TopoSem: In-Context Planning with Semantically-Informed Tooling Graph Similarity

Shengjie Liu*
Amazon
zycjlsj@amazon.com

Huilin Lu*
Amazon
huilinlu@amazon.com

Li Dong
Amazon
ldonga@amazon.com

Jason Zhu
Amazon
jasonzhu@amazon.com

Manish Gawali
Amazon
mdgawali@amazon.com

Alice Zhou
Amazon
bingrou@amazon.com

Abstract

Designing intelligent assistants for e-commerce sellers presents significant challenges, primarily due to the abstract nature of seller queries and the complexity of orchestrating multiple internal tools. In-context planning (ICP) has emerged as a promising adaptive problem-solving approach for this setting. However, selecting effective exemplars for ICP remains a difficult problem, largely because of the intricate coordination among underlying APIs. Relying solely on semantic similarity between textual queries can misguide large language models (LLMs) during planning, as semantically similar queries may correspond to vastly different API execution graphs. To address this, we propose TopoSem, a novel framework that enhances ICP by jointly considering the topological distance of API execution graphs and the semantic differences in API payloads. We leverage a contrastive learning approach to learn meaningful embeddings, which are then used in an enhanced dynamic clustering mechanism to reduce noise and redundancy in exemplar selection. Empirical results demonstrate that TopoSem substantially outperforms traditional exemplar selection methods in terms of planning accuracy and generalization, particularly in scenarios involving complex API orchestration.

CCS Concepts

• **Information systems** → **Language models**.

Keywords

LLM, Agentic Workflow, Retrieval, Planning

ACM Reference Format:
Shengjie Liu, Huilin Lu, Li Dong, Jason Zhu, Manish Gawali, and Alice Zhou. 2025. TopoSem: In-Context Planning with Semantically-Informed Tooling Graph Similarity. In *Proceedings of LLM4ECommerce Workshop at the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2 (LLM4ECommerce Workshop at KDD '25).* ACM, Toronto, CA, 10 pages. https://doi.org/XXXXXXX.XXXXXXX

*These authors contributed equally to this work.

LLM4ECommerce Workshop at KDD '25, Toronto, ON, Canada

ACM ISBN 978-1-4503-XXXX-X/2018/06
https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

Large Language Models (LLMs) ([3–5, 24, 32]) have been at the forefront of advancing artificial intelligence, marking significant breakthroughs in diverse fields. The planning capabilities of LLMs, particularly their ability to use tools ([30, 31]), enable them not only to execute commands and perform web searches but also to enhance their advanced mathematical reasoning abilities. LLM Compiler [12] and its subsequent work ([7, 8]) propose constructing tooling usage as a directed acyclic graph (DAG) to enable the parallel execution of independent tools, thereby improving tool-calling efficiency. CodeAct [26] and CodePlan [27] propose leveraging the generation of pseudo-Python code to outline high-level reasoning processes for complex multi-step reasoning tasks, where each tool usage is represented as a function call within the code. ReWOO [28] proposes a modular framework that decouples the reasoning process from the external observations of each tool usage, thereby reducing token consumption and improving efficiency. [16] clusters the provided tools into groups of toolkits, plans at the toolkit level, and replans by selecting tools within the same toolkit if error comes out. [17] proposes a method called Predictive-Decoding, which leverages Model Predictive Control from the optimal control field to mitigate early errors in planning and promote non-myopic planning, thereby enhancing overall accuracy. ReasonFlux [29] proposes a framework in which the LLM reasons over template fields, executes tools based on the templates, and employs reinforcement learning to improve planning accuracy using an action completion reward.

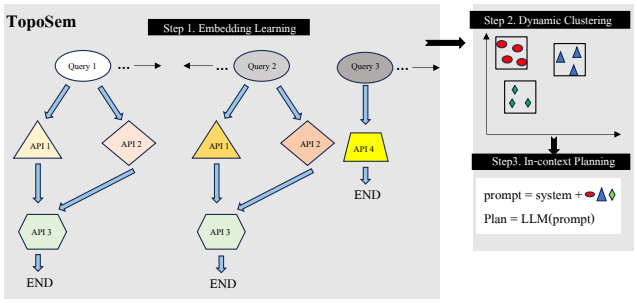

Figure 1: Illustration of the TopoSem pipeline. During embedding learning, TopoSem separates queries with different APIs graphs despite similar semantics, while aligning those with the same APIs graph and semantically similar payloads. Color intensity reflects semantic similarity.

Existing literature on tool-using capabilities primarily focuses on general real-world API usage [20], such as *send email* and *make calendar*, as well as related functionalities in *web search* applications like Manus. These APIs are uni-functional in that each API is designed to perform a single action or answer a single question and the description. Meanwhile, the descriptions and argument definitions of these APIs are easily recognized by LLMs such as GPT-4o [19] and Claude 3.5. For instance, the description of the *send-email* API can be simply stated as *send the email to the destination*, with arguments including *email content*, *email address source*, and *email address destination*. However, the APIs that a selling assistant interacts with in the e-commerce domain—such as those related to inventory status and performance metrics—are significantly more complex and domain-specific, making them difficult for general-purpose LLMs to interpret accurately. As a result, in-context planning (ICP) has become a widely adopted approach, wherein exemplars of relevant API executions are provided to the LLMs prior to plan generation.

Selecting appropriate in-context planning artifacts is critical, as irrelevant exemplars can mislead the LLM and result in suboptimal or incorrect planning. [33] enhances in-context planning and tool-use capabilities by dynamically clustering action sequences and selecting highly similar exemplars based on action sequence similarity. However, defining similarity using the longest common subsequence of actions may be insufficient for selecting effective exemplars, as seller assistants often rely on more complex API graphs—rather than simple linear API chains—to address seller-related queries. In this paper, We propose TopoSem, a novel framework that enhances ICP by jointly considering the topological distances of API execution graphs and the semantic differences in API payloads to identify more effective exemplar candidates. These candidates are then used in an enhanced dynamic clustering mechanism to reduce noise and redundancy in exemplar selection.

In summary, this work makes several pivotal contributions:

- We define a novel distance metric between two queries based on the API execution graph edit distance (GED), where node and edge operations are weighted according to the semantic similarity of their corresponding API payloads.
- We propose a synthetic data augmentation pipeline to augment the existing (query, API graph) pairs, aiming to better adjust the semantic embeddings of queries under the defined distance metric.
- Extensive experimental results demonstrate the effectiveness of TopoSem, highlighting the importance of incorporating the topological structure of API execution graphs and the critical role of high-quality exemplars.

2 Preliminaries

2.1 LLM Reasoning with Tools

Given a user query x and a pretrained LLM $\rho_\theta(\cdot)$, the LLM generates an API execution plan represented as a graph with $p = \{\mathcal{P}_1, \ldots, \mathcal{P}_n\} \sim \rho_\theta(p \mid \mathcal{T}, \mathcal{D}, x)$, where p is the plan list after topological sorting, $\mathcal{T}$ is the set of available tools, and $\mathcal{D}$ is the collection of descriptions for all available tools. At each step t, the LLM generates an intermediate reasoning output $r_t \sim \rho_\theta(r_t \mid \mathcal{T}, \mathcal{D}, x, p, O_1, \ldots, O_{t-1})$ and executes the plan step $\mathcal{P}_t$ to obtain the observation O_t. The final response is then generated as $\mathcal{R} \sim \rho_\theta(\mathcal{R} \mid \mathcal{T}, \mathcal{D}, x, p, O_1, \ldots, O_n)$. In-context planning occurs when the LLM is provided with exemplar plans to guide the planning process: $p \sim \rho_\theta(p \mid \mathcal{T}, \mathcal{D}, x, p_1, \ldots, p_m)$, where p_i is an exemplar plan and m is the number of exemplars.

2.2 APIs Execution Graph

Given the plan p determined by the LLM, it can be represented as a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = (v_1, \ldots, v_n)$ is the set of nodes and $\mathcal{E} = (e_1, \ldots, e_m)$ is the set of edges. The node v_1 corresponds to the seller's query, and v_n denotes the final node that collects observations and returns the response to the seller. The intermediate nodes $v_2, \ldots, v_{n-1}$ represent API calls, each with a node attribute corresponding to its API payload. We represent all API payloads as text strings for simplicity and leave the argument parsing to future exploration. e_i denotes a dependency between two APIs, where the output of the source API serves as the input to the target API. For evaluation and task assignment

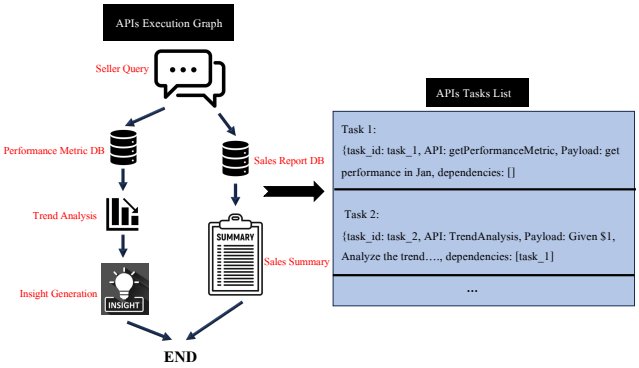

Figure 2: Illustration of the API execution graph and its equivalent representation as a list of API tasks.

convenience, the API graph is represented as a list of tasks, where each task is defined: {*task_id* : task_3, *API* : analyzeData, *payload* : Analyze data from $2, *dependencies* : [task_2]}.

3 Methodology

In the e-commerce domain, both API descriptions and orchestrations are significantly more complex, making it difficult for LLMs pretrained on general public data to accurately and robustly generate plans for diverse seller queries. Therefore, in-context planning (ICP) has been widely adopted during the planning stage to help LLMs generate more accurate execution plans. In ICP, selecting the appropriate plan artifacts and presenting them effectively in the prompt is crucial. TopoSem offers an end-to-end automated approach for selecting diverse and relevant plan artifacts.

3.1 Tooling Graph-based Similarity Calculation

In ICP, given a user query, we first retrieve exemplars from a ground-truth database of (query, API graph) pairs based on the semantic similarity between the user query and queries in the database, and then retrieve the corresponding API graphs. However, as discussed

in [11, 33], relying solely on semantic similarity can result in selecting suboptimal in-context plan artifacts, potentially introducing misleading exemplars that degrade planning performance.

The core of our approach is to measure similarity between two queries by considering both the structure of their underlying API execution graphs and the semantic similarity of their payloads. Let q_1 and q_2 be two user queries, with their corresponding API execution graphs denoted as $\mathcal{G}(q_1)$ and $\mathcal{G}(q_2)$. We compute the graph edit distance (GED) between them using the Exact GED algorithm [1], as the API graphs are typically of manageable size. We customize the node substitution cost in the GED computation based on the following mechanism:

- In e-commerce, a single API can often address multiple in-scope queries, leading to variations in payloads even when the API node remains the same. Therefore, we define part of the node substitution cost based on the semantic difference between the API payloads.
- The other component of the substitution cost is a binary metric that indicates whether the APIs themselves are the same.

We define sim_{payload} as the semantic similarity between the payloads, and sim_{api} as a binary indicator of whether two API nodes correspond to the same API. The node substitution cost is then defined as:

$$\text{cost} = 1 - (w_p \cdot sim_{\text{payload}} + w_a \cdot sim_{\text{api}}), \tag{1}$$

where w_p and w_a are weighting coefficients for payload similarity and API identity, respectively, and they satisfy $w_p + w_a = 1$. In our case, we set $w_p = w_a = 0.5$. In this way, we define a semantically-informed topological distance between two queries. Since this distance is always positive, we convert it into a similarity score within the range $(0, 1)$ using the following formulation:

$$TopoSem_{\text{sim}}(q_1, q_2) = \exp\left(-\left(\text{GED}(\mathcal{G}(q_1), \mathcal{G}(q_2)) + \tau\right)\right), \tag{2}$$

where τ is a temperature hyperparameter, which we set to 0.01 in our experiments.

During the production stage, the API execution graph of an incoming seller query is not available. This motivates us to directly embed the seller query using a learned embedding model. We also precompute and index the queries in the (query, API graph) database using the same model. The embedding model is trained such that the similarity between query embeddings reflects our defined $TopoSem_{\text{sim}}$ (See Figure 1).

3.2 (query, APIs graph) Data Augmentation

To train the embedding model, we begin by fine-tuning a pre-trained model rather than training one from scratch, to ensure better generalizability. Our goal is that if two queries are similar under $TopoSem_{\text{sim}}$, then any other queries semantically similar to these should also be close in the embedding space according to $TopoSem_{\text{sim}}$.

Organizations typically start with a small pool of APIs and often lack sufficient ground-truth (query, API graph) pairs. Consequently, it is challenging to initiate domain-specific embedding fine-tuning to generate the necessary volume of triplets (query_1, query_2, similarity score). [22] proposes generating synthetic queries along with corresponding ground-truth plan artifacts in a scalable manner. However, relying solely on this pipeline may lead to a left-tailed distribution of similarity scores, where most query pairs are not similar under $TopoSem_{\text{sim}}$. Moreover, since some query pairs exhibit subtle relative differences in similarity, this approach may fail to capture and rank these nuanced similarities accurately.

To build an effective synthetic data augmentation pipeline for constructing triplets for fine-tuning, we propose the following four scenarios to strategically augment the ground-truth (query, API graph) pairs, using LLMs guided by scenario-specific prompts.

- **Scenario 1.** In this scenario, we prompt the LLM to generate query pairs with entirely different semantic contexts. For example, one query might be about *organizing a trip to Italy and requesting hotel contact information*, while another could state *my home in Italy has a power shortage and I need temporary accommodation, along with hotel contact information.* Although these queries are semantically dissimilar, they should exhibit high $TopoSem_{\text{sim}}$ similarity because they share the same API execution graph and identical payloads.
- **Scenario 2 & 3.** Given a pair $(q, \mathcal{G}(q))$, Scenario 2 randomly drops a subset of tasks from the API execution graph $\mathcal{G}(q)$, including any dependent tasks. We then prompt an LLM to generate a new query that reflects this modified task set, producing diverse queries whose API execution graphs and payloads remain identical to the original except for the omitted tasks. Scenario 3 operates similarly, but instead of dropping tasks, it substitutes a subset with entirely new tasks (APIs). For example, the original query might ask about *the ticket price of a concert*, while the substituted query might ask *Are there any tickets left?*
- **Scenario 4.** As discussed earlier, many APIs in e-commerce can handle multiple in-scope questions. Therefore, in Scenario 4, we prompt LLMs to examine the API execution graph of the original query along with the API descriptions for each node. The LLM is then asked to generate entirely new, semantically different payloads that are still valid inputs for the same APIs. Based on these modified payloads, the LLM generates a new query that can be answered by the original API orchestration structure, now using the updated payloads at each node.

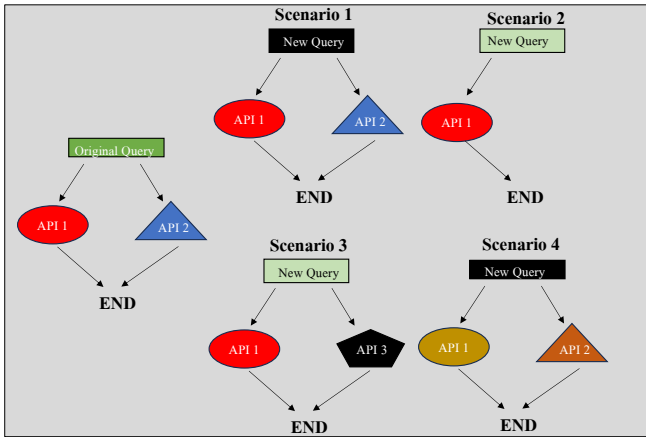

Figure 3: Illustration of four scenarios in the (query, APIs graph) Data Augumentation. Note that Color intensity and color type reflect semantic similarity.

The underlying motivation behind constructing these scenarios is to augment the original (query, API graph) pairs in a way that introduces and captures similarity under $TopoSem_{\text{sim}}$ at various levels. Scenario 1 constructs diverse queries that maintain a high degree of similarity to the original query. In contrast, Scenarios 2, 3, and 4 generate queries that may not be as closely aligned with the original query as those in Scenario 1; however, they are still more similar to the original query than other queries in the original set of (query, API graph) pairs. During the ICP stage, the LLM may still be able to infer the correct plan by leveraging the partially correct API graphs retrieved in response to the queries from Scenarios 2, 3, and 4.

Based on the augmented (query, API graph) pairs, we construct the required triplets $(\text{query}_1, \text{query}_2, \text{similarity score})$ for embedding fine-tuning. Importantly, we do not compute $TopoSem_{\text{sim}}$ for all possible query pairs. Instead, we compute it only between the original query and its augmented variants, as well as between the original query and a randomly selected subset of other queries from the original dataset.

We denote the triplets as $\mathcal{D} = \{(q_{1i}, q_{2i}, s(q_{1i}, q_{2i}))\}_{i=1}^{m}$, where each triplet consists of two queries and their associated similarity score. The embedding model is fine-tuned using the AnglE loss [13], with the following objective:

$$\log\left[1 + \sum_{s(q_i,q_j)>s(q_m,q_n)} \exp\left(\frac{\cos(q_m,q_n) - \cos(q_i,q_j)}{\tau}\right)\right] \quad (3)$$

where (q_i, q_j) are query pairs within the same batch that have a higher similarity score than the reference pair (q_m, q_n), and τ is a temperature parameter.

3.3 Dynamic Clustering to Select Exemplars

With the fine-tuned embedding model obtained from the augmented data stage, we efficiently index the queries in the training set using their corresponding embeddings and store the associated plan artifacts. During inference, we retrieve the top-K artifacts based on embedding similarity using approximate nearest neighbor search methods such as FAISS [10]. These retrieved artifacts are then used as exemplar candidates.

Following [33], we apply Agglomerative Clustering [18] to the retrieved top-K exemplars and select one exemplar from each cluster to form the final exemplar set. This promotes diversity among the exemplars presented in the prompt, while reducing redundancy and noise. For instance, if a seller query involves both performance monitoring and a summary of last month's sales, and the database contains exemplars addressing only one aspect each, clustering enables the selection of both relevant partial exemplars. This allows the LLM to accurately infer a comprehensive plan.

4 Experiments

4.1 Experiment Setup

Dataset. Similar to prior work [16], we adopt ToolBench [20] as our benchmark dataset. ToolBench includes 16,464 APIs and provides three levels of prompts—G_1, G_2, and G_3—for generating queries and corresponding plans using depth-first search (DFS) planning. Specifically, G_1 corresponds to single-tool instructions, G_2 to intra-category multi-tool instructions, and G_3 to intra-collection multi-tool instructions. We repurpose the ToolBench data by randomly selecting 1,000 queries—300 from G_1, 300 from G_2, and 400 from G_3—and use the ground-truth provided in the dataset to construct the corresponding plan artifacts based on the defined task list (see Figure 2).

Given the selected queries, we augment the dataset using the (query, API graph) augmentation pipeline (section 3.2) that generates additional queries along with their corresponding plan artifacts, producing 10 augmented instances for each original scenario. Therefore, the total number of (query, API graph) pairs amounts to 40,000.

To construct the triplets for embedding fine-tuning, we use only the (query, API graph) pairs from the augmented dataset. Specifically, for each original query, we randomly select one augmented query from Scenario 1 and compute $TopoSem_{\text{sim}}$ with all other 39 augmented queries derived from the same original query. Additionally, we randomly sample 61 augmented pairs from other original queries and compute their similarities to complete the triplet construction. This results in a training dataset containing 100,000 triplets. Since the original queries are not used during training, we reserve them for the test set to evaluate in-context planning performance.

To assess the effectiveness of $TopoSem_{\text{sim}}$ in real-world scenarios, we also evaluate it on real-world seller-related APIs, such as status checking or data fetching. We collect synthetic queries in the magnitude of hundreds along with their ground-truth plan artifacts and construct training triplets using the same methodology as applied to the ToolBench dataset.

Baselines. Since the target task is ICP, the selection of exemplars presented in the prompt plays a critical role in determining performance. Therefore, we select baseline models based on their exemplar selection strategies, including zero-shot planning, semantic search and hybrid search methods that combine sparse vector retrieval (e.g., BM25[21]) with semantic similarity. We also evaluate a baseline that selects exemplars based on action sequence similarity [33]. For these methods, we evaluate performance using different base LLMs, including GPT-4o, Claude 3.7 (v1), and Claude 3.5 (v2).

We use Jina Embedding (v3) [23] for computing payload semantic similarity in the $TopoSem_{\text{sim}}$ calculation, and employ BGE-M3 for embedding fine-tuning.

Metrics. For evaluation metrics, similar to [2, 25, 33], we employ *planning accuracy*, which measures the percentage of test samples where the generated plan artifacts correctly call the intended API at each step, including the appropriate dependency APIs.

It is important to note that, in the context of e-commerce, a single API may be capable of answering multiple in-scope questions, and the input payload to each API is in textual form. This corresponds to Scenario 4 in the (query, API graph) augmentation pipeline. Therefore, beyond accurately predicting the API types in the plan, it is also essential to verify the correctness of all payloads. In this context, assessing the semantic consistency between two payloads is a straightforward task for LLMs such as Claude 3.5 (v2). Therefore, we adopt an LLM-as-a-judge [9] approach to measure payload accuracy. Specifically, we prompt the LLM to verify whether the payload is in the correct format and whether it appropriately reflects the query by including the correct entity or temporal information.

4.2 Analysis of the Augmented Data

As mentioned in Section 4.1, we construct the embedding fine-tuning triplets using only the augmented (query, API graph) pairs, while reserving the original queries from ToolBench as the test set to evaluate ICP performance. For each triplet, we randomly select one query from Scenario 1—where the API execution graph and payloads remain consistent, but the narrative context differs significantly—as the anchor. We then compute $TopoSem_{\text{sim}}$ between this anchor and both the other augmented queries of the same original query and those associated with different original queries. Therefore, it is important to verify whether the LLMs used in the augmentation pipeline truly generate distinct narrative contexts in Scenario 1. Otherwise, the similarity between the anchor and original queries may introduce bias into the test set, as the original queries from ToolBench are used for evaluation.

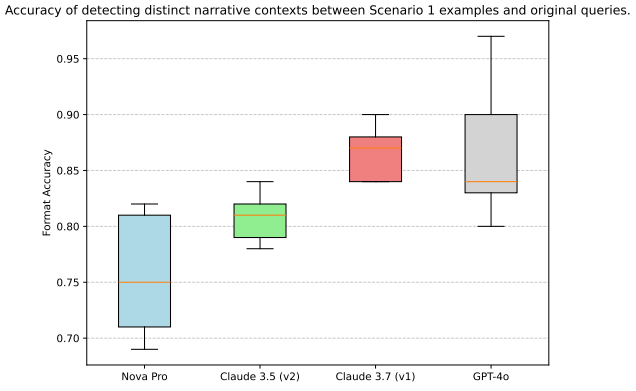

Figure 4: Accuracy of detecting distinct narrative contexts between Scenario 1 examples and original queries for different base LLMs

For each base LLM, we use the Scenario 1 prompt (see Appendix A.1) to generate 2 Scenario 1 examples for each of the 1,000 original queries. We then employ an LLM-as-a-judge prompt to evaluate whether the generated examples differ in narrative context, including changes in verbs or locations. We bootstrap 100 samples from the total generated examples five times, and report the resulting accuracy in Figure 4. As shown in Figure 4, there remains a chance that the base LLM produces a narrative context similar to the original query. Among the models, GPT-4o and Claude 3.7 perform best at following the instructions in the prompt.

A key motivation behind our data augmentation pipeline is to address the issue of a left-skewed similarity score distribution during embedding fine-tuning, as most original queries are not $TopoSem_{\text{sim}}$-similar to one another.

As shown in Figure 5, the augmentation method mitigates the left-skewed distribution to some extent and introduces a density peak near 1, primarily due to the inclusion of Scenario 1 examples. Additionally, the presence of the other three scenarios contributes to a more balanced distribution by introducing density in the middle similarity range. However, a left tail still persists, as each Scenario 1 anchor also computes $TopoSem_{\text{sim}}$ scores against augmented queries from different original queries. By modifying

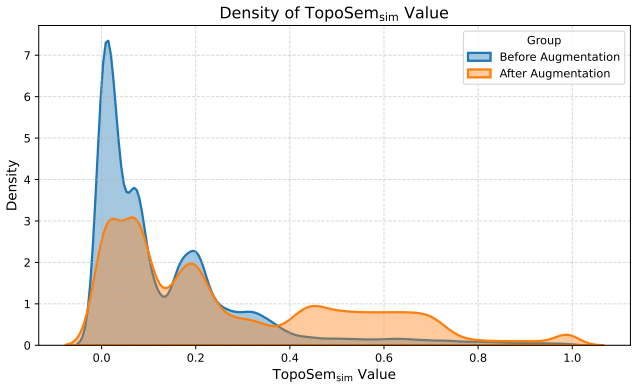

Figure 5: Density of $TopoSem_{\text{sim}}$ similarity scores before and after augmentation.

the similarity distributions through augmentation, we enable the embedding model to learn the relative ordering of similarity scores and better capture what constitutes the most relevant exemplars in our setting.

4.3 Experiment Results

As noted in Figure 4, a small percentage of anchors still produce narrative contexts that are similar to their original queries. In this case, using the original queries as the entire test set may introduce bias in evaluating ICP planning accuracy and payload accuracy. Therefore, to provide a more comprehensive assessment of TopoSem, we also report its out-of-distribution (OOD) planning performance. We collect an additional 100 queries from the ToolBench query collection and apply the same augmentation pipeline to generate corresponding (query, API graph) pairs. These augmented pairs are added to the vector store using the fine-tuned embedding model without further fine-tuning. The resulting OOD performance is reported later in this section.

Moreover, it is valuable to examine how planning performance varies with different values of K, the number of planning artifacts retrieved. A larger K may introduce dissimilar exemplars into the prompt, potentially misleading the LLM and resulting in incorrect planning. Based on this, we analyze how the clustering method described in Section 3.3 helps select exemplars for the ICP prompt presented to the LLM. Specifically, we investigate whether reducing noise and increasing exemplar diversity contribute to improved ICP performance in our setting.

We denote the models in the experiments using the following format: if zero-shot prompting is used with the planning LLM Claude 3.7, we refer to it as *Zero-shot Claude 3.7*; if exemplar selection is performed via semantic search and planning is done using GPT-4o, we denote it as *Semantic Search GPT-4o*. We summarize all testing questions in Table 1.

4.3.1 How does TopoSem perform on the original query sets? Table 2 presents ICP planning accuracy and payload accuracy across various base LLMs and exemplar selection methods. We evaluate hybrid search with the largest number of base LLMs, as it leverages both sparse vector methods like BM25 and the strengths of semantic

Table 1: Testing Questions

Test Question	Test set	Metrics
ICP	1000 original queries	Planning & Payload Acc.
ICP	100 OOD queries	Planning & Payload Acc.
Sensitivity Analysis of K	1000 original queries	Planning Acc.

search. Zero-shot and semantic-search-based ICP methods are used as baselines for comparison.

Table 2: TopoSem ICP performance on the original queries (All experiments were run 10 times with the temperature of the base LLMs set to 0.1 and $K = 10$)

Models	Planning Acc.	Payload Acc.
Zero-shot Claude 3.7	(0.67 ± 0.06)	(0.75 ± 0.11)
Semantic Search Claude 3.7	(0.58 ± 0.09)	(0.71 ± 0.08)
Hybrid Search Claude 3.5	(0.76 ± 0.02)	(0.82 ± 0.04)
Hybrid Search Claude 3.7	(0.75 ± 0.03)	(0.84 ± 0.06)
Hybrid Search GPT 4o	(0.78 ± 0.03)	(0.80 ± 0.02)
Action Similarity Claude 3.7	(0.81 ± 0.04)	(0.86 ± 0.03)
TopoSem Claude 3.5	(0.87 ± 0.02)	(0.85 ± 0.02)
TopoSem Claude 3.7	(0.87 ± 0.01)	(0.88 ± 0.04)
TopoSem GPT 4o	(0.85 ± 0.03)	(0.86 ± 0.02)

Table 2 reveals several key findings and insights. Notably, the results for Semantic Search with Claude 3.7 show that its performance is not consistently superior to zero-shot planning, where the LLM is provided only with the correct API description and plan in a zero-shot manner. This observation aligns with the findings of [33], which highlight that semantic search may retrieve exemplars with incorrect plans, potentially misleading LLM planning, particularly in domain-specific scenarios.

The hybrid search demonstrates better results compared to both pure semantic search and zero-shot methods. This improvement is attributed to the BM25 component, which helps filter out misleading exemplar candidates retrieved by semantic search through key token matching. However, as shown, hybrid search combined with different base LLMs yields similar planning and payload accuracy, indicating that given the same in-context plans, these LLMs perform comparably in terms of ICP planning and payload accuracy.

Both action similarity-based methods and TopoSem outperform traditional search methods by assessing similarity at the plan artifact level, rather than inferring plan similarity solely from query similarity. Moreover, our method surpasses action similarity-based approaches in planning accuracy while maintaining comparable performance in payload accuracy.

Here is an example (see A.5) that illustrates why our method outperforms traditional search approaches such as semantic search and hybrid search.

4.3.2 How does TopoSem perform on OOD query sets? Our method performs well not only because it considers similarity at both the topological and semantic levels of the payload, but also because the embedding model is fine-tuned on triplets constructed from the augmented dataset. However, analysis of the augmented queries reveals that the anchor used for pairwise similarity computation may sometimes share a similar narrative context. Therefore, evaluating performance on OOD data is also important.

Table 3: TopoSem ICP performance on OOD queries (All experiments were run 10 times with the temperature of the base LLMs set to 0.1 and $K = 10$)

Models	Planning Acc.	Payload Acc.
Hybrid Search Claude 3.7	(0.73 ± 0.05)	(0.79 ± 0.06)
Action Similarity Claude 3.7	(0.79 ± 0.03)	(0.83 ± 0.03)
TopoSem Claude 3.7	(0.77 ± 0.04)	(0.80 ± 0.02)

To further analyze performance on OOD queries, we manually investigated the sources of error. We found that the model often fails when faced with entirely new request patterns involving unseen API orchestrations. For example, queries related to shipment tracking, which were absent from both the original and augmented training data, were frequently planned incorrectly. In contrast, for queries involving familiar request patterns, the model was able to generate correct plans

Table 4: Error Matrix of OOD queries

Pattern	New Request Pattern	Old Request Pattern
Correct	25	52
Wrong	18	5

Although the model exhibits performance degradation on OOD data, it remains a powerful tool when combined with the (query, API graph) generation pipeline, such as in [22], to quickly adapt to new request patterns involving entirely new APIs or API orchestrations. Moreover, since the inference stage only requires using the fine-tuned embedding model to index augmented queries in a vector store, the system achieves efficient retrieval performance when integrated with tools like FAISS. This makes it well-suited for production deployment.

4.3.3 Sensitivty Analysis of K in TopoSem. As noted in prior work [25, 33], the number of exemplars can significantly impact ICP performance. Including too many exemplars may introduce noise and redundancy, and when up to 20 plans are provided, it may lead to the *lost-in-the-middle* issue [14].

To conduct the experiments, we evaluate planning accuracy using TopoSem with varying values of K, where K denotes the number of retrieved exemplars, and perform an ablation study to assess the impact of dynamic clustering on ICP.

From Figure 6, we observe that the planning accuracy of TopoSem without clustering is initially high because more exemplars are presented to the LLM, providing richer information for ICP. However, at this value of K, the number of clusters is often just one or two, meaning only a few exemplars are ultimately used, some of which may contain incorrect artifacts. As K increases, TopoSem with

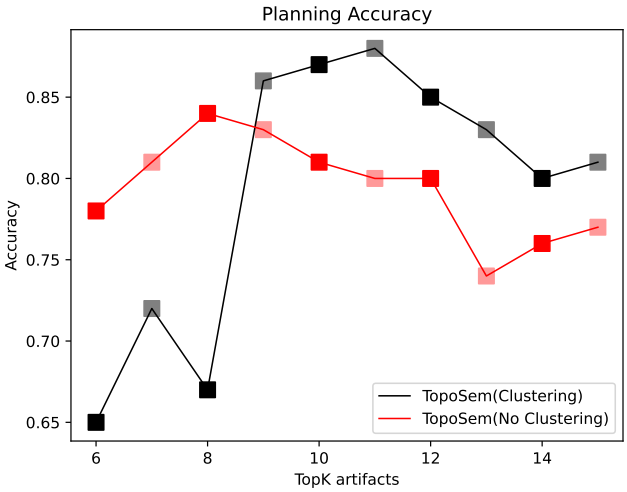

Figure 6: Planning Accuracy as K changes

clustering outperforms the no-clustering variant by reducing redundancy and increasing diversity. This prevents overwhelming the LLM with too many exemplars in the prompt, leading to better overall performance.

However, both methods show that increasing K indefinitely is not always beneficial, as larger K values can introduce noise—even with clustering—since not all clusters provide valid information for the LLM to perform correct planning. Thus, in production, selecting the optimal K requires hyperparameter tuning techniques.

4.4 Online Experiments

As mentioned in Section 4.1, we also evaluate our method on real-world seller-related APIs. We collect synthetic queries in the magnitude of hundreds along with their ground-truth plan artifacts. Note that each of these APIs can answer a handful of in-scope questions, and real-life seller queries may be more abstract than those in ToolBench.

We apply the same data augmentation pipeline and still uses the augmentaed (query, APIs graph) pair as the vector store and test on the origional queries. Using TopoSem with clustering, we obtain

Table 5: TopoSem ICP performance on real data with respect to Hybrid Search Claude 3.7. Note that only relative performance is reported for confidentiality reasons. (All experiments were run 10 times with the temperature of the base LLMs set to 0.1 and $K = 10$)

Models	Planning Acc.	Payload Acc.
TopoSem Claude 3.7	(+0.15 ± 0.02)	(+0.12 ± 0.01)

5 Conclusion, Limitation, and Future Work

The LLM Agentic framework has seen extensive adoption in industry, particularly within the e-commerce sector, due to its ability to significantly lower costs and enhance productivity through automation. Planning is central to the agentic framework, harnessing the LLM's ability to utilize tools to formulate an effective execution plan and ensure that responses provided to users are accurate and relevant. In the context of e-commerce, APIs tend to be more complex, their coordination more abstract, and the scope of related queries more domain-specific. Consequently, in-context planning (ICP) has become a standard approach to assist LLMs in navigating this complexity. However, relying solely on semantic similarity or hybrid search methods may fail to retrieve the appropriate planning artifacts, as different organizations or teams often employ distinct API orchestrations to address semantically similar queries.

In this paper, we propose TopoSem, an end-to-end framework designed to rapidly adapt pretrained embedding models to domain-specific queries for retrieving plan exemplars. TopoSem accounts for both the topological distance between API execution graphs and the semantic distance between payloads to the APIs within those graphs. Extensive experimental results demonstrate the effectiveness of TopoSem, highlighting the importance of integrating the topological structure of API execution graphs alongside the critical role of high-quality exemplars.

Despite its strengths, TopoSem has limitations. When confronted with a novel user request involving entirely new APIs or API orchestrations, it relies on a synthetic plan artifact generation pipeline to expand the vector store and improve performance. In future work, we plan to integrate this approach with tool knowledge graphs, such as those proposed in [6, 15], to enhance generalization to previously unseen APIs and orchestrations.

References

[1] Zeina Abu-Aisheh, Romain Raveaux, Jean-Yves Ramel, and Patrick Martineau. 2015. An Exact Graph Edit Distance Algorithm for Solving Pattern Recognition Problems. In *4th International Conference on Pattern Recognition Applications and Methods 2015*. Lisbon, Portugal. doi:10.5220/0005209202710278

[2] Bernd Bohnet, Azade Nova, Aaron T Parisi, Kevin Swersky, Katayoon Goshvadi, Hanjun Dai, Dale Schuurmans, Noah Fiedel, and Hanie Sedghi. 2024. Exploring and Benchmarking the Planning Capabilities of Large Language Models. arXiv:2406.13094 [cs.CL] https://arxiv.org/abs/2406.13094

[3] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models are Few-Shot Learners. arXiv:2005.14165 [cs.CL] https://arxiv.org/abs/2005.14165

[4] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. 2023. PaLM: Scaling Language Modeling with Pathways. *Journal of Machine Learning Research* 24, 240 (2023), 1–113. http://jmlr.org/papers/v24/22-1144.html

[5] DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li,

Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Shengfeng Ye, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanjia Zhao, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. 2025. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning. arXiv:2501.12948 [cs.CL] https://arxiv.org/abs/2501.12948

[6] Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, Dasha Metropolitansky, Robert Osazuwa Ness, and Jonathan Larson. 2025. From Local to Global: A Graph RAG Approach to Query-Focused Summarization. arXiv:2404.16130 [cs.CL] https://arxiv.org/abs/2404.16130

[7] Lutfi Eren Erdogan, Nicholas Lee, Siddharth Jha, Sehoon Kim, Ryan Tabrizi, Suhong Moon, Coleman Hooper, Gopala Anumanchipalli, Kurt Keutzer, and Amir Gholami. 2024. TinyAgent: Function Calling at the Edge. arXiv:2409.00608 [cs.CL] https://arxiv.org/abs/2409.00608

[8] Lutfi Eren Erdogan, Nicholas Lee, Sehoon Kim, Suhong Moon, Hiroki Furuta, Gopala Anumanchipalli, Kurt Keutzer, and Amir Gholami. 2025. Plan-and-Act: Improving Planning of Agents for Long-Horizon Tasks. arXiv:2503.09572 [cs.CL] https://arxiv.org/abs/2503.09572

[9] Jiawei Gu, Xuhui Jiang, Zhichao Shi, Hexiang Tan, Xuehao Zhai, Chengjin Xu, Wei Li, Yinghan Shen, Shengjie Ma, Honghao Liu, Saizhuo Wang, Kun Zhang, Yuanzhuo Wang, Wen Gao, Lionel Ni, and Jian Guo. 2025. A Survey on LLM-as-a-Judge. arXiv:2411.15594 [cs.CL] https://arxiv.org/abs/2411.15594

[10] Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2017. Billion-scale similarity search with GPUs. arXiv:1702.08734 [cs.CV] https://arxiv.org/abs/1702.08734

[11] Subbarao Kambhampati, Karthik Valmeekam, Lin Guan, Mudit Verma, Kaya Stechly, Siddhant Bhambri, Lucas Paul Saldyt, and Anil B Murthy. 2024. Position: LLMs can't plan, but can help planning in LLM-modulo frameworks. In *Forty-first International Conference on Machine Learning*.

[12] Sehoon Kim, Suhong Moon, Ryan Tabrizi, Nicholas Lee, Michael W. Mahoney, Kurt Keutzer, and Amir Gholami. 2024. An LLM Compiler for Parallel Function Calling. arXiv:2312.04511 [cs.CL] https://arxiv.org/abs/2312.04511

[13] Xianming Li and Jing Li. 2024. AnglE-optimized Text Embeddings. arXiv:2309.12871 [cs.CL] https://arxiv.org/abs/2309.12871

[14] Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2023. Lost in the Middle: How Language Models Use Long Contexts. arXiv:2307.03172 [cs.CL] https://arxiv.org/abs/2307.03172

[15] Xukun Liu, Zhiyuan Peng, Xiaoyuan Yi, Xing Xie, Lirong Xiang, Yuchen Liu, and Dongkuan Xu. 2024. ToolNet: Connecting Large Language Models with Massive Tools via Tool Graph. arXiv:2403.00839 [cs.AI] https://arxiv.org/abs/2403.00839

[16] Yanming Liu, Xinyue Peng, Jiannan Cao, Shi Bo, Yuwei Zhang, Xuhong Zhang, Sheng Cheng, Xun Wang, Jianwei Yin, and Tianyu Du. 2025. Tool-Planner: Task Planning with Clusters across Multiple Tools. arXiv:2406.03807 [cs.AI] https://arxiv.org/abs/2406.03807

[17] Chang Ma, Haiteng Zhao, Junlei Zhang, Junxian He, and Lingpeng Kong. 2024. Non-myopic Generation of Language Models for Reasoning and Planning. arXiv:2410.17195 [cs.AI] https://arxiv.org/abs/2410.17195

[18] Nicholas Monath, Kumar Avinava Dubey, Guru Guruganesh, Manzil Zaheer, Amr Ahmed, Andrew McCallum, Gokhan Mergen, Marc Najork, Mert Terzihan, Bryon Tjanaka, Yuan Wang, and Yuchen Wu. 2021. Scalable Hierarchical Agglomerative Clustering. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. ACM, 1245–1255. doi:10.1145/3447548.3467404

[19] OpenAI, :, Aaron Hurst, Adam Lerer, Adam P. Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, Aleksander Mądry, Alex Baker-Whitcomb, Alex Beutel, Alex Borzunov, Alex Carney, Alex Chow, Alex Kirillov, Alex Nichol, Alex Paino, Alex Renzin, Alex Tachard Passos, Alexander Kirillov, Alexi Christakis, Alexis Conneau, Ali Kamali, Allan Jabri, Allison Moyer, Allison Tam, Amadou Crookes, Amin Tootoochian, Amin Tootoonchian, Ananya Kumar, Andrea Vallone, Andrej Karpathy, Andrew Braunstein, Andrew Cann, Andrew Codispoti, Andrew Galu, Andrew Kondrich, Andrew Tulloch, Andrey Mishchenko, Angela Baek, Angela Jiang, Antoine Pelisse, Antonia Woodford, Anuj Gosalia, Arka Dhar, Ashley Pantuliano, Avi Nayak, Avital Oliver, Barret Zoph, Behrooz Ghorbani, Ben Leimberger, Ben Rossen, Ben Sokolowsky, Ben Wang, Benjamin Zweig, Beth Hoover, Blake Samic, Bob McGrew, Bobby Spero, Bogo Giertler, Bowen Cheng, Brad Lightcap, Brandon Walkin, Brendan Quinn, Brian Guarraci, Brian Hsu, Bright Kellogg, Brydon Eastman, Camillo Lugaresi, Carroll Wainwright, Cary Bassin, Cary Hudson, Casey Chu, Chad Nelson, Chak Li, Chan Jun Shern, Channing Conger, Charlotte Barette, Chelsea Voss, Chen Ding, Cheng Lu, Chong Zhang, Chris Beaumont, Chris Hallacy, Chris Koch, Christian Gibson, Christina Kim, Christine Choi, Christine McLeavey, Christopher Hesse, Claudia Fischer, Clemens Winter, Coley Czarnecki, Colin Jarvis, Colin Wei, Constantin Koumouzelis, Dane Sherburn, Daniel Kappler, Daniel Levin, Daniel Levy, David Carr, David Farhi, David Mely, David Robinson, David Sasaki, Denny Jin, Dev Valladares, Dimitris Tsipras, Doug Li, Duc Phong Nguyen, Duncan Findlay, Edede Oiwoh, Edmund Wong, Ehsan Asdar, Elizabeth Proehl, Elizabeth Yang, Eric Antonow, Eric Kramer, Eric Peterson, Eric Sigler, Eric Wallace, Eugene Brevdo, Evan Mays, Farzad Khorasani, Felipe Petroski Such, Filippo Raso, Francis Zhang, Fred von Lohmann, Freddie Sulit, Gabriel Goh, Gene Oden, Geoff Salmon, Giulio Starace, Greg Brockman, Hadi Salman, Haiming Bao, Haitang Hu, Hannah Wong, Haoyu Wang, Heather Schmidt, Heather Whitney, Heewoo Jun, Hendrik Kirchner, Henrique Ponde de Oliveira Pinto, Hongyu Ren, Huiwen Chang, Hyung Won Chung, Ian Kivlichan, Ian O'Connell, Ian O'Connell, Ian Osband, Ian Silber, Ian Sohl, Ibrahim Okuyucu, Ikai Lan, Ilya Kostrikov, Ilya Sutskever, Ingmar Kanitscheider, Ishaan Gulrajani, Jacob Coxon, Jacob Menick, Jakub Pachocki, James Aung, James Betker, James Crooks, James Lennon, Jamie Kiros, Jan Leike, Jane Park, Jason Kwon, Jason Phang, Jason Teplitz, Jason Wei, Jason Wolfe, Jay Chen, Jeff Harris, Jenia Varavva, Jessica Gan Lee, Jessica Shieh, Ji Lin, Jiahui Yu, Jiayi Weng, Jie Tang, Jieqi Yu, Joanne Jang, Joaquin Quinonero Candela, Joe Beutler, Joe Landers, Joel Parish, Johannes Heidecke, John Schulman, Jonathan Lachman, Jonathan McKay, Jonathan Uesato, Jonathan Ward, Jong Wook Kim, Joost Huizinga, Jordan Sitkin, Jos Kraaijeveld, Josh Gross, Josh Kaplan, Josh Snyder, Joshua Achiam, Joy Jiao, Joyce Lee, Juntang Zhuang, Justyn Harriman, Kai Fricke, Kai Hayashi, Karan Singhal, Katy Shi, Kavin Karthik, Kayla Wood, Kendra Rimbach, Kenny Hsu, Kenny Nguyen, Keren Gu-Lemberg, Kevin Button, Kevin Liu, Kiel Howe, Krithika Muthukumar, Kyle Luther, Lama Ahmad, Larry Kai, Lauren Itow, Lauren Workman, Leher Pathak, Leo Chen, Li Jing, Lia Guy, Liam Fedus, Liang Zhou, Lien Mamitsuka, Lilian Weng, Lindsay McCallum, Lindsey Held, Long Ouyang, Louis Feuvrier, Lu Zhang, Lukas Kondraciuk, Lukasz Kaiser, Luke Hewitt, Luke Metz, Lyric Doshi, Mada Aflak, Maddie Simens, Madelaine Boyd, Madeleine Thompson, Marat Dukhan, Mark Chen, Mark Gray, Mark Hudnall, Marvin Zhang, Marwan Aljubeh, Mateusz Litwin, Matthew Zeng, Max Johnson, Maya Shetty, Mayank Gupta, Meghan Shah, Mehmet Yatbaz, Meng Jia Yang, Mengchao Zhong, Mia Glaese, Mianna Chen, Michael Janner, Michael Lampe, Michael Petrov, Michael Wu, Michele Wang, Michelle Fradin, Michelle Pokrass, Miguel Castro, Miguel Oom Temudo de Castro, Mikhail Pavlov, Miles Brundage, Miles Wang, Minal Khan, Mira Murati, Mo Bavarian, Molly Lin, Murat Yesildal, Nacho Soto, Natalia Gimelshein, Natalie Cone, Natalie Staudacher, Natalie Summers, Natan LaFontaine, Neil Chowdhury, Nick Ryder, Nick Stathas, Nick Turley, Nik Tezak, Niko Felix, Nithanth Kudige, Nitish Keskar, Noah Deutsch, Noel Bundick, Nora Puckett, Ofir Nachum, Ola Okelola, Oleg Boiko, Oleg Murk, Oliver Jaffe, Olivia Watkins, Olivier Godement, Owen Campbell-Moore, Patrick Chao, Paul McMillan, Pavel Belov, Peng Su, Peter Bak, Peter Bakkum, Peter Deng, Peter Dolan, Peter Hoeschele, Peter Welinder, Phil Tillet, Philip Pronin, Philippe Tillet, Prafulla Dhariwal, Qiming Yuan, Rachel Dias, Rachel Lim, Rahul Arora, Rajan Troll, Randall Lin, Rapha Gontijo Lopes, Raul Puri, Reah Miyara, Reimar Leike, Renaud Gaubert, Reza Zamani, Ricky Wang, Rob Donnelly, Rob Honsby, Rocky Smith, Rohan Sahai, Rohit Ramchandani, Romain Huet, Rory Carmichael, Rowan Zellers, Roy Chen, Ruby Chen, Ruslan Nigmatullin, Ryan Cheu, Saachi Jain, Sam Altman, Sam Schoenholz, Sam Toizer, Samuel Miserendino, Sandhini Agarwal, Sara Culver, Scott Ethersmith, Scott Gray, Sean Grove, Sean Metzger, Shamez Hermani, Shantanu Jain, Shengjia Zhao, Sherwin Wu, Shino Jomoto, Shirong Wu, Shuaiqi, Xia, Sonia Phene, Spencer Papay, Srinivas Narayanan, Steve Coffey, Steve Lee, Stewart Hall, Suchir Balaji, Tal Broda, Tal Stramer, Tao Xu, Tarun Gogineni, Taya Christianson, Ted Sanders, Tejal Patwardhan, Thomas Cunninghman, Thomas Degry, Thomas Dimson, Thomas Raoux, Thomas Shadwell, Tianhao Zheng, Todd Underwood, Todor Markov, Toki Sherbakov, Tom Rubin, Tom Stasi, Tomer Kaftan, Tristan Heywood, Troy Peterson, Tyce Walters, Tyna Eloundou, Valerie Qi, Veit Moeller, Vinnie Monaco, Vishal Kuo, Vlad Fomenko, Wayne Chang, Weiyi Zheng, Wenda Zhou, Wesam Manassra, Will Sheu, Wojciech Zaremba, Yash Patil, Yilei Qian, Yongjik Kim, Youlong Cheng, Yu Zhang, Yuchen He, Yuchen Zhang, Yujia Jin, Yunxing Dai, and Yury Malkov. 2024. GPT-4o System Card. arXiv:2410.21276 [cs.CL] https://arxiv.org/abs/2410.21276

[20] 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. arXiv:2307.16789 [cs.AI] https://arxiv.org/abs/2307.16789

[21] Stephen Robertson and Hugo Zaragoza. 2009. The Probabilistic Relevance Framework: BM25 and Beyond. *Found. Trends Inf. Retr.* 3, 4 (April 2009), 333–389. doi:10.1561/1500000019

[22] Ying Sheng, Sudeep Gandhe, Bhargav Kanagal, Nick Edmonds, Zachary Fisher, Sandeep Tata, and Aarush Selvan. 2024. Measuring an LLM's Proficiency at using APIs: A Query Generation Strategy. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining* (Barcelona, Spain) *(KDD '24)*. Association for Computing Machinery, New York, NY, USA, 5680–5689. doi:10.1145/3637528.3671592

[23] Saba Sturua, Isabelle Mohr, Mohammad Kalim Akram, Michael Günther, Bo Wang, Markus Krimmel, Feng Wang, Georgios Mastrapas, Andreas Koukounas, Andreas Koukounas, Nan Wang, and Han Xiao. 2024. jina-embeddings-v3: Multilingual Embeddings With Task LoRA. arXiv:2409.10173 [cs.CL] https://arxiv.org/abs/2409.10173

[24] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open Foundation and Fine-Tuned Chat Models. arXiv:2307.09288 [cs.CL] https://arxiv.org/abs/2307.09288

[25] Karthik Valmeekam, Matthew Marquez, Alberto Olmo, Sarath Sreedharan, and Subbarao Kambhampati. 2023. PlanBench: An Extensible Benchmark for Evaluating Large Language Models on Planning and Reasoning about Change. arXiv:2206.10498 [cs.CL] https://arxiv.org/abs/2206.10498

[26] Xingyao Wang, Yangyi Chen, Lifan Yuan, Yizhe Zhang, Yunzhu Li, Hao Peng, and Heng Ji. 2024. Executable Code Actions Elicit Better LLM Agents. arXiv:2402.01030 [cs.CL] https://arxiv.org/abs/2402.01030

[27] Jiaxin Wen, Jian Guan, Hongning Wang, Wei Wu, and Minlie Huang. 2024. Unlocking Reasoning Potential in Large Langauge Models by Scaling Code-form Planning. arXiv:2409.12452 [cs.CL] https://arxiv.org/abs/2409.12452

[28] Binfeng Xu, Zhiyuan Peng, Bowen Lei, Subhabrata Mukherjee, Yuchen Liu, and Dongkuan Xu. 2023. ReWOO: Decoupling Reasoning from Observations for Efficient Augmented Language Models. arXiv:2305.18323 [cs.CL] https://arxiv.org/abs/2305.18323

[29] Ling Yang, Zhaochen Yu, Bin Cui, and Mengdi Wang. 2025. ReasonFlux: Hierarchical LLM Reasoning via Scaling Thought Templates. arXiv:2502.06772 [cs.CL] https://arxiv.org/abs/2502.06772

[30] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of Thoughts: Deliberate Problem Solving with Large Language Models. arXiv:2305.10601 [cs.CL] https://arxiv.org/abs/2305.10601

[31] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. ReAct: Synergizing Reasoning and Acting in Language Models. arXiv:2210.03629 [cs.CL] https://arxiv.org/abs/2210.03629

[32] Aohan Zeng, Xiao Liu, Zhengxiao Du, Zihan Wang, Hanyu Lai, Ming Ding, Zhuoyi Yang, Yifan Xu, Wendi Zheng, Xiao Xia, Weng Lam Tam, Zixuan Ma, Yufei Xue, Jidong Zhai, Wenguang Chen, Peng Zhang, Yuxiao Dong, and Jie Tang. 2023. GLM-130B: An Open Bilingual Pre-trained Model. arXiv:2210.02414 [cs.CL] https://arxiv.org/abs/2210.02414

[33] Xinran Zhao, Hanie Sedghi, Bernd Bohnet, Dale Schuurmans, and Azade Nova. 2025. Improving Large Language Model Planning with Action Sequence Similarity. arXiv:2505.01009 [cs.AI] https://arxiv.org/abs/2505.01009

A Prompts in Data Augmentation Pipeline

Given the original query along with its ground-truth plan artifacts,

A.1 Scenario 1

Prompt for scenario 1

You are given a query which a customer ask to an assistant, and you will be also given a list of tasks about how the assistant execute the underlying executors and its executor tools to ask the query.
You task must generate a different story context, different words and style, different order of sentences, and also the verb (for example, if the original query may talks about trip, then the modified query could talks about go to hospital), and also you could change the date or the location, but the executors, executors tools, and the task dependencies need to keep as the same with only changing the time or location for the query since you could answer the modified query with only changing the time, date or amount or location in the payload.
Here is the {query} and here are the {tasks}.
Here is the modified query:
Here is the modified tasks:
Return me the result without saying anything else.

A.2 Scenario 2

Prompt for scenario 2

You are given a query which a customer ask to an assistant, and you will be also given a list of tasks about how the assistant execute the underlying executors and its executor tools to answer the query.
You task is to delete one task from the task lists and the most important thing is that this task should not be a dependent task of another task.
After that deletion, you need to also remove the part in the query where contains this task, and after removing the part, you need to use the different words and styles to rewrite the rest of the query, but the query for each left task after task deletion should be the same with only changing the time, location or date or amount or cost.
The task lists after deletion should answer the modified query after deletion exactly.
Here is the {query} and here are the {tasks}.
Here is the modified query:
Here is the modified tasks:
Return me the result without saying anything else.

A.3 Scenario 3

Prompt for scenario 3

You are given a query which a customer ask to an assistant, and you will be also given a list of tasks about how the assistant execute the underlying executors and its executor tools to answer the query.
You need to look the query and change some parts of the query and also change the task list based your change in the customer query. For example, if part of customer query asks 'I am planning on a trip to New York. I am looking for the public transit sector. Could you tell me their location?' Then you could tweak this query to 'I am thinking about traveling to Tokyo. I want to take the public transportation for looking the cityview. Could you tell me their ticket price?'. Therefore, correspondingly, in the task lists, you need to corresponding change that task by changing the executor tool to ticketpricing, and also changes the query of this task to get the ticket price of the Tokyo public transportation. Meanwhile, keep other parts of the query and also the tasks in the task list unchanged by only rewriting them using different words or style but should be the same meaning,
Here is the {query} and here are the {tasks}.
Think step by step and give me the answers as follow:
Here is the modified query:
Here is the modified tasks:
Return me the result without saying anything else.

A.4 Scenario 4

Prompt for scenario 4

You are given a query which a customer ask to an assistant, and you will be also given a list of tasks about how the assistant execute the underlying executors and its executor tools to answer the query.
You need to come up with a total new query that completely different with the original and totally new queries for each task in the list. Note that you need to keep all tasks unchanged except for the query. Therefore, you need to open the imgination to think about a total irrelevant new query with roughly same length for each task [if the old query is asking about the address, then the new query may asks about dinner] and this new query should be finished by the executor and its executor tools. you need to think about this new query by the name of the executor and executor tools. After putting each new query in the task list, you need to think about a case where you could use these tasks in the task list to answer.
Here is the query and here is the tasks tasks.
Think step by step and give me the answers as follow:
Here is the modified query:
Here is the modified tasks:
Return me the result without saying anything else.

A.5 Offline running example

Offline running example

Query: = I need to prepare a trip to China, I need to rent an Airbnb and book reservation tickets for the visit.
TopoSem: I'm planning a surprise trip and need to find a cozy cabin in the mountains. Can you help me locate a cabin rental and assist with the booking?
Semantic Search: I'm planning a trip to China and I want to stay informed with the latest news articles in Chinese. Could you help me with that?
Hybrid Search: I'm planning a weekend getaway to Tokyo with my partner and I need some information to make the trip more enjoyable. Can you suggest some popular dining spots in Tokyo along with their reviews? Additionally, I would like to know the current weather forecast for Tokyo and find a cozy Airbnb in the Shibuya district for our stay.

Received 20 February 2007; revised 12 March 2009; accepted 5 June 2009