Neox and CodeGen models are publicly available and can be hosted locally, whereas Codex is accessible solely through a commercial API provided by OpenAI. Codex cannot be downloaded and implemented directly with our proposed method kNN-ICL. Hence, we employ CodeGen and GPT-Neox for experimenting with kNN-ICL.

We evaluate model performance using exact match criteria, where a prediction is considered correct (scored as 1) if it matches the ground-truth precisely and incorrect (scored as 0) if there is any discrepancy between the prediction and the ground-truth. This evaluation metric disregards the order of the semantic parse tree branches, focusing solely on the correctness of the prediction.

3.2 Baselines

We consider three sets of baselines. **Supervised State-of-the-Art Models**: To establish a performance benchmark for LLMs on TOP, we include supervised state-of-the-art models. RINE (Mansimov and Zhang, 2022) introduces a recursion insertion-based encoder tailored for TOP, breaking the task into smaller steps where each step predicts either an intent or slot name along with its starting and ending positions. CodeT5 (Wang et al., 2021), a unified pre-trained encoder-decoder model based on T5, showcases strong performance in code generation tasks. **ICL**: Unless specified otherwise, we will use ICL with randomly selected demos as the prompt. **kNN-LM**: In this baseline, we use kNN-LM without the inclusion of exemplar prompts for LLMs.

3.3 Implementation Details

For ICL, we utilize 10 exemplars to construct the prompt for LLMs. In the case of kNN-ICL, we explore a range of parameter values, including temperature (chosen from 50, 100, 200, 300, 400, 500), interpolation weight (selected from 0.1, 0.3, 0.5, 0.7), and the number of neighbors (picked from 20, 100, 1000). We conduct an exhaustive search for the best parameter combinations within each domain. We use FAISS (Johnson et al., 2019), an open-source library for fast nearest-neighbor retrieval in high dimensional spaces.

Our experiments are conducted on machines equipped with 8 Tesla V100 GPUs, each with 16GB of memory. In the context of kNN-ICL, the inference batch size is set to 3 for CodeGen and 1 for GPT-NeoX, respectively.

To assess kNN-ICL's performance, we establish the SPIS10 data split for each domain, where each domain comprises a maximum of 10 examples for intent or slot labels, simulating a few-shot setting. Additionally, we evaluate kNN-ICL on a larger data setting with a demo pool containing 2000 randomly sampled examples.

3.4 Prompt Design Results (RQ1)

Table 1 presents the exact match scores achieved through the ablation of different prompt components, including the presence of API documentation and the use of three exemplar selection strategies. Due to resource constraints, we randomly selected 20% of the validation data from the Navigation domain, resulting in 596 examples. Additionally, we include the results obtained using RINE and CodeT5 to provide a benchmark for ICL by comparing them with fine-tuning state-of-the-art results.

To summarize, first, the best performance for GPT-NeoX and CodeGen is achieved with prompts that do not include API documentation and utilize similarity-based selection strategies. This suggests that these two models struggle to leverage the documentation due to the input sequence length limitations, making it challenging to extract useful information. However, Codex benefits from API documentation when combined with the Sentence-BERT similarity strategy. This difference may be

LLM	Method	Navigation	Reminder	Alarm	Weather	Avg
CODET5	FINE-TUNE	10.02	6.61	15.72	6.60	9.74
RINE		42.28	36.87	32.09	32.53	35.94
GPT-NEOX	ICL	1.81	5.28	9.54	16.56	8.30
	kNN-LM	1.22	3.48	6.54	2.28	3.38
	kNN-ICL	**5.69**	**8.48**	**19.40**	**24.52**	**14.52**
CODEGEN	ICL	3.94	5.99	10.88	13.84	8.66
	kNN-LM	0.51	0.14	5.07	1.01	1.68
	kNN-ICL	**8.37**	**10.49**	**19.10**	**25.19**	**15.79**
CODEX	ICL	18.78	30.46	45.08	45.70	35.01
	kNN-ICL *	**35.74**	**41.36**	**57.56**	**53.35**	**47.00**

Table 2: Results of Exact Match on SIPS10 demo pool. We selected four domains using three LLMs: GPT-NEOX, CODEGEN, and CODEX on TOP. ICL refers to vanilla LM using randomly selected examples as demo; kNN-LM uses the external dataset as retrieval pool while no demo is used as prompt; and kNN-ICL is the proposed method using retrieved demo and external datastore. * indicates an estimated result from the black-box CODEX model using retrieved semantic similar demo examples.

attributed to Codex's stronger capacity to handle longer prompts and effectively utilize API documentation.

When comparing the SentenceBERT and paraphrase similarity-based selection strategies, SentenceBERT yields a better prompt when documentation is not included. The paraphrased model provides higher-quality exemplars in terms of outermost intent name similarity, but it falls short of capturing the nested semantic parse tree structure, which is crucial for predicting the correct API.

3.5 ICL vs. Supervised Methods (RQ2)

Table 2 reports the results for supervised models: RINE and CODET5 on the few-shot setting. We create SPIS10 data split for each domain in which each domain contains at most 10 examples for intent or slot label.

We compare ICL with fine-tuned SOTA models. Since CODEX is a black-box model, we use the top-k most similar examples as prompt to estimate its kNN-ICL performance, as shown in Table 2. We observed that the CODEX model consistently outperforms RINE by an average margin of 11.06 across four domains, with notable improvements of 4.5%, 25.5%, and 20.8% in the Reminder, Alarm, and Weather domains, respectively. However, RINE shows better performance on the most challenging domain Navigation. To summarize, the CODEX model demonstrates superior performance over SOTA models especially on flattener domains, while slightly under-performs on the domain with more challenging examples. Although GPT-NEOX and CODEGEN models lag behind CODET5 and RINE model, CODEX results show the potential of ICL from the high capacity model.

3.6 kNN-ICL Results (RQ3)

Few-shot setting In Table 2, we showcase the results for our novel approach, kNN-ICL, alongside the baseline method, kNN-LM. From this table, we can observe that kNN-ICL outperforms kNN-LM across all domains, demonstrating its effectiveness in leveraging prompts for TOP. Notably, kNN-ICL achieves improvements of 11.1% and 14.1% when compared to kNN-LM, using GPT-NEOX and CODEGEN, respectively.

We also compare kNN-ICL with its ICL counterpart. The results show that for GPT-NEOX and CODEGEN, kNN-ICL provides a boost in performance across an average of four domains, with gains of 6.22% and 7.13% in exact match scores. It's worth noting that our datastore size is limited, containing approximately 100 examples, which matches the size of the demo pool. Therefore, kNN-ICL effectively leverages all available example information to achieve these improved results.

Scale Up We conducted experiments to test the performance of kNN-ICL on a larger datastore, which includes 2000 examples. The goal was to determine if kNN-ICL could benefit more from a larger datastore compared to kNN-LM. When comparing kNN-ICL with kNN-LM, we found that the performance gap between the two models becomes larger in the presence of the larger datastore. Notably, kNN-ICL outperforms kNN-LM by 20.3% and 19.0% using CODEGEN and GPT-NEOX as backbone models respectively. Further-

LLM	Method	Navigation	Reminder	Alarm	Weather	average
CodeGen	ICL-Retrieve	37.00	23.40	48.10	65.80	43.58
	kNN-LM	21.30	3.10	26.99	46.60	24.50
	kNN-ICL	**38.20**	**24.20**	**50.40**	**66.60**	**44.85**
GPT-NeoX	ICL-Retrieve	**33.40**	**24.10**	46.70	65.70	42.48
	kNN-LM	14.30	6.70	33.50	40.00	23.63
	kNN-ICL	33.30	24.00	**46.90**	**66.30**	**42.63**

Table 3: Results for Exact Match on a randomly sampled pool of 2000 demonstrations, illustrating the scalability of kNN-ICL.

Depth	GPT-NeoX		CodeGen	
	ICL	kNN-ICL	ICL	kNN-ICL
1	2.53	**6.42**	5.90	**10.16**
2	1.14	**5.72**	1.72	**6.45**
3	0.00	**1.93**	0.16	**4.98**

Table 4: Analysis of accuracy with respect to nesting depth (1/2/3) for the TOPv2 dataset in the Navigation domain using different models.

more, we compared kNN-ICL to ICL-Retrieve, a method that retrieves the most semantically similar example from the demo. In this comparison, kNN-ICL outperforms CodeGen by 1.2%, 0.8%, 2.3%, and 0.8% on the Navigation, Reminder, Alarm, and Weather domains, respectively. This improvement highlights that even when a demo selection strategy is fixed, kNN-ICL can still bring about performance enhancements.

For GPT-NeoX, the kNN-ICL shows similar performance on the Navigation and Reminder domains, with a slight improvement observed in the Alarm and Weather domain.

3.7 Additional Analysis: Depth vs. Accuracy

Table 4 illustrates the model's performance concerning the semantic depth in the Topv2 dataset. Our analysis focuses on the Navigation domain, which contains the most complex average semantic parse tree depth. Notably, ICL on GPT-NeoX and CodeGen exhibits a consistent performance drop with increasing depth. On the other hand, kNN-ICL significantly enhances the performance of examples compared to its ICL counterpart.

3.8 Case Study

In order to assess the performance of state-of-the-art models, specifically ICL, kNN-LM, and kNN-ICL, we present a case study in Figure 3. This case study provides examples of utterances, corresponding ground-truth APIs, and model predictions.

Utterance: How long will it take to drive to New Orleans if I leave my house at 3 am
GroundTruth: GET_ESTIMATED_DURATION (METHOD_TRAVEL = " drive " , DESTINATION = " New Orleans " , SOURCE = GET_LOCATION_HOME (CONTACT = " my ") , DATE_TIME_DEPARTURE = " at 3 am ")

CodeT5: GET_ESTIMATED_DURATION (METHOD_TRAVEL = " drive ", DESTINATION = GET_LOCATION (POINT_ON_MAP = " New Orleans "), DATE_TIME_DEPARTURE = " at 3 am ")

ICL: "GET_ESTIMATED_DURATION (METHOD_TRAVEL = " drive " , SOURCE = GET_LOCATION_HOME (CONTACT_RELATED = " my ") , DESTINATION = GET_LOCATION_WORK (CONTACT = " my ") , DATE_TIME_ARRIVAL = " at 3 am ")

kNN-LM: GET_INFO_TRAFFIC (LOCATION = GET_LOCATION ( ))

kNN-ICL: GET_ESTIMATED_DURATION (METHOD_TRAVEL = " drive ", DESTINATION = " New Orleans ", SOURCE = GET_LOCATION_HOME (CONTACT = " my "), DATE_TIME_DEPARTURE = " at 3 am ")

Figure 3: A case study from Topv2. The wrong intent and slot name from prediction are marked in red, and the hallucinated nested API call is marked in blue. The ICL, kNN-ICL, kNN-LM uses GPT-NeoX model.

The showcased example includes a nested semantic parse tree structure with a depth of 3.

First, CodeT5 exhibits overall good quality, except for hallucinating the slot name `POINT_ON_MAP` on the deepest tree branch. However, ICL had a hard time generating API calls for deeper-level APIs, such as hallucinating `GET_LOCATION_WORK ( CONTACT = " my " )` for slot `DESTINATION`. kNN-LM suffers from the wrong outermost intent name prediction as well as the slot name prediction because no prompt is provided, causing severe deviation from the expected output. In contrast, kNN-ICL performs best on this complex example with the aid of both prompting, producing an accurate utterance span as a slot value, and the k nearest neighbor guidance, choosing the proper nested API structure.

4 Related Work

Semantic Parsing Semantic parsing plays a vital role in developing commercial personal assistants for understanding spoken language. Supervised models, including sequence-to-sequence models (Mesnil et al., 2013; Liu and Lane, 2016) and pretrained language models (Devlin et al., 2018; Chen et al., 2019), have demonstrated competitive performance. In structured semantic parsing, recent approaches break down semantic tree construction into multiple steps. For instance, RINE (Mansimov and Zhang, 2022) recursively inserts predicted labels at predicted positions to construct the semantic parse tree. Additionally, some methods, like (Zhao et al., 2022), transform semantic parsing into multi-turn abstractive question answering. In few-shot scenarios, techniques such as meta-learning (Chen et al., 2020) and label semantics (Paolini et al., 2021; Desai et al., 2021) enhance neural semantic parsers. Furthermore, (Shu et al., 2022) simplifies semantic parsing to program synthesis and integrates the predicted program into an executable environment.

In-Context Learning With the emergence of Large Language Models (Nijkamp et al., 2022b; Black et al., 2022; Brown et al., 2020; Scao et al., 2022; Chen et al., 2021; Chowdhery et al., 2022), model parameters have scaled from 16 billion to over 500 billion. The substantial resource requirements for hosting LLMs have made it challenging to fine-tune them for downstream tasks. As a result, the paradigm for utilizing language models has shifted from traditional pre-training and fine-tuning to more efficient approaches, such as parameter-efficient fine-tuning(Lester et al., 2021; Li and Liang, 2021; He et al., 2021) and in-context learning (Dong et al., 2022; Liu et al., 2022). ICL has demonstrated its reasoning capabilities in both plain text (Wei et al., 2022; Zhang et al., 2022) and structured text formats (Chen, 2022; Chen et al., 2022). Researchers have also analyzed the factors contributing to the effectiveness of ICL (Dai et al., 2022; Min et al., 2022; von Oswald et al., 2022) in general tasks. However, existing literature primarily focuses on the overall performance of ICL in generic tasks. Our work, on the other hand, falls within a specific domain, where we investigate the performance of ICL in the context of task-oriented semantic parsing problems.

Retrieval-Augmented Large Language Model Recent advancements in Retrieval-Augmented Language Models (RALMs) have garnered significant attention within the NLP community. LLMs face challenges in terms of scaling (Khandelwal et al., 2019) and acquiring long-tail knowledge (Mallen et al., 2023; Izacard et al., 2022). RALMs (Guu et al., 2020; Liu et al., 2023) present a promising solution by seamlessly integrating non-parametric data stores with their parametric counterparts. To enhance the adaptability of RALMs for downstream tasks, researchers have devised strategies such as zero or few-shot prompting (Shi et al., 2022; Xu et al., 2023) and fine-tuning (Shi et al., 2023). To the best of our knowledge, our work stands as the pioneering effort to combine the k nearest neighbor retrieval language model with in-context learning on generative tasks.

Code Generation Automated code generation dates back as far as a few decades ago (Backus et al., 1957; Manna and Waldinger, 1971). Recently, the code generation has been dominated by the LLMs, including the closed-source models AlphaCode (Li et al., 2022) and ChatGPT (OpenAI, 2022), as well the open-source models such as CodeT5 (Wang et al., 2021), CodeT5+ (Wang et al., 2023), CodeGen (Nijkamp et al., 2022a), InCoder (Fried et al., 2022), StarCoder (Li et al., 2023), and Code Llama (Rozière et al., 2023). Inspired by this line of research, our work extends the scope of TOP to encompass code generation.

5 Conclusion

In this paper, we delved into the intricacies of prompt design and introduced kNN-ICL to include all demo examples on the Task-Oriented semantic parsing task, addressing input length challenges during decoding and lessening prompt engineering efforts. Our findings highlight the substantial impact of prompt design on TOP, with models like CODEX deriving more benefits from structured documentation than less capable models. The similarity-based exemplar retrieval strategy enhanced model performance significantly. Furthermore, kNN-ICL's integration with a large language model and k Nearest Neighbor search ensures comprehensive use of training data, offering more consistent results across tasks. This study underscores the critical role of prompt design, pointing to promising avenues for future research.

6 Limitations

Generalization of Prompt Design This study highlights the varying impact of prompt design on different models based on their capacities. Instead of advocating a universal prompt design strategy, the optimal choice should be model-agnostic. A potential avenue for future research is enabling models to autonomously select their preferred prompt designs (Zamfirescu-Pereira et al., 2023; Zhou et al., 2022). Depending on the LLM's chosen strategy, kNN-ICL can serve as a plugin to further enhance model performance by harnessing all available examples.

Generalization to High-Capacity Models This paper offers an initial exploration of kNN-ICL with LLMs. While CODEX is not publicly accessible and the resource limitation, we have used GPT-NEOX and CODEGEN as representative models for kNN-ICL experiments. Nevertheless, given the substantial performance disparity between GPT-NEOX, CODEGEN, and CODEX, future research should delve into assessing the influence of kNN-ICL on more robust models.

References

John W Backus, Robert J Beeber, Sheldon Best, Richard Goldberg, Lois M Haibt, Harlan L Herrick, Robert A Nelson, David Sayre, Peter B Sheridan, Harold Stern, et al. 1957. The fortran automatic coding system. In *Papers presented at the February 26-28, 1957, western joint computer conference: Techniques for reliability*, pages 188–198.

Sid Black, Stella Biderman, Eric Hallahan, Quentin Anthony, Leo Gao, Laurence Golding, Horace He, Connor Leahy, Kyle McDonell, Jason Phang, et al. 2022. Gpt-neox-20b: An open-source autoregressive language model. *arXiv preprint arXiv:2204.06745*.

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George Bm Van Den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, Diego De Las Casas, Aurelia Guy, Jacob Menick, Roman Ring, Tom Hennigan, Saffron Huang, Loren Maggiore, Chris Jones, Albin Cassirer, Andy Brock, Michela Paganini, Geoffrey Irving, Oriol Vinyals, Simon Osindero, Karen Simonyan, Jack Rae, Erich Elsen, and Laurent Sifre. 2022. Improving language models by retrieving from trillions of tokens. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 2206–2240. PMLR.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gašić. 2018. MultiWOZ - a large-scale multi-domain Wizard-of-Oz dataset for task-oriented dialogue modelling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5016–5026, Brussels, Belgium. Association for Computational Linguistics.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Qian Chen, Zhu Zhuo, and Wen Wang. 2019. Bert for joint intent classification and slot filling. *arXiv preprint arXiv:1902.10909*.

Wenhu Chen. 2022. Large language models are few (1)-shot table reasoners. *arXiv preprint arXiv:2210.06710*.

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W. Cohen. 2022. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *arXiv preprint arXiv:2211.12588*.

Xilun Chen, Asish Ghoshal, Yashar Mehdad, Luke Zettlemoyer, and Sonal Gupta. 2020. Low-resource domain adaptation for compositional task-oriented semantic parsing. *arXiv preprint arXiv:2010.03546*.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2022. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*.

Damai Dai, Yutao Sun, Li Dong, Yaru Hao, Zhifang Sui, and Furu Wei. 2022. Why can gpt learn in-context? language models secretly perform gradient descent as meta optimizers. *arXiv preprint arXiv:2212.10559*.

Shrey Desai, Akshat Shrivastava, Alexander Zotov, and Ahmed Aly. 2021. Low-resource task-oriented semantic parsing via intrinsic modeling. *arXiv preprint arXiv:2104.07224*.

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

Li Dong and Mirella Lapata. 2018. Coarse-to-fine decoding for neural semantic parsing. *arXiv preprint arXiv:1805.04793*.

Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Zhiyong Wu, Baobao Chang, Xu Sun, Jingjing Xu, and Zhifang Sui. 2022. A survey for in-context learning. *arXiv preprint arXiv:2301.00234*.

Andrew Drozdov, Nathanael Schärli, Ekin Akyürek, Nathan Scales, Xinying Song, Xinyun Chen, Olivier Bousquet, and Denny Zhou. 2022. Compositional semantic parsing with large language models. *arXiv preprint arXiv:2209.15003*.

Daniel Fried, Armen Aghajanyan, Jessy Lin, Sida Wang, Eric Wallace, Freda Shi, Ruiqi Zhong, Wen-tau Yih, Luke Zettlemoyer, and Mike Lewis. 2022. Incoder: A generative model for code infilling and synthesis. *arXiv preprint arXiv:2204.05999*.

Sonal Gupta, Rushin Shah, Mrinal Mohit, Anuj Kumar, and Mike Lewis. 2018. Semantic parsing for task oriented dialog using hierarchical representations. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2787–2792, Brussels, Belgium. Association for Computational Linguistics.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. 2020. Retrieval augmented language model pre-training. In *International conference on machine learning*, pages 3929–3938. PMLR.

Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. 2021. Towards a unified view of parameter-efficient transfer learning. *arXiv preprint arXiv:2110.04366*.

Gautier Izacard, Patrick Lewis, Maria Lomeli, Lucas Hosseini, Fabio Petroni, Timo Schick, Jane Dwivedi-Yu, Armand Joulin, Sebastian Riedel, and Edouard Grave. 2022. Few-shot learning with retrieval augmented language models. *arXiv preprint arXiv:2208.03299*.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*, 7(3):535–547.

Urvashi Khandelwal, Angela Fan, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. 2020. Nearest neighbor machine translation. *arXiv preprint arXiv:2010.00710*.

Urvashi Khandelwal, Angela Fan, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. 2021. Nearest neighbor machine translation. In *International Conference on Learning Representations (ICLR)*.

Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. 2019. Generalization through memorization: Nearest neighbor language models. *arXiv preprint arXiv:1911.00172*.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*.

Raymond Li, Loubna Ben Allal, Yangtian Zi, Niklas Muennighoff, Denis Kocetkov, Chenghao Mou, Marc Marone, Christopher Akiki, Jia Li, Jenny Chim, Qian Liu, Evgenii Zheltonozhskii, Terry Yue Zhuo, Thomas Wang, Olivier Dehaene, Mishig Davaadorj, Joel Lamy-Poirier, João Monteiro, Oleh Shliazhko, Nicolas Gontier, Nicholas Meade, Armel Zebaze, Ming-Ho Yee, Logesh Kumar Umapathi, Jian Zhu, Benjamin Lipkin, Muhtasham Oblokulov, Zhiruo Wang, Rudra Murthy, Jason Stillerman, Siva Sankalp Patel, Dmitry Abulkhanov, Marco Zocca, Manan Dey, Zhihan Zhang, Nour Fahmy, Urvashi Bhattacharyya, Wenhao Yu, Swayam Singh, Sasha Luccioni, Paulo Villegas, Maxim Kunakov, Fedor Zhdanov, Manuel Romero, Tony Lee, Nadav Timor, Jennifer Ding, Claire Schlesinger, Hailey Schoelkopf, Jan Ebert, Tri Dao, Mayank Mishra, Alex Gu, Jennifer Robinson, Carolyn Jane Anderson, Brendan Dolan-Gavitt, Danish Contractor, Siva Reddy, Daniel Fried, Dzmitry Bahdanau, Yacine Jernite, Carlos Muñoz Ferrandis, Sean Hughes, Thomas Wolf, Arjun Guha, Leandro von Werra, and Harm de Vries. 2023. Starcoder: may the source be with you!

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*.

Yujia Li, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Rémi Leblond, Tom Eccles, James Keeling, Felix Gimeno, Agustin Dal Lago, et al. 2022. Competition-level code generation with alphacode. *Science*, 378(6624):1092–1097.

Bing Liu and Ian Lane. 2016. Attention-based recurrent neural network models for joint intent detection and slot filling. *arXiv preprint arXiv:1609.01454*.

Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. 2022. 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*, Dublin, Ireland and Online.

Ye Liu, Semih Yavuz, Rui Meng, Meghana Moorthy, Shafiq Joty, Caiming Xiong, and Yingbo Zhou. 2023. Exploring the integration strategies of retriever and large language models. *arXiv preprint arXiv:2308.12574*.

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

Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9802–9822, Toronto, Canada. Association for Computational Linguistics.

Zohar Manna and Richard J Waldinger. 1971. Toward automatic program synthesis. *Communications of the ACM*, 14(3):151–165.

Elman Mansimov and Yi Zhang. 2022. Semantic parsing in task-oriented dialog with recursive insertion-based encoder. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 11067–11075.

Grégoire Mesnil, Xiaodong He, Li Deng, and Yoshua Bengio. 2013. Investigation of recurrent-neural-network architectures and learning methods for spoken language understanding. In *Interspeech*, pages 3771–3775.

Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2022. Rethinking the role of demonstrations: What makes in-context learning work? *arXiv preprint arXiv:2202.12837*.

Erik Nijkamp, Bo Pang, Hiroaki Hayashi, Lifu Tu, Huan Wang, Yingbo Zhou, Silvio Savarese, and Caiming Xiong. 2022a. Codegen: An open large language model for code with multi-turn program synthesis. *arXiv preprint arXiv:2203.13474*.

Erik Nijkamp, Bo Pang, Hiroaki Hayashi, Lifu Tu, Huan Wang, Yingbo Zhou, Silvio Savarese, and Caiming Xiong. 2022b. A conversational paradigm for program synthesis. *arXiv preprint arXiv:2203.13474*.

OpenAI. 2022. ChatGPT. `https://openai.com/blog/chatgpt/`.

Giovanni Paolini, Ben Athiwaratkun, Jason Krone, Jie Ma, Alessandro Achille, Rishita Anubhai, Cicero Nogueira dos Santos, Bing Xiang, and Stefano Soatto. 2021. Structured prediction as translation between augmented natural languages. *arXiv preprint arXiv:2101.05779*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, Peter J Liu, et al. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(140):1–67.

Baptiste Rozière, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Tal Remez, Jérémy Rapin, et al. 2023. Code llama: Open foundation models for code. *arXiv preprint arXiv:2308.12950*.

Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, et al. 2022. Bloom: A 176b-parameter open-access multilingual language model. *arXiv preprint arXiv:2211.05100*.

Weijia Shi, Julian Michael, Suchin Gururangan, and Luke Zettlemoyer. 2022. Nearest neighbor zero-shot inference. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 3254–3265, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Weijia Shi, Sewon Min, Michihiro Yasunaga, Minjoon Seo, Rich James, Mike Lewis, Luke Zettlemoyer, and Wen-tau Yih. 2023. Replug: Retrieval-augmented black-box language models. *arXiv preprint arXiv:2301.12652*.

Eui Chul Shin, Miltiadis Allamanis, Marc Brockschmidt, and Alex Polozov. 2019. Program synthesis and semantic parsing with learned code idioms. *Advances in Neural Information Processing Systems*, 32.

Richard Shin, Christopher Lin, Sam Thomson, Charles Chen, Subhro Roy, Emmanouil Antonios Platanios, Adam Pauls, Dan Klein, Jason Eisner, and Benjamin Van Durme. 2021. Constrained language models yield few-shot semantic parsers. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7699–7715, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Raphael Shu, Elman Mansimov, Tamer Alkhouli, Nikolaos Pappas, Salvatore Romeo, Arshit Gupta, Saab Mansour, Yi Zhang, and Dan Roth. 2022. Dialog2api: Task-oriented dialogue with api description and example programs. *arXiv preprint arXiv:2212.09946*.

Johannes von Oswald, Eyvind Niklasson, Ettore Randazzo, João Sacramento, Alexander Mordvintsev, Andrey Zhmoginov, and Max Vladymyrov. 2022. Transformers learn in-context by gradient descent. *arXiv preprint arXiv:2212.07677*.

Yue Wang, Hung Le, Akhilesh Deepak Gotmare, Nghi DQ Bui, Junnan Li, and Steven CH Hoi. 2023. Codet5+: Open code large language models for code understanding and generation. *arXiv preprint arXiv:2305.07922*.

Yue Wang, Weishi Wang, Shafiq Joty, and Steven CH Hoi. 2021. Codet5: Identifier-aware unified pre-trained encoder-decoder models for code understanding and generation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 8696–8708.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed Chi, Quoc Le, and Denny Zhou. 2022. Chain of thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*.

Qingyang Wu, James Gung, Raphael Shu, and Yi Zhang. 2023. DiactTOD: Learning generalizable latent dialogue acts for controllable task-oriented dialogue systems. In *Proceedings of the 24th Meeting of the Special Interest Group on Discourse and Dialogue*, pages 255–267, Prague, Czechia. Association for Computational Linguistics.

Benfeng Xu, Quan Wang, Zhendong Mao, Yajuan Lyu, Qiaoqiao She, and Yongdong Zhang. 2023. k nn prompting: Beyond-context learning with calibration-free nearest neighbor inference. *arXiv preprint arXiv:2303.13824*.

JD Zamfirescu-Pereira, Richmond Y Wong, Bjoern Hartmann, and Qian Yang. 2023. Why johnny can't prompt: how non-ai experts try (and fail) to design llm prompts. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, pages 1–21.

John M Zelle and Raymond J Mooney. 1996. Learning to parse database queries using inductive logic programming. In *Proceedings of the national conference on artificial intelligence*, pages 1050–1055.

Zhuosheng Zhang, Aston Zhang, Mu Li, and Alex Smola. 2022. Automatic chain of thought prompting in large language models. *arXiv preprint arXiv:2210.03493*.

Wenting Zhao, Konstantine Arkoudas, Weiqi Sun, and Claire Cardie. 2022. Compositional task-oriented parsing as abstractive question answering. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4418–4427, Seattle, United States. Association for Computational Linguistics.

Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. 2022. Large language models are human-level prompt engineers.