# PEARL: Preference Extraction with Exemplar Augmentation and Retrieval with LLM Agents

**Vijit Malik**
Amazon AI
vijitvm@amazon.com

**Akshay Jagatap**
Amazon AI
ajjagata@amazon.com

**Vinayak Puranik**
Amazon AI
puranikv@amazon.com

**Anirban Majumder**
Amazon AI
majumda@amazon.com

## Abstract

Identifying preferences of customers in their shopping journey is a pivotal aspect in providing product recommendations. The task becomes increasingly challenging when there is a multi-turn conversation between the user and a shopping assistant chatbot. In this paper, we address a novel and complex problem of identifying customer preferences in the form of key-value filters on an e-commerce website in a multi-turn conversational setting. Existing systems specialize in extracting customer preferences from standalone customer queries which makes them unsuitable to multi-turn setup. We propose PEARL(**P**reference **E**xtraction with ICL **A**ugmentation and **R**etrieval with **L**LM Agents) that leverages collaborative LLM agents, generates in-context learning exemplars and dynamically retrieves relevant exemplars during inference time to extract customer preferences as a combination of key-value filters. Our experiments on proprietary and public datasets show that PEARL not only improves performance on exact match by $\approx 10\%$ compared to competitive LLM-based baselines but additionally improves inference latency by $\approx 110\%$.

## 1 Introduction

Large selection, attractive pricing and convenience have made online shopping very popular in recent years. However, traditional e-commerce services still offer search-based interface which is inadequate for broad, ambiguous and *upper funnel* user queries. Absence of a conversational interface often leaves customers feeling the need of human-like assistance to explain their requirements and navigate towards the right set of products (Abbey, 2023; Cheng et al., 2023; Gumusel et al., 2023; Liu et al., 2023b; Guo et al., 2023a; Roller et al., 2021; Li et al., 2021; Liu et al., 2023a). With the emergence of generative artificial intelligence (GenAI) in recent years, much research efforts have been devoted on building multi-turn conversational chatbots that can serve as virtual shopping assistant, akin to the trained sales agents commonly found in physical stores.

In contrast to traditional search-based e-commerce services that operate on single-shot queries, multi-turn conversations require the identification of an evolving set of user preferences embedded within the dialogue. The primary objective of the chatbot is to extract customer preferences from the conversation and map them to preference filters (e.g. `Brand`, `CPU`, `Price` etc.). Extracting preference filters from multi-turn conversation is challenging for several reasons. Firstly, the filter mentions in user utterance can be non-standardized (e.g., *ram* and *memory* are both surface forms of the filter key `RAM`), implicit (e.g., *"HP laptop 16GB"* refers to the filter key `RAM` with a value of `16GB`) or having complex preferences (e.g., *"not windows os"*, *"16gb or more"*). User requirements can be ambiguous with no clear mention of the requirement, e.g., *"heavy-duty laptop"* may imply a high-performance laptop or a rugged one. Another layer of complexity arises from the fact that customers may modify their preferences over the course of multiple turns. For instance, a customer utterance such as *"what about MacBooks with M2"* would change their previously stated preference for the `CPU` from `M1` to `M2`. Refer to Table 1 for more such examples.

To address these challenges, we propose a novel architecture for extracting refinements from multi-turn conversation history that leverages multiple large language models (LLMs) as collaborative agents (Guo et al., 2023b; Gao et al., 2023a; Clarke et al., 2023). Our system (PEARL- **P**reference **E**xtraction with ICL **A**ugmentation and **R**etrieval with **L**LM Agents) dynamically retrieves the most relevant in-context exemplars from an index during inference time. These ICL exemplars (Dong et al., 2022; Min et al., 2023; Kim and et al., 2023) are a combination of human-curated and synthet-

| Conversational E-commerce | Why navigating preferences is a difficult task? |
|---|---|
| Current set of preferences: {'Price': '$1000 and above'}<br>Next utterance: 'not windows os' | Complex preference combinations |
| Current set of preferences: {'Brand': 'Dell', 'CPU Type': 'Intel Core i7'<br>Next utterance: 'now can you show apple laptops with m2?' | Preference editing |
| Current set of preferences: {'Brand': 'Dell'}<br>Next utterance: 'should be lightweight and good battery' | Ambiguity |
| Current set of preferences: {'Brand': 'HP', 'HDD-Size': '1 TB & above'}<br>Next Utterance: 'HP probook 445 G7 , how much storage' | Complexity in intent |
| Current set of preferences: {'CPU Type': 'Intel Core i5\|Intel Core i7\|Intel Core i9',..<br>..'Price': '$700 to $800', 'CPU Processor Speed': '1.80 to 1.99 GHz'}<br>Next Utterance: 'windows laptop' | Number of preferences per conversation |

Table 1: Examples showing the complexity of the task of preferences navigation in a multi-turn chat scenario. The examples are shown on an utterance level showing the innate natural language understanding required in handling this task. Note that we show multiple refinement picker values for same refinement key separated by '|'.

ically generated exemplars by PEARL through an offline process. We compare the performance of PEARL against existing production systems adapted to similar tasks, demonstrating its superior accuracy and inference latency. Through empirical evaluation and analysis, we highlight the efficacy of our approach and its potential to improve the identification of customer preferences in conversational e-commerce settings. Our comprehensive set of experiments shows that PEARL not only improves the performance of extracting refinements from conversational logs by $10\%$ compared to a production system but also reduces the latency by $110\%$.

## 2 Related Works

**Large Language Models (LLMs) in Conversational AI** The advent of Large Language Models (LLMs) has revolutionized conversational chatbots by enabling them to comprehend and generate human-like text (Radford et al., 2019; Roller et al., 2021). The integration of LLMs in personalized product recommendations has emerged as a promising avenue for enhancing the e-commerce experience (Gao et al., 2023b; Dong et al., 2022; Zhao et al., 2023; Gao et al., 2023a) with collaborative LLM agents (Cambon et al., 2023; Guo et al., 2023b; Haslberger et al., 2023) having the potential to enhance productivity further by addressing complex challenges. Recently, the problem of demonstration selection (Xu and Zhang, 2024; S. et al., 2024; Li et al., 2023) for **in-context learning (ICL)** has received a significant attention in the literature with several works incorporating an ICL retriever (Li et al., 2023; Wang et al., 2024a) that augments the LLM by inserting relevant ICLs in the prompt to improve task level performance.

**NL2API** The task of translating natural language inputs into API calls typically hinges on rule-based techniques (Woods, 1973) and Deep Neural Networks (DNNs) (Sun et al., 2016; Yih et al., 2015) and found applications in databases (Kothyari et al., 2023), knowledge graphs (Campêlo et al., 2023) and web tables (Sun et al., 2016). Only recently, there has been successful application of using LLMs to invoke external tools or APIs (Qin et al., 2023; Patil et al., 2023). The key challenge in this space is the lack of domain-specific datasets, an aspect that we specifically address in PEARL.

## 3 Problem Formulation

The conversation between user and chatbot is represented as a sequence of textual utterances $\mathcal{U}_t = \{u_0, u_1, \cdots u_{t-1}\}$ where $\texttt{Speaker}(u_j) \in \{\texttt{USER}, \texttt{BOT}\}$ for all $j \in [t]$, $t$ being the current timestamp. We assume that the conversation has been mapped to a product category $\mathcal{C}$ for which we have access to the set of valid filter preferences $F_\mathcal{C} = \{(K_i, V_i)\}$ containing filter schema keys $K_i \in K_\mathcal{C}$ where $K_\mathcal{C}$ is the universe of filter keys for category $\mathcal{C}$ (e.g. for "laptop" category, valid filter keys are RAM, CPU etc) and set $V_i$ of corresponding picker values (e.g. {4GB, 8GB, 16GB, 32GB} for the filter key RAM). Given the conversation history $\mathcal{U}_t$, our goal is to map the user requirement into a set of filter key-value pairs. Mathematically, we want to learn a function $f$ such that,

$$f(\mathcal{U}_t, F_\mathcal{C}) = \{(k_p, v_p)\} \quad (1)$$

where $\{k_p\} \subseteq K_\mathcal{C}$ is a subset of filter keys mentioned in $\mathcal{U}_t$ with corresponding values $v_p \subseteq V_p$. Refer to Figure 1 for an example of multi-turn conversation and user preference extracted by PEARL.

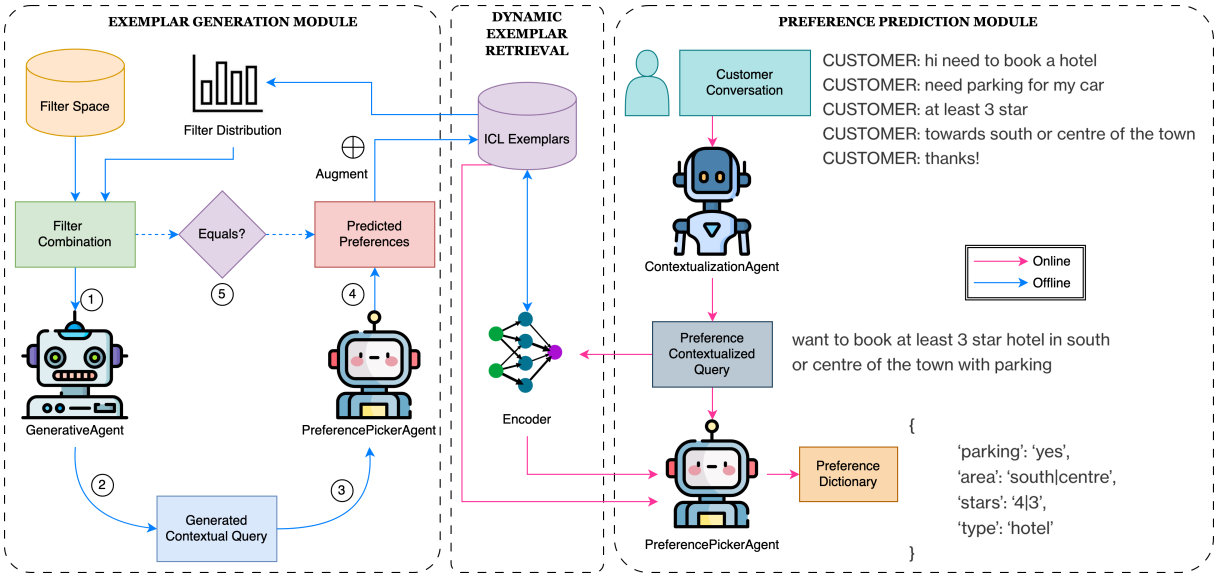


Figure 1: Schematic flow diagram of PEARLconsisting of three modules: ICL Generation, Dynamic ICL Retrieval and Preference Prediction module. Steps which are offline and online are marked with differently coloured arrows in the figure (best viewed in color).

## 4 Dataset Details

We analyze conversation logs collected from a chatbot application where human agents acted as shopping assistant to customers for their purchase in a specific product category (laptop). We used chat logs with personally identifiable information (PII) redacted for privacy and compliance. We conduct annotations over complete chats and record the set of preferences of the customer at the end of the chat session. Due to the complexity and effort in labelling these chat transcripts, we use a set of $\approx 400$ sessions as our test and 100 more labelled chat sessions as our support set for ICL exemplars. During the annotation process, we consider preferences of user that can be mapped unambiguously to a combination of filter keys and values (Li et al., 2021; Zhang et al., 2020; Wang et al., 2020b; Liu et al., 2019). Refer to Appendix A.1 for more details on the `Internal` dataset.

To demonstrate generalization of our techniques, we report performance of PEARL on the MultiWoZ (Zang et al., 2020) dataset. We specifically choose the 'Hotel' domain of MultiWoZ (`MultiWoZ-H`) having the largest number of preference filters, thereby, increasing the complexity of the task. We curate a dataset of $\approx 350$ chat sessions as test set and a set of 100 chat sessions as ICL exemplars. Refer to Appendix A.1 for data preparation steps and analyses.

## 5 Methodology

We propose PEARL, a collaborative multi-agent framework that leverages LLM-based agents with diverse functionalities to effectively navigate customer preferences in a multi-turn conversational system. Note that in our experiments, chat conversations have been PII redacted. Figure 1 shows our framework consisting of the primary component for preference prediction accompanied by two auxiliary components: for dynamic retrieval of ICL exemplars and ICL generation.

**Preference Prediction Module** The preference prediction module is the core component that takes the current utterance from customer and the conversation history as input and generates a dictionary of filter key-values summarizing customer preferences (c.f. Figure 1). It leverages two LLM agents: `ContextualizationAgent` and `PreferencePickerAgent`. In the first step, `ContextualizationAgent` summarizes the entire conversation into a contextualized utterance that captures all preferences of the customer in a singe sentence. subsequently, `PreferencePickerAgent` generates a dictionary of filter key-values from the contextualized utterance, filter space $Fc$ and ICL exemplars. This dual-agent architecture mimics how a salesperson listens to the customer, summarizes their requirement and then executes a search based on those preferences.

**Dynamic Exemplar Retrieval** Recent stud-

ies (Rajapakse, 2023; Jiang et al., 2023; Lewis et al., 2020; Izacard and Grave, 2021; Hofstätter et al., 2023) have shown that LLM performance can be improved by inserting relevant ICL exemplars in the prompt. Guided by this observation, the dynamic ICL retrieval module in PEARL stores an index of ICL examplars of input, output pairs where input is the pre-curated response from ContextualizationAgent and output is the preference key-value filters. At inference time, PreferencePickerAgent obtains the top-k closest exemplar matches from the index based on the current contextualized query and inserts them in the LLM prompt.

**Exemplar Generation Module** Manually curating relevant and diverse pool of ICL exemplars is a time-consuming and error-prone process. To reduce dependency on human effort, PEARL uses the exemplar generation module to generate synthetic but plausible ICL exemplars. To promote diversity, we leverage the large filter space present in $F_{\mathcal{C}}$ to sample a preference combination $\mathcal{P}_{\texttt{sampled}}$. To avoid sampling invalid combinations [1] we fit a distribution over filter pairs based on filter combinations observed in catalog of existing products. PEARL uses a generative model GenerativeAgent to generate potential exemplar $\mathcal{I}_{\texttt{syn}}$ from $\mathcal{P}_{\texttt{sampled}}$ which is verified by PreferencePickerAgent to generate a set of filters $\mathcal{P}_{\texttt{generated}}$. Only when $\mathcal{P}_{\texttt{generated}} = \mathcal{P}_{\texttt{sampled}}$, the exemplar $\mathcal{I}_{\texttt{syn}}$ is added to the index.

## 6 Experiments and Results

To the best of our knowledge, there is no known published work on user preference extraction from multi-turn conversation logs. Therefore, we explore strong baselines as mentioned below to evaluate PEARL. Experiments are performed on LLMs hosted on AWS Bedrock and unless specified otherwise all results are reported for Claude-instance-v1 (c.f. Section 6.2 for results on different choices of LLMs). Refer to Appendix A.3 for prompt details.

**NL2API** For this baseline, an LLM takes the conversation history and the filter space as input and identifies the preferences as a combination of filters in a single-step prompting approach. Three setups are explored: 1) Basic (no ICL), 2) 10-shot ICL with static exemplars, and 3) 10-shot ICL with step-by-step chain-of-thoughts reasoning. Additionally, NL2API-2s is proposed as a two-step stronger baseline, where the first step determines the filter keys in the conversation, and the second step determines the filter picker values for each key.

**OperatorLLM** This approach navigates customer preferences through the conversation by iteratively editing the preference set based on each user utterance, using operations like adding, removing, or updating filter key-value pairs. Results are reported for three setups: 1) Basic (no ICL), 2) 10-shot ICL and 2) 10-shot ICL with chain-of-thoughts reasoning.

**PEARL** In our proposed approach, ContextualizationAgent uses 10 static ICL exemplars constructed from real conversation logs and GenerativeAgent augments the initial index of 100 ICLs with $1.2k$ synthetic exemplars. For dynamic exemplar retrieval, we use neural representations (refer to Table 4) to obtain contextual query embeddings. During inference, the top 10 ($k = 10$) exemplars are retrieved from the augmented exemplar set for the PreferencePickerAgent agent based on cosine similarity.

### 6.1 Results

**Metrics** Our primary metric is exact match (EM) which is defined as $\frac{1}{N}\sum_{i=1}^{n}\mathbb{I}(\mathcal{P}_i = \mathcal{G}_i)$, where $\mathcal{P}_i$ is the predicted set of filters, $\mathcal{G}_i$ is the ground truth for chat session $i$ and $\mathbb{I}(\cdot)$ is the indicator function. Note that while we also report microF1, exact match is a stricter metric: for example, predicting 8GB in place of 8GB|16GB gives us exact match of 0 but F1 of 0.67. We define another metric "Filter Edit Distance" (FED) (Zhu et al., 2023; Kaji, 2023), which measures the edit distance between the predicted set of filters to the ground-truth value. An interpretation of this metric is the number of operations (delete/add) that need to be applied on the predicted filters to obtain the true values. We also report inference latency of each approach which is averaged over all the chat sessions in our evaluation dataset.

Due to confidentiality reason, we report only relative improvements of PEARL over baselines on Internal test set; however absolute numbers are reported on MultiWoZ-H. Note that during evaluation, we execute each approach 6 times on the test set and report metrics with mean and standard error. Also, for MultiWoZ-H, filter values for some keys can be unbounded, due to which we use fuzzy

[1] An invalid preference pair could be Apple branded laptop with Nvidia GeForce GTX graphics card.

| Setup | Paradigm | Internal | | | | MultiWOZ-H | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | EM% ↑ | F1 ↑ | FED ↓ | Latency (s)↓ | EM% ↑ | F1 ↑ | FED ↓ | Latency (s)↓ |
| NL2API | Basic | – | – | – | – | $9.20_{0.24}$ | $0.7405_{0.0049}$ | 2.60 | 1.14 |
| NL2API | ICL@10 | $+8.34_{0.14}$ | $+0.0505_{0.0028}$ | −0.34 | +0.38 | $18.10_{0.16}$ | $0.8105_{0.0031}$ | 2.15 | 4.25 |
| NL2API | ICL@10+CoT | $+11.00_{0.14}$ | $+0.0669_{0.0022}$ | −0.60 | +3.67 | $26.41_{0.12}$ | $0.8368_{0.0052}$ | 1.84 | 5.89 |
| NL2API-2s | Basic | $-9.08_{0.17}$ | $-0.1059_{0.0048}$ | +0.90 | +1.10 | $6.82_{0.25}$ | $0.6912_{0.0037}$ | 2.73 | 2.11 |
| NL2API-2s | ICL@10 | $-6.21_{0.20}$ | $-0.0763_{0.0024}$ | +0.78 | +1.53 | $15.13_{0.19}$ | $0.7761_{0.0027}$ | 2.31 | 5.42 |
| NL2API-2s | ICL@10+CoT | $-1.62_{0.15}$ | $-0.0579_{0.0018}$ | +0.65 | +3.97 | $18.69_{0.26}$ | $0.8005_{0.0028}$ | 2.16 | 6.23 |
| OperatorLLM | Basic | $-6.62_{0.20}$ | $-0.0428_{0.0050}$ | +0.51 | +1.61 | $8.60_{0.14}$ | $0.7182_{0.0034}$ | 2.66 | 2.63 |
| OperatorLLM | ICL@10 | $-2.57_{0.17}$ | $-0.0241_{0.0052}$ | +0.17 | +2.94 | $19.58_{0.16}$ | $0.8113_{0.0051}$ | 2.08 | 3.82 |
| OperatorLLM | ICL@10+CoT | $+9.34_{0.13}$ | $+0.0481_{0.0038}$ | −0.82 | +4.94 | $28.48_{0.26}$ | $0.8426_{0.0041}$ | 1.73 | 5.58 |
| PEARL | DynICL@10 w. Aug. | $\mathbf{+20.31_{0.11}}$ | $\mathbf{+0.1209_{0.0023}}$ | **−1.47** | +1.38 | $\mathbf{36.79_{0.19}}$ | $\mathbf{0.8672_{0.0038}}$ | **1.55** | 2.76 |

Table 2: Comparison of PEARL against baselines. In addition to performance metrics, we also report mean latency per chat conversation of each method. Standard error is reported across 6 runs. Note that for our Internal dataset, we report relative numbers w.r.t. NL2API-Basic.

| Setup | Paradigm | Internal | | | MultiWOZ-H | | |
|---|---|---|---|---|---|---|---|
| | | EM% ↑ | F1 ↑ | FED ↓ | EM% ↑ | F1 ↑ | FED ↓ |
| NL2API | ICL@10 + CoT | $+11.00_{0.14}$ | $+0.0669_{0.0022}$ | −0.60 | $26.41_{0.12}$ | $0.8368_{0.0052}$ | 1.84 |
| PEARL | w. SummaryContextualization | $+13.47_{0.19}$ | $+0.0783_{0.0027}$ | −1.04 | $25.39_{0.13}$ | $0.8172_{0.0030}$ | 1.98 |
| PEARL | w. PreferenceContextualization | $+16.39_{0.16}$ | $+0.0976_{0.0034}$ | −1.15 | $33.82_{0.17}$ | $0.8511_{0.0043}$ | 1.68 |
| PEARL | w. PreferenceContextualization<br>w. DynamicICL@10 | $+19.09_{0.15}$ | $+0.1204_{0.0032}$ | −1.31 | $34.93_{0.31}$ | $0.8598_{0.0041}$ | 1.61 |
| PEARL | w. PreferenceContextualization<br>w. DynamicICL@10<br>w. ExemplarGeneration | $\mathbf{+20.31_{0.11}}$ | $\mathbf{+0.1209_{0.0023}}$ | **−1.47** | $\mathbf{36.79_{0.19}}$ | $\mathbf{0.8672_{0.0038}}$ | **1.55** |

Table 3: Impact of each module in PEARL. NL2API results are provided for reference. We report mean performance and standard error across 6 runs. For our Internal dataset, we report relative performance w.r.t. NL2API-Basic.

matching to calculate the evaluation metrics (refer to Appendix A.2).

**Baseline Comparison** Table 2 reports the performance of PEARL in comparison to baselines on both Internal and MultiWoZ-H datasets. In summary, PEARL outperforms all baselines by significant margin across all metrics in extracting the user preference accurately. In particular, PEARL obtains a lift of 10% in exact match, 5% in Micro F-1 and 47% reduction in FED over NL2API (with ICL and CoT) which is the prior production model for preference extraction. We notice similar trend on MultiWoZ-H, where we obtain improvement of 10% in exact match and 3% in F1. In addition to performance, PEARL reduces inference latency by 110% on Internal test set over the production model. Similarly, we see reduction in latency from $5.89s$ to $2.76s$ on MultiWoZ-H. This reduction in latency primarily comes from having no chain-of-thoughts reasoning steps for any LLM agent in PEARL. OperatorLLM is precise but slow at utterance-level, however, it still lags behind PEARL in performance.

**PEARL Ablation:** To study the effect of each component in PEARL towards the task of navigating customer preferences, we perform a detailed ablation study. Table 3 shows that the proposed ContextualizationAgent agent outperforms a simpler summary-based approach, as summarizing conversations loses preference details amidst noisy information. Adding dynamic in-context learning and exemplar augmentation further improves results, highlighting the value each PEARL module provides.

## 6.2 Analysis

**Latency:** In comparison with competitive baselines like NL2API, PEARL has much lower in latency which is primarily because no step-by-step Chain-of-Thought is involved in PEARL. Our proposed approach relies solely on the quality of retrieved set of ICL exemplars (the retrieval is an embedding match, hence it takes negligible time to retrieve). Unlike NL2API, which generated several CoT steps to arrive at the answer, PEARL generates the answer without generating any thoughts.

**Number of Preferences:** As the number of preferences of the customer increases, the generated set of preferences also explodes. To analyze the performance of PEARL in depth, we conduct a study

| Setup | Encoder | Internal | | | MultiWOZ-H | | |
|---|---|---|---|---|---|---|---|
| | | EM% ↑ | F1 ↑ | RED ↓ | EM% ↑ | F1 ↑ | RED ↓ |
| PEARLw/o Exem.G. | MiniLM-L6 | $+13.00_{0.15}$ | $+0.0894_{0.0031}$ | $-0.76$ | $29.35_{0.13}$ | $0.8402_{0.0040}$ | 2.52 |
| PEARLw/o Exem.G. | Instructor | $+18.34_{0.17}$ | $+0.0935_{0.0012}$ | $-1.11$ | $33.89_{0.22}$ | $\mathbf{0.8647_{0.0059}}$ | 1.77 |
| PEARLw/o Exem.G. | UDR | $+16.71_{0.19}$ | $+0.0909_{0.0034}$ | $-1.02$ | $\mathbf{34.93_{0.31}}$ | $0.8598_{0.0041}$ | **1.61** |
| PEARLw/o Exem.G. | Internal* | $\mathbf{+19.23_{0.11}}$ | $\mathbf{+0.1114_{0.0027}}$ | $\mathbf{-1.31}$ | _ | _ | _ |

Table 4: Impact of different text encoders in Dynamic Exemplar Retrieval on performance. We report metrics and standard error across 6 runs. On internal dataset, we report performance relative to NL2API-Basic. Notation: Exem.G. is ExemplarGenerationModule and w/o is 'without'. Also 'Internal*' is the proprietary deep embedding model we only used on our Internal dataset.

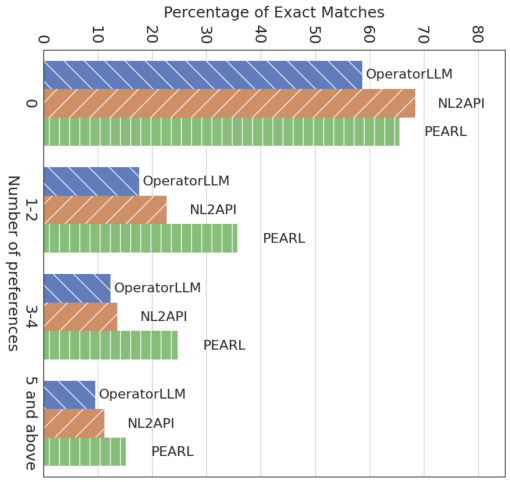


Figure 2: PEARLvs baselines w.r.t. number of preferences per chat session on MultiWoZ-H.

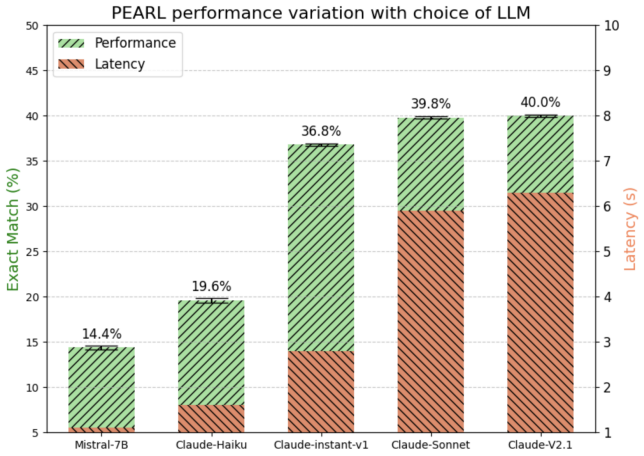


Figure 3: Effect of choice of LLM in PEARL. The results are presented on MultiWoZ-H.

where, we divide our evaluation set into different bins based on the number of preferences in the ground truth (see Figure 2). We observe that PEARL is comparable on the subset of data having zero refinement preferences, and outperforms each baseline significantly when at least 1 preference is mentioned.

**Choice of LLM:** In order to see the effect of choice of LLM for PEARL, we choose different LLMs to be used in `PreferencePickerAgent`. In addition to `Claude-instant-v1`, we also benchmark `Mistral-7b`, `Claude-v2.1`, `Claude-Haiku` and `Claude-Sonnet`. Refer to Figure 3 for performance and latency of each model. We observe that among the `Claude` family, `Claude-Haiku` under-performs compared to other models. `Mistral-7B` also provides comparable performance to `Claude-Haiku`. We also note that `Claude-instant-v1` not only delivers comparable performance to `Claude-v2.1` and `Claude-Sonnet` but it also has the least latency among them.

**Impact of Exemplar encoder:** We experiment with several different encoders in Dynamic Exemplar Retrieval. Specifically, we try out public deep models MiniLM-L6 (Wang et al., 2020a), Instructor (Su et al., 2023) and UDR (Li et al., 2023) (Universal Demonstration Retriever which specializes in ICL retrieval) as the encoder $\mathcal{S}$ in the module. For our Internal test set, we use our Internal/proprietary deep model to obtain text representations. This deep model is specifically trained on the filter keys and values of ecommerce domain. We record our findings in Table 4 and notice that UDR provides the best performance on MultiWoZ-H. For Internal test set, we see that our proprietary model provides the best performance since the neural network is trained on in-domain filter keys and values.

**Scale of Synthetic Exemplars:** In order to study the effect of the scale of synthetic exemplars, we study PEARL as we increase the synthetic data generated from Exemplar Generation Module. To make the study more transparent and indicative of the impact of generated synthetic data, we do not consider the 100 hand-labelled ICL-set. We vary the number of generated exemplars from $[10, 50, 500, 1000]$ and note the performance of PEARL on our task.

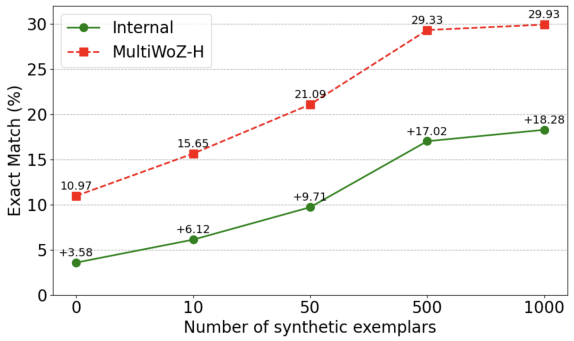


Figure 4: Scaling synthetic exemplars generated by `GenerativeAgent` module. Here, we only use synthetically generated exemplar set. For internal dataset, we report relative improvements over NL2API-Basic.

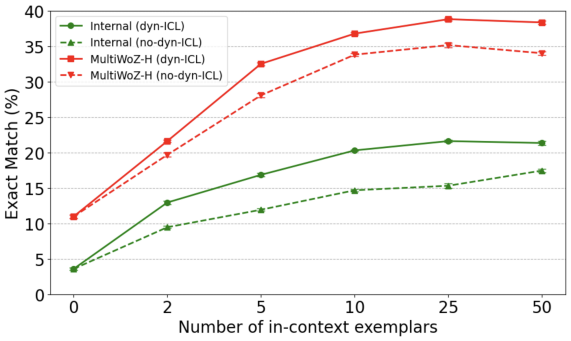


Figure 5: Effect of number dynamic ICL exemplars in PEARL. We also compare against PEARL with randomly sampled exemplars. For internal dataset, we report relative improvements over NL2API-Basic.

As shown in Figure 4, we see that the number of generated exemplars does indeed affect the performance of PEARLpositively (over 10% improvement on EM as we scale from 10 to 1000 exemplars (Nguyen and Wong, 2023; Agarwal et al., 2024; Wang et al., 2024b)).

**Number of retrieved exemplars:** Table 2 shows that PEARL outperforming baselines with 10 dynamically retrieved in-context learning (ICL) examples. Evaluating PEARL by varying ICL examples (c.f. Fig 5) from 0 to 50 reveals a positive correlation between performance and example count, aligning with prior work (Nguyen and Wong, 2023; Agarwal et al., 2024; Wang et al., 2024b). Notably, increasing examples from 2 to 50 only added around 0.5s latency, suggesting potential for further performance gains. Comparing against randomly sampled ICL examples shows higher variability but confirms the significant contribution of dynamically retrieved examples in enhancing PEARL's performance. The dynamic retrieval of relevant examples clearly outperforms random sampling, with minimal latency impact from increasing example count.

## 7 Conclusion

In this work, we address a novel challenge of navigating customer preferences in a conversational setting. We proposed PEARLwhich is a collaborative multi-agent way of handling this problem. We show that our proposed approach not only outperforms the current production systems, but also has lower latency. We learn that exemplar retrieval and breaking down complex tasks into simpler subtasks during inference is an effective approach to achieve promising results. However, retrieving top-$k$ exemplars might not always be the best idea since diversity and even negative exemplars are helpful to large language models. In future, we would like to study a smarter way of retrieving exemplars during inference to assist our prediction module complete the task with better performance.

## 8 Limitations

**Reliance on Exemplars**: Effectiveness of PEARL hinges on the quality of its ICL exemplars. Variability in data quality or coverage may hinder the system's ability to generalize effectively across different user preferences and conversational styles.

**Domain Generalization**: While PEARL demonstrates robust performance in the e-commerce domain, its applicability may vary in domains with distinct conversational dynamics or less structured data. Adapting the framework to diverse domains would necessitate customization efforts.

**Language Adaptation**: While components of PEARL are theoretically capable of supporting multiple languages, our study predominantly focused on English-language support. Evaluating performance and adapting the framework for non-English languages would require additional datasets and language-specific optimization.

**LLM Dependence**: PEARL involves multiple stages that rely on invoking an LLM, with current prompts optimized specifically for models like Claude. Future exploration with different LLMs would require automating prompt optimization to ensure consistent performance across various models.

## References


Eve Abbey. 2023. Developing multi-turn conversational chatbots for e-commerce.

Rishabh Agarwal, Avi Singh, Lei M. Zhang, Bernd Bohnet, Stephanie Chan, Ankesh Anand, Zaheer Abbas, Azade Nova, John D. Co-Reyes, Eric Chu, Feryal Behbahani, Aleksandra Faust, and Hugo Larochelle. 2024. Many-shot in-context learning. *arXiv preprint arXiv:2404.11018*.

A. Cambon et al. 2023. Early llm-based tools for enterprise information workers likely provide meaningful boosts to productivity.

Robson A. Campêlo, Alberto H. F. Laender, and Altigran S. da Silva. 2023. Using knowledge graphs tonbsp;generate sql queries fromnbsp;textual specifications. In *Advances in Conceptual Modeling: ER 2023 Workshops, CMLS, CMOMM4FAIR, EmpER, JUSMOD, OntoCom, QUAMES, and SmartFood, Lisbon, Portugal, November 6–9, 2023, Proceedings*, page 85–94, Berlin, Heidelberg. Springer-Verlag.

Xusen Cheng, Ying Bao, Alex Zarifis, Wankun Gong, and Jian Mou. 2023. Exploring consumers' response to text-based chatbots in e-commerce. *arXiv preprint arXiv:2401.12247*.

Christopher Clarke, Karthik Krishnamurthy, Walter Talamonti, Yiping Kang, Lingjia Tang, and Jason Mars. 2023. One agent too many: User perspectives on approaches to multi-agent conversational ai. *arXiv preprint arXiv:2401.07123*.

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*.

Xiang Gao, Lei Wang, Jingjing Xu, Yangjun Hou, Zheng Dong, Dawei Yin, and Ji-Rong Wen. 2023a. A survey on large language model based autonomous agents. *arXiv preprint arXiv:2308.11432*.

Xiang Gao, Kaiquan Zhou, Jie Li, Tianyi Tang, Xiaoyang Wang, Yangjun Hou, Yuxiang Min, Baorui Zhang, Jiaxin Zhang, Zheng Dong, et al. 2023b. A survey of large language models. *arXiv preprint arXiv:2303.18223*.

Ece Gumusel, Kyrie Zhixuan Zhou, and Madelyn Rose Sanfilippo. 2023. Conversational ai for personalized shopping assistance. *arXiv preprint arXiv:2402.09716*.

Taicheng Guo, Xiuying Chen, Yaqi Wang, Ruidi Chang, Shichao Pei, Nitesh V. Chawla, Olaf Wiest, and Xiangliang Zhang. 2023a. Unlocking the potential of chatgpt. *arXiv preprint arXiv:2304.02017*.

Taicheng Guo, Xiuying Chen, et al. 2023b. Large language model based multi-agents: A survey of progress and challenges. *arXiv preprint arXiv:2403.02164*.

M. Haslberger et al. 2023. No great equalizer: Experimental evidence on ai in the uk labor market. *SSRN Working Paper 4594466*.

Sebastian Hofstätter, Jiecao Chen, Karthik Raman, and Hamed Zamani. 2023. Fid-light: Efficient and effective retrieval-augmented text generation. *arXiv preprint arXiv:2304.14619*.

Gautier Izacard and Edouard Grave. 2021. Leveraging passage retrieval with generative models for open domain question answering. *In Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 874–880.

Guochao Jiang, Zepeng Ding, Yuchen Shi, and Deqing Yang. 2024. P-icl: Point in-context learning for named entity recognition with large language models. *Preprint*, arXiv:2405.04960.

Haiyun Jiang, Zhaochun Ren, Yangjun Hou, Wayne Xin Zhao, and Ji-Rong Wen. 2023. Retrieval-augmented generation for large language models. *arXiv preprint arXiv:2312.10997*.

Nobuhiro Kaji. 2023. Lattice path edit distance: A romanization-aware edit distance for extracting misspelling-correction pairs from japanese search query logs. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 233–242.

Y. Kim and et al. 2023. Techniques for generating in-context learning exemplars. *arXiv preprint arXiv:2311.06668*.

Mayank Kothyari, Dhruva Dhingra, Sunita Sarawagi, and Soumen Chakrabarti. 2023. CRUSH4SQL: Collective retrieval using schema hallucination for Text2SQL. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 14054–14066, Singapore. Association for Computational Linguistics.

Patrick S. H. Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *In Advances in Neural Information Processing Systems*, 33:9459–9474.

Xiaolin Li, Yue Zhang, Xiaohui Zhang, and Xiaolin Zhang. 2021. Customer preference modeling and personalized recommendation in chatbots. *IEEE Access*, 9:166861–166868.

Xiaonan Li, Kai Lv, Hang Yan, Tianyang Lin, Wei Zhu, Yuan Ni, Guotong Xie, Xiaoling Wang, and Xipeng Qiu. 2023. Unified demonstration retriever for in-context learning. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4644–4668, Toronto, Canada. Association for Computational Linguistics.

Jiajia Liu, Mengyuan Yang, Yankai Yu, Haixia Xu, Kang Li, and Xiaobo Zhou. 2023a. Enhancing customer experience with ai-driven conversational agents. *arXiv preprint arXiv:2306.04325*.

Jiajia Liu, Mengyuan Yang, Yankai Yu, Haixia Xu, Kang Li, and Xiaobo Zhou. 2023b. A study on chinese social perspective regarding chatgpt. *arXiv preprint arXiv:2306.04325*.

Xia Liu, Xiaolin Li, Xiaohui Zhang, and Xiaolin Zhang. 2019. A deep learning model for chatbot preference modeling and personalized recommendation. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 49(11):3725–3733.

Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2023. Rethinking the role of demonstrations: What makes in-context learning work? *arXiv preprint arXiv:2304.14619*.

Tai Nguyen and Eric Wong. 2023. In-context example selection with influences. *arXiv preprint arXiv:2302.11042*.

Shishir G. Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. 2023. Gorilla: Large language model connected with massive apis. *arXiv preprint arXiv:2305.15334*.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. 2023. Toolllm: Facilitating large language models to master 16000+ real-world apis. *Preprint*, arXiv:2307.16789.

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

Thilina C Rajapakse. 2023. Dense passage retrieval: Architectures and augmentation methods. *arXiv preprint arXiv:2304.14619*.

Stephen Roller, Emily Dinan, Naman Goyal, Da Ju, Mary Williamson, Yinhan Liu, Jing Xu, Myle Ott, Eric Michael Smith, Y-Lan Boureau, and Jason Weston. 2021. Recipes for building an open-domain chatbot. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*.

Vinay M. S., Minh-Hao Van, and Xintao Wu. 2024. In-context learning demonstration selection via influence analysis. *Preprint*, arXiv:2402.11750.

Hongjin Su, Weijia Shi, Jungo Kasai, Yizhong Wang, Yushi Hu, Mari Ostendorf, Wen tau Yih, Noah A. Smith, Luke Zettlemoyer, and Tao Yu. 2023. One embedder, any task: Instruction-finetuned text embeddings. *Preprint*, arXiv:2212.09741.

Huan Sun, Hao Ma, Xiaodong He, Wen-tau Yih, Yu Su, and Xifeng Yan. 2016. Table cell search for question answering. In *Proceedings of the 25th International Conference on World Wide Web*, WWW '16, page 771–782, Republic and Canton of Geneva, CHE. International World Wide Web Conferences Steering Committee.

Liang Wang, Nan Yang, and Furu Wei. 2024a. Learning to retrieve in-context examples for large language models. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1752–1767, St. Julian's, Malta. Association for Computational Linguistics.

Liang Wang, Nan Yang, and Furu Wei. 2024b. Learning to retrieve in-context examples for large language models. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics*, pages 1752–1767.

Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. 2020a. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. *Preprint*, arXiv:2002.10957.

Xiaolin Wang, Xiaolin Zhang, Xiaohui Zhang, and Xiaolin Zhang. 2020b. A deep learning approach for chatbot sentiment analysis and personalized response generation. *IEEE Transactions on Intelligent Transportation Systems*, 21(11):5728–5737.

W. A. Woods. 1973. Progress in natural language understanding: An application to lunar geology. In *Proceedings of the June 4-8, 1973, National Computer Conference and Exposition*, AFIPS '73, page 441–450, New York, NY, USA. Association for Computing Machinery.

Shangqing Xu and Chao Zhang. 2024. Misconfidence-based demonstration selection for llm in-context learning. *Preprint*, arXiv:2401.06301.

Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. 2015. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1321–1331, Beijing, China. Association for Computational Linguistics.

Xiaoxue Zang, Abhinav Rastogi, Srinivas Sunkara, Raghav Gupta, Jianguo Zhang, and Jindong Chen. 2020. Multiwoz 2.2: A dialogue dataset with additional annotation corrections and state tracking baselines. In *Proceedings of the 2nd Workshop on Natural Language Processing for Conversational AI, ACL 2020*, pages 109–117.

Xiaofei Zhang, Xiaolin Li, Xiaohui Zhang, and Xiaolin Zhang. 2020. A deep learning framework for chatbot

preference modeling and personalized recommendation. *IEEE Transactions on Neural Networks and Learning Systems*, 32(1):167–178.

Wayne Xin Zhao, Yangjun Hou, Zhaochun Ren, Yaliang Ding, Dawei Yin, and Ji-Rong Wen. 2023. When large language models meet personalization. *arXiv preprint arXiv:2307.16376*.

Kaijie Zhu, Qinlin Zhao, Hao Chen, Jindong Wang, and Xing Xie. 2023. Pctoolkit: A unified toolkit for prompt compression. *arXiv preprint arXiv:2403.17411*.

# A Appendix

## A.1 Dataset and annotation details

We only consider explicit preferences which can be mapped/quantified to a combination of filter keys and values. For example, "intel processors" can be mapped to CPU Type filter values like Intel Core i3, i5, i7, i9, Celeron, and Pentium. Consequently, subjective preferences like "lightweight" or "good battery" are not considered as explicit preferences due to their ambiguity. Further, we note that high-ambiguity preferences related to use-cases, such as "laptop for stock trading" are ignored as they cannot be directly mapped to known filters.

**MultiWoZ-H** : For MultiWoZ dataset, we process the annotated data at chat session level. We note that the filter value of 'dontcare' was ambiguous for our usecase. We reason that if a filter value is 'dontcare', it suggests that the customer has no preference for this particular filter. Thus, we discard such keys from the gold preference dictionaries that have 'dontcare' as filter value.

## A.2 Evaluation Details

In all our experiments, we calculate Micro F1 at a key-value pair level. We consider a key-value pair to be true positive if the same key-value pair exists in the gold preference dictionary. However, if the key does not exist in the gold dictionary, we consider it to be false positive. If the key exists in gold dictionary but not in predicted dictionary, we consider it as false negative. Finally, if the key exists in both the dictionaries but the values are different, we consider it a false negative as well as false positive.

Given the unbounded nature of some filter key values in MultiWoZ-H dataset (for example 'Hotel Name' can be any string), we resort to a fuzzy matching based logic. For this, we use 'thefuzz' library[2] for Levenshtein distance. Specifically, we consider $\{k, v_{pred}\}$ to be the same as $\{k, v_{gold}\}$ if and only if, the distance ratio is at least $\epsilon$. Note that we use $\epsilon = 0.75$ in all our experiments (on MultiWoZ-H).

**Error Computation for Internal dataset:** Note that we calculate standard error metrics across 6 runs. However, in case of our Internal dataset, since we are providing relative performance against NL2API-Basic, we make sure to follow the best practices of error computation. Specifically, for the 6 runs of NL2API-Basic, and 6 runs of any other approach, we calculate average all-pair error and report that as the standard error in case of our Internal dataset.

## A.3 Prompting Details

As we will see in the prompts in Appendix A.4, we need the exhaustive filter space to provide signal to the LLM that the filter keys and values are bounded in this space. However, for MultiWoZ-H, as we said in Appendix A.2, the filter values are unbounded for some filter keys. Therefore, we follow a strategy similar to P-ICL work for NER (Jiang et al., 2024). We obtain the unique set of values for each filter key in our ICL set. Using the set of all values (observed in ICL set) for filter key $k$, we obtain their deep representations using SBERT. Further, we use K-Means clustering method and nearest neighbor decoding strategy are to identify the point filter values for each filter key.

## A.4 Prompts

We provide exact prompts we used for our experimentation on MultiWoZ-H dataset. The prompts used for Internal dataset were similar.

[2] https://github.com/seatgeek/thefuzz

## Prompt 1: NL2API

**Prompt:**
“Human:
<purpose>
You are an expert in detecting filters or customer preferences from a chatbot conversation between a bot and a user about recommendations for hotels. The user might specify information regarding their preferences for such places.
</purpose>

<filterdetails>Each row in filter table consists of a filter key and example possible list of values it can take.</filterdetails>

<filters>
’area’: [’north’, ’east’, ’south’, ’west’, ’centre’],
’bookday’: [’tuesday’, ’thursday’, ’wednesday’, ’monday’, ’friday’, ’saturday’, ’sunday’],
’bookpeople’: [’4’, ’6’, ’5’, ’3’, ’7’, ’2’, ’1’, ’8’],
’bookstay’: [’4’, ’3’, ’2’, ’5’, ’7’],
’internet’: [’yes’, ’no’],
’name’: [’archway house’, ’acorn guest house’, ’aylesbray lodge guest house’, ’ashley hotel’,
’arbury lodge guesthouse’, ’hobsons house’, ’alexander bed and breakfast’, ’autumn house’,
’hamilton lodge’, ’bridge guest house’],
’parking’: [’yes’, ’no’],
’pricerange’: [’moderate’, ’cheap’, ’expensive’],
’stars’: [’4’, ’3’, ’2’, ’0’],
’type’: [’guesthouse’, ’hotel’]
</filters>

<instruction>Given the information in the above table, and the chat conversation tourist location preferences. Your task is to identify the filters that the customer has specified and construct JSON with filter key as key and filter values as value.</instruction>
<rules>
<rule>In the conversation, a customer might ask other non-preferential related questions to the bot. Make sure to only include a filter if the customer SPECIFIES it.</rule>
<rule>Select a filter ONLY if it is mentioned and preferred by the customer. You need to be highly precise about which filters are being specified by the customer and not assume a filter.</rule>
<rule>If there are multiple mentions of the same filter, choose the latest specified filter value for that filter.</rule>
</rules>
”

**In-context example:**
“<example>
<conversation>
`{icl_conversation}`
</conversation>

Assistant:
<thinking>
`{icl_cot_steps}`
</thinking>
<response> `{icl_ground_truth}` </response>
</example>
”

## Prompt 2: PEARL-PreferencePrediction

**Prompt:**
“Human:
<purpose>
You are an expert in detecting filters or customer preferences from a customer query about recommendations for hotels. The user might specify information regarding their preferences for such places.
</purpose>

<filterdetails>Each row in filter table consists of a filter key and example possible list of values it can take.</filterdetails>

<filters>
’area’: [’north’, ’east’, ’south’, ’west’, ’centre’],
’bookday’: [’tuesday’, ’thursday’, ’wednesday’, ’monday’, ’friday’, ’saturday’, ’sunday’],
’bookpeople’: [’4’, ’6’, ’5’, ’3’, ’7’, ’2’, ’1’, ’8’],
’bookstay’: [’4’, ’3’, ’2’, ’5’, ’7’],
’internet’: [’yes’, ’no’],
’name’: [’archway house’, ’acorn guest house’, ’aylesbray lodge guest house’, ’ashley hotel’,
’arbury lodge guesthouse’, ’hobsons house’, ’alexander bed and breakfast’, ’autumn house’,
’hamilton lodge’, ’bridge guest house’],
’parking’: [’yes’, ’no’],
’pricerange’: [’moderate’, ’cheap’, ’expensive’],
’stars’: [’4’, ’3’, ’2’, ’0’],
’type’: [’guesthouse’, ’hotel’]
</filters>

<instruction>Given the information in the above table, and the customer query about their preferences. Your task is to identify the filters that the customer has specified and construct JSON with filter key as key and filter values as value.</instruction>
<rules>
<rule>Select a filter ONLY if it is mentioned and preferred by the customer. You need to be highly precise about which filters are being specified by the customer and not assume a filter.</rule>
<rule>If there are multiple mentions of the same filter, choose the latest specified filter value for that filter.</rule>
</rules>
”

**In-context example:**
“<example>
<query> `{icl_preference_contextualized_query}` </query>

Assistant:
<response> `{icl_ground_truth}` </response>
</example>
”

## Prompt 3: PEARL-PreferenceContextualization

**Prompt:**
"Human:
<purpose>
You are given a chatbot conversation between a bot and a user about recommendations for hotel recommendations. You need to read the conversation and specify the preferences of the customer in a single sentence.
</purpose>

<rules>
<rule>You will be given a conversation between a chatbot (system) and the user.</rule> <rule>Read the customer utterances in the conversation step by step and determine the LATEST preferences of the customer and summarize them in a single sentence.</rule>
<rule>Some preferences of the customer might be overriden in the later part of the chat. Hence, make sure to summarize the LATEST preference of the customer.</rule>
<rule>In the conversation, a customer might ask other non-preferential related questions to the bot. Make sure to only include a preference in the summary if the customer actually SPECIFIES it.</rule>
<rule>In the conversation, a customer might be interested in a preference in the start of the conversation, but replaces it with another preference later. Only include the new preference and not the old one.</rule>
<rule>If the customer provides conflicting preferences, pick the one that customer has suggested more recently. The most recent preferences are at the end of the chat session.</rule>
</rules>
"

**In-context example:**
"<example>
<conversation>
`{icl_conversation}`
</conversation>

Assistant:
<response> `{icl_preference_contextualized_query}` </response>
</example>
"

## Prompt 4: PEARL-ExemplarGeneration

**Prompt:**
"Human:
<purpose>
You are given a summary of the preferences of a customer who wants to search for accomodations like hotel/guesthouses. Read the preference dictionary of the customer and write a single line customer query that corresponds to the same preferences as mentioned in the dictionary.
</purpose>

<instruction>Look at the preference dictionary provided in between the <preference-dictionary></preference-dictionary> tags and generate query which encompasses ALL the preferences in the dictionary.</instruction>

<rules>
<rule>Include ALL the preferences mentioned in the dictionary and write the query in between the <query></query> tags.</rule>
<rule>Make sure that the generated query is about hotels/guesthouses.</rule>
</rules>
"

**In-context example:**
"<example>
<preference-dictionary>
`{icl_preference_dictionary}`
</preference-dictionary>

<query> `{icl_generated_query}` </query>
</example>
"