Concept based Ambiguity Resolution in LLMs

Zhibo Hu[1, 2], Chen Wang[1, 2], Yanfeng Shu[2], Hye-Young Paik[1], Liming Zhu[1, 2]
[1]The University of New South Wales, [2]CSIRO Data61
zhibo.hu@student.unsw.edu.au, chen.wang@data61.csiro.au,
yanfeng.shu@data61.csiro.au, h.paik@unsw.edu.au, liming.zhu@data61.csiro.au

Ambiguity in natural language is a significant obstacle for achieving accurate text to structured data mapping through large language models (LLMs), which affects the performance of tasks such as mapping text to agentic tool calling and text-to-SQL queries. Existing methods to ambiguity handling either rely on the ReACT framework to obtain correct mappings through trial and error, or on supervised fine-tuning to bias models toward specific tasks. In this paper, we adopt a different approach that characterizes representation differences of ambiguous text in the latent space and leverages these differences to identify ambiguity before mapping them to structured data. To detect sentence-level ambiguity, we focus on the relationship between ambiguous questions and their interpretations. Unlike distances calculated by dense embeddings, we introduce a new distance measure based on a path kernel over concepts. With this measurement, we identify patterns to distinguish ambiguous from unambiguous questions. Furthermore, we propose a method for improving LLM performance on ambiguous agentic tool calling through missing concept prediction. Both achieve state-of-the-art results.

1. Introduction

Question answering using large language models (LLMs) often fails when user questions are ambiguous. A growing strand of work like [1, 2] shows that a surprising fraction of their "errors" can be traced back, not due to lack of knowledge in LLM, but to ambiguity in the user's question itself. This ambiguity does not just mean that the question does not provide enough information, but also that the question has ambiguous semantics, i.e., multiple interpretations.

Existing studies focus more on pragmatic or lexical ambiguity, ambiguity handling in these studies either exploits the ReACT[3] framework to produce correct mappings through trial and error, or supervised fine tuning to guide models to produce biased mappings to improve on certain tasks[4]. Kamath et al. [5] attempt to use LLMs to detect ambiguity of sentences whose meaning changes with the relative scope of quantifiers, negation, or modals. They show that powerful LLMs trained on the most comprehensive datasets, such as GPT-4 sometimes default to a non-preferred semantic reading, and that success of disambiguating text varies sharply with different phrasing, which indicates disambiguation can not be easily solved by LLMs themselves. The ambiguity detection results in [6] also confirm this observation. On the other hand, there is limited research on representational differences of ambiguous text. In this work, we study the representation of ambiguous text in the latent space and leverage the differences to identify ambiguity.

As an ambiguous utterance has multiple interpretations, studying the distribution of interpretations is a natural way for ambiguity detection [7]. Figure 1 provides an example of the relationships between the ambiguous query q and its corresponding two interpretations, denoted by i_1 and i_2. Ideally, a good distance measurement may uncover the pattern of the triplet associated with an ambiguous utterance. Unfortunately, current distance measurement by dense embedding vectors[8] cannot give us such a measurement. The distances computed by dense vectors focus more on the semantics of individual words than on the structure of the entire sentence, which is not sensitive to the ambiguity caused by the structure of the sentence, particularly when some concepts are missing in the sentence.

Third Conference on Parsimony and Learning (CPAL 2026).

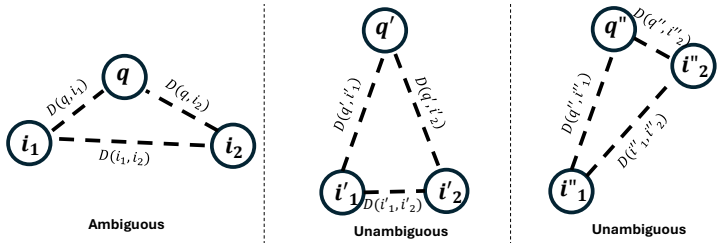


Figure 1: An example to show the difference of the distance measurement on triplets for ambiguous question $(q,\, i_1,\, i_2)$ and two kind of unambiguous question $(q',\, i'_1,\, i'_2)$, $(q'',\, i''_1,\, i''_2)$. For similar queries ambiguous q and unambiguous q', q'', we expect $\overline{(\mathrm{D}(q,i_1), \mathrm{D}(q,i_2), \mathrm{D}(i_1,i_2))} > \overline{(\mathrm{D}(q',i'_1), \mathrm{D}(q',i'_2), \mathrm{D}(i'_1,i'_2))}$ and $|\mathrm{D}(q,i_1) - \mathrm{D}(q,i_2)| \ll |\mathrm{D}(q'',i''_1) - \mathrm{D}(q'',i''_2)|$ (the overline means average).

In our study, we observe that the ambiguity is often associated with missing concepts in the input utterances. With the recent progress on LLM interpretability [9, 10], the human-understandable concepts embodied in a utterance can be extracted together with their representation in the latent space through a sparse autoencoder (SAE). This inspires us to design methods to learn the concept differences of multiple interpretations of the input utterance in the latent space to identify ambiguity. We further leverage a kernel method [11] to develop a distance metric for such latent concept comparison. We turn SAE into a kernel machine to measure the similarity between the concept representations of different interpretations of an ambiguous utterance. The computing is done through the integral of gradient values in a path kernel for each concept extracted by SAE. To make the similarity measurement focus on the target semantic patterns, we filter out concepts irrelevant to input utterances. By doing this, we successfully discover the pattern of ambiguous questions.

Once ambiguous utterances are identified, incorporating additional information can improve their mapping to structured data. When the structured data requires intermediate such as SQL generation to access, users often need to be asked to clarify the utterances and provide additional information. When the structured data are finite and well-defined, e.g., tools defined within an agentic framework, this additional information can be obtained from the concepts embodied in the data. We exploit the difference of concepts between ambiguous queries and target structured data, and design a missing concept prediction model to assist the mapping. We show in the experiments that our method achieves the best API calling performance on Gorilla[12] TensorFlow Hub bench.

In summary, our work make the following contributions:

1. We observed that ambiguity arises from *missing concepts* in the latent space of LLMs (Section 3.1 and 3.2). Using this insight, we designed a new distance measure that enhances interpretability and targets specific semantic patterns.
2. We identify patterns to distinguish ambiguous from unambiguous questions with this measurement.
3. We propose a new framework to enhance the performance of LLMs in handling ambiguous agentic tool calls by predicting missing concepts.

2. Preliminary

Path Kernel. Path kernels are used to measure how similarly a model at two data points varies during learning. Here we refer to the explanation for kernel machine from [11], a *kernel machine* predicts

$$y = g\Big(\sum_i a_i\, K(x, x_i) + b\Big),$$

with the kernel K measuring the similarity between data points. Gradient-descent training (learning rate $\varepsilon \to 0$) implies that the final predictor behaves like a **path kernel** machine:

$$\mathrm{K}_{\text{path}}(x, x') \;=\; \int_{c(t)} \nabla_w y(x) \cdot \nabla_w y(x')\, dt,$$

where $c(t)$ is the parameter trajectory during training. The more aligned the gradients of y at x and x', the larger the kernel value, thus the variations of x and x' are more similar during training.

Sparse autoencoder (SAE). Neurons in modern language models often behave such that the same unit fires for several unrelated concepts. A leading hypothesis (Superposition Hypothesis) is [13]: the model stores many more features than it has neurons by packing them into an over-complete set of directions in activation space. Recovering those directions is therefore a natural route to mechanistic interpretability. The work by Anthropic shows that a *sparse auto-encoder* (SAE) trained directly on a layer's activations can do exactly this, yielding thousands of highly interpretable, near-monosemantic concepts[9].

Let $\mathbf{H}(x) \in \mathbb{R}^d$ denote the hidden-state (e.g. residual-stream) vector produced by an LLM for a token sequence x. The goal is to learn a *dictionary* $\{\mathbf{d}_i\}_{i=1}^n \subset \mathbb{R}^d$ such that every activation can be reconstructed from a **sparse** combination of these directions:

$$\mathbf{H}(x) \;\approx\; \mathbf{b} \;+\; \sum_{i=1}^{n} f_i\big(\mathbf{H}(x)\big)\, \mathbf{d}_i \tag{1}$$

in which, $\mathbf{b} \in \mathbb{R}^d$ is a learned bias that captures the mean activation. $f_i(\mathbf{H})$ is a *gate* that decides whether feature i is present; the ReLU promotes non-negativity and sparsity: $f_i(\mathbf{H}) = \mathrm{ReLU}\big(\langle \mathbf{w}_i, \mathbf{H} \rangle + b_i^{\text{enc}}\big)$; $\mathbf{d}_i$ is the **decoder** vector that take the feature back into the original space.

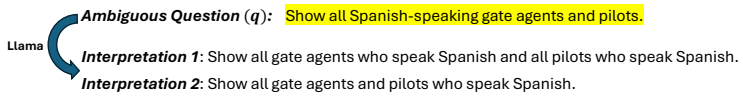
Ambiguity in NLP. Ambiguity has been studied across various NLP tasks including machine translation [14], natural language inference [15], question answering [16, 17], and semantic parsing [4, 18]. Recent approaches leverage LLMs to detect ambiguities by sampling multiple candidate solutions and resolving ambiguities through clarification questions or by prompting alternative interpretations. For instance, Mu et al. [18] samples multiple outputs from an LLM and examines their consistency to identify potential ambiguities. When inconsistencies are detected, the LLM is prompted to generate targeted clarification questions. However, due to the inherent biases of LLMs, the sampled solutions may lack diversity, making some ambiguities difficult to detect. To address this limitation, Saparina and Lapata [4] generates an initial set of default interpretations using an LLM, which are then augmented using a specialized infilling model that requires supervised training. Our work instead examines ambiguity in the latent concept space.

3. Methodology

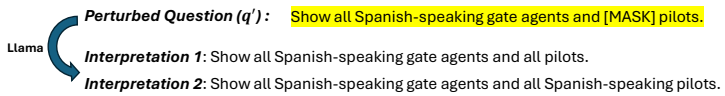
In this section, we first define the ambiguity resolution problem as a missing concept problem (3.1). We then show the effect of adding missing concepts(3.2), followed by describing our ambiguity detection method (3.3). Finally we describe how we predict missing concepts in the context of agentic tool calling (3.4).

3.1. Ambiguity Resolution as a Missing Concept Problem

LLMs often bias towards generating one interpretation among many for an ambiguous utterance [4]. Prompting LLMs to produce multiple interpretations and directly comparing their semantics do not help ambiguity detection. To show LLMs do not produce different interpretations for an ambiguous utterance, we extract a sentence from the AMBROSIA dataset [6] and prompt Llama-3.3-70B-Instruct[19] to generate two interpretations for this sentence.



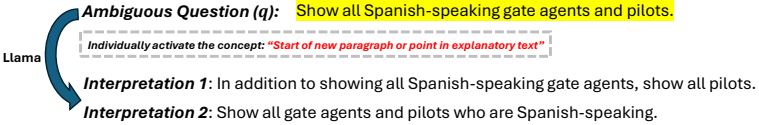
These two interpretations in fact have the same meaning. To trigger generating the diverse interpretations, we exploit special tokens' role in steering LLMs' responses. When we insert "[MASK]" in the original sentence, the Llama model produces two interpretations aligned with the ground-truth:



Note that while "[MASK]" has no semantic meaning by itself, its presence in this position increases the sentence's uncertainty. To understand what changes are triggered by the "[MASK]" token in the concept space that make the model produce different interpretations. We use a sparse-autoencoder (SAE)[20] trained on the outputs of 50 layer of this Llama model to track the new concepts after the "[MASK]" token is inserted. We get the following key concept from the SAE:

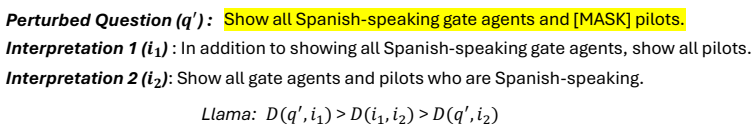
"Start of new paragraph or point in explanatory text".

To verify the new interpretation is indeed triggered by this concept, we individually clamp the activation value of this concept (increase it to 10) while keeping the original input sentence unchanged (without inserting the "[MASK]" token). We obtain the following interpretations:



These interpretations match the ground-truth, which indicates an LLM can be steered to generate diverse interpretations for ambiguous utterances. We further show in the experiments that injecting examples in the prompt can effectively "remind" the model the missing concepts, therefore trigger the generation of diverse interpretations (see Appendix B.1 for examples). With the diverse interpretations of ambiguous inputs, we can then detect such ambiguity.

However, a naive approach of using the distances of dense vectors of generated interpretations to detect ambiguity does not work well. We use the following example to elaborate this. We denote the top two interpretations of the input utterance q as i_1 and i_2. By using the output of the last hidden layer of Llama-3.3-70B-Instruct for the three sentences separately, we obtain the dense vectors of the triplets $(v(q), v(i_1), v(i_2))$:



Although the "[MASK]" token activates additional concepts of the Llama model and makes it generate diverse interpretations, these additional concepts do not lead to sufficient changes in the dense vectors for ambiguity detection. The distance between $v(q')$ and $v(i_1)$ is 0.17 and the distance between $v(q')$ and $v(i_2)$ is 0.092, while the distance between $v(i_1)$ and $v(i_2)$ is 0.13. It is difficult to leverage the distance contrast in the triplet to derive a threshold to classify q as ambiguous as the distance between interpretations can be arbitrarily smaller. We have done experiments with advanced embedding models and they do not have satisfactory sensitivity for distinguishing ambiguity patterns either.

However, we notice that q' and i_1 activated some concepts in common, which inspires us to utilize the concept differences of the triplet in the latent space to detect pattern of ambiguity. We show that such distance measure can produce sufficient sensitivity for ambiguity detection. The distances measured using our method are as follows: $\mathrm{D}(q', i_1) = 0.039$; $\mathrm{D}(q', i_2) = 0.027$; $\mathrm{D}(i_1, i_2) = 0.043$, meaning the distance between interpretations is larger than their distances to the query. This property produces a sensitive metric for ambiguity detection. We explain the proposed distance metric in Section 3.3.

3.2. Effect of adding missing concepts into the latent space

To further explore the relationship between semantic ambiguity (uncertainty in LLMs) and missing concepts, we use semantic entropy[21] to measure the ambiguity of query semantics (see Appendix A for algorithm details). We compute the semantic entropy produced by LLama-3.3-70B-Instruct [19] on 1) 20 ambiguous questions; 2) the same 20 questions with random concepts activated; and 3) the same 20 questions with missing concepts activated. Figure 2 shows that without any additional concept activated, the semantic entropy is close to zero, indicating the LLM produces a single interpretation on the input. This explains the semantic entropy of queries produced by Llama alone cannot detect ambiguity. With the missing concepts activated, the interpretations become diverse, evidenced by the increase of semantic entropy of queries. Activating random concepts also increases the semantic entropy because the noise introduced leads to diverse semantics, but the increase is less significant than that caused by adding target missing concepts.

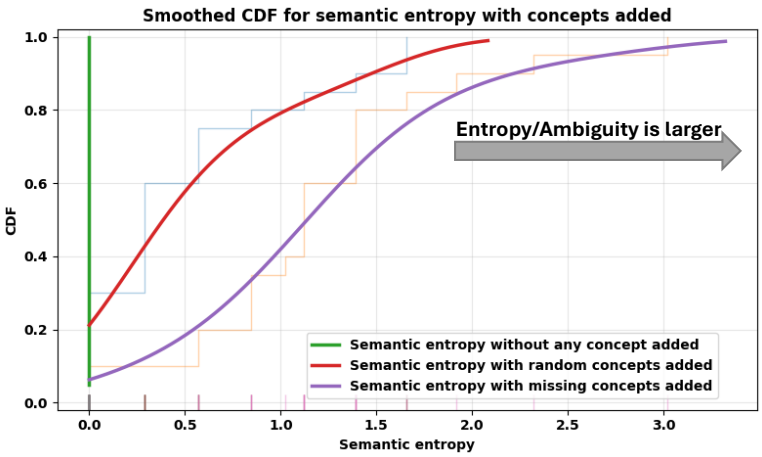


Figure 2: Entropy/ambiguity change with missing concepts added.

This explains the effect of background knowledge on semantic ambiguity. For example, for this question: "Who won the war between ethiopia and italy?", if the LLM lacks the context of Italo-Ethiopian War, it does not know where the ambiguity is. Once the LLM retrieves the context of Italo-Ethiopian War from external sources, the concept space is enriched with the "First War" or the "Second War", which in turn increases the semantic entropy of the question.

This also explains why fine-tuning works on ambiguity resolution. Fine-tuning can be seen as learning to activate the missing concepts and therefore increase the semantic entropy of ambiguous queries.

3.3. Representation-based Ambiguity Detection

Our solution (see Figure 3) is to use the path kernel with a sparse autoencoder (SAE) as the kernel machine for calculating distances between data points.

SAE as Kernel Machine. We consider SAE as y, the input sentences are x and x', their hidden states on LLM's layer where the SAE trained on are $H(x)$ and $H(x')$. Therefore, we have

$$\mathrm{K}(x, x') = \int_{c(t)} \left(\nabla_{\mathbf{w}} \mathrm{SAE}(H(x))\right) \cdot \left(\nabla_{\mathbf{w}} \mathrm{SAE}(H(x'))\right) dt \tag{2}$$

Here, we denote the SAE on the given concept dictionary as $\mathbf{f}_{\mathrm{SAE}}$, where $\mathbf{f}_{\mathrm{SAE}}(\mathbf{H}(\mathbf{x})) = (f_1(\mathbf{H}(\mathbf{x})), \ldots, f_N(\mathbf{H}(\mathbf{x})))^\top$, ($f(\mathbf{H})$ in Equation 1). The i-th activation is then simply $f_i(\mathbf{x})$.

$$\mathbf{f}_{\text{SAE}}(\mathbf{H}(\mathbf{x})) \;=\; \text{ReLU}\big(W_e(\mathbf{H}(\mathbf{x}) - \mathbf{b}_d) + \mathbf{b}_e\big) \;\in\; \mathbb{R}^N, \tag{3}$$

where W_e is the weight matrix of the encoder and $\mathbf{b}_d, \mathbf{b}_e$ are a pre-encoder and an encoder bias, respectively.

Not all N concepts are necessary for the path kernel calculation. To obtain the variation for input sentences x and x', we only need to focus on the target concepts. Accordingly, we apply a mask M on the features used in gradient computation when calculating the path kernel:

$$\text{K}(x, x') = \int_{c(t)} \nabla_{\mathbf{w}} \mathbf{f}_{\text{SAE}}^{\text{mask}}(\mathbf{H}(x)) \;\cdot\; \nabla_{\mathbf{w}} \mathbf{f}_{\text{SAE}}^{\text{mask}}(\mathbf{H}(x'))\, dt, \tag{4}$$

$$\mathbf{f}_{\text{SAE}}^{\text{mask}}(\mathbf{H}(x)) = M \circ \Big[\text{ReLU}\big(W_e\,(\mathbf{H}(x) - \mathbf{b}_d) + \mathbf{b}_e\big)\Big] \tag{5}$$

where M is the concept mask (explained below) and $\circ$ is hadamard product.

Determining Unmasked/Target Concepts.

In interpretation generation, we use concept embodied examples (see Appendix B.1) for triggering the generation of diverse interpretations. To ensure our distance calculation captures the semantic meaning of sentences, we distill the concepts activated by their semantics and restrict the path kernel computation to these concepts. The distillation process involves three steps:

1. Collect the concepts activated by the example triplet sentences by LLM with SAE.
2. Remove the concepts activated by each individual token t_i in the example triplet sentences from the set of concepts recorded in step 1.
3. Include the remaining concepts in the mask vector $\boldsymbol{M}$, which are considered valid:

$$\boldsymbol{M} = \{\mathbf{f}(\mathbf{H}(\mathbf{x}))\} \setminus \{\mathbf{f}(\mathbf{H}(\mathbf{t_1}), \ldots, \mathbf{H}(\mathbf{t_n})\} \tag{6}$$

Here x is the example sentence and $t_1, \ldots, t_n$ are the tokens in the sentence.

Path State Approximation. We use a path kernel to characterize relationship of the obtained latent representations of concepts. Path states are the snapshots of a model's parameters saved after each optimization step during training or fine-tuning. For a pre-trained SAE we usually only have the final weights, so the original series of path states cannot be reconstructed exactly. When re-training is impossible or costly, we can replace the unknown gradient-descent path with a straight-line interpolation in parameter space. Let

$$\Theta = \{\theta_k\}_{k=1}^P, \Theta^{(0)} = \{\theta_k^{(0)}\}_{k=1}^P, \Theta^* = \{\theta_k^*\}_{k=1}^P$$

be, respectively, the parameter set, the (zero) initialization, and the final pre-trained weights.

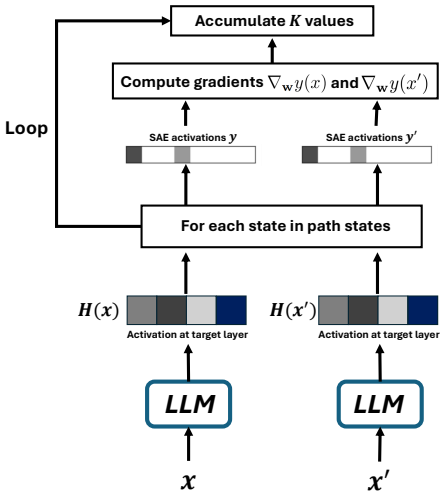


Figure 3: The workflow of the path kernel calculation with SAE.

By choosing n interpolation steps and define $\alpha_j = \frac{j}{n-1}, \quad j = 0, 1, \ldots, n-1.$, the j-th intermediate snapshot is then

$$\Theta^{(j)} = (1 - \alpha_j)\,\Theta^{(0)} + \alpha_j\,\Theta^*, \quad \theta_k^{(j)} = (1 - \alpha_j)\,\theta_k^{(0)} + \alpha_j\,\theta_k^*$$

Collecting them gives the full set of path states: $\{\Theta^{(0)}, \Theta^{(1)}, \ldots, \Theta^{(n-1)}\}$.

Here, α increases linearly from 0 to 1, forming a straight-line path. It provides a simple, deterministic path that is often adequate for estimating a path kernel.

Distance Measurement. The path kernel measures how similar two data points are according to the model based on their changing trajectories along the paths. To convert the (unnormalized) path kernel $K(\cdot, \cdot)$ into a proper distance metric between data points x and x', we apply the following two standard normalizations:

$$\mathrm{D}_1(x, x') = 1 - \frac{\mathrm{K}(x, x')}{\sqrt{\mathrm{K}(x, x)\,\mathrm{K}(x', x')}}, \tag{7}$$

$$\mathrm{D}_2(x, x') = \sqrt{\mathrm{K}(x, x) + \mathrm{K}(x', x') - 2\,\mathrm{K}(x, x')}, \tag{8}$$

We show in the experiments that both D_1 and D_2 can identify ambiguity and can be used for serving different objectives.

3.4. Predicting Missing Concepts to Mitigate Ambiguity

In Section 3.1, we argue that the ambiguity problem arises from concepts missing in LLM's latent space, and thus distance measurements should be sensitive to this. Based on this hypothesis, we propose using path kernels with SAE to measure distances between questions and their interpretations. As shown in our experiments, this method reveals patterns that distinguish ambiguous questions from unambiguous ones. Motivated by this, we investigate whether ambiguity can be exploited to reduce incorrect responses and better align outputs with training data. To this end, we introduce a framework - within the context of tool calling - that retrieves data chunks by training a concept predictor on labeled data. As illustrated in Figure 4, instead of using dense embedding vectors to retrieve API calls (as in [8]), we first collect the concepts activated by questions and documents on LLM by its SAE. We then use the trained concept predictor to predict missing concepts in input questions. Finally, we rank API calls using union joint based on concept matching. Appendix C.1 presents examples of ambiguous questions in tool calling, while Appendix C.2 illustrates concept matching in this context.

For efficiency, we use LightGBM [22] to train the concept predictor. For each concept activated by the training data and documents, the predictor is trained to determine whether it is missing in query:

$$p\big(y = 1 \mid x\big) \;=\; \sigma\Big(\sum_{t=1}^{T} \eta\, f_t(x)\Big) \;=\; \frac{1}{1 + \exp\big(-\sum_{t=1}^{T} \eta\, f_t(x)\big)} \tag{9}$$

4. Experiments

In this section, we conduct three sets of experiments:

1. Ambiguity detection by our proposed distance metrics: investigate whether the distances between questions and their interpretations can distinguish ambiguous questions from unambiguous ones.
2. Ambiguity sensitivity improvement of LLMs: We show if adding missing concepts can improve LLM's self-judgment on ambiguous questions. (see Appendix B.3)
3. Ambiguity resolution on agentic tool calling: We investigate whether predicting missing concepts in ambiguous questions can reduce the number of incorrect responses.

4.1. Ambiguity Detection

Experiment Settings. To evaluate the effectiveness of our method for ambiguity detection by distance differences, we conduct experiments primarily on AMBROSIA [6], a benchmark designed for parsing ambiguous questions into database queries across multiple domains. The benchmark consists of 1,277 ambiguous questions, each paired with human-provided unambiguous interpretations

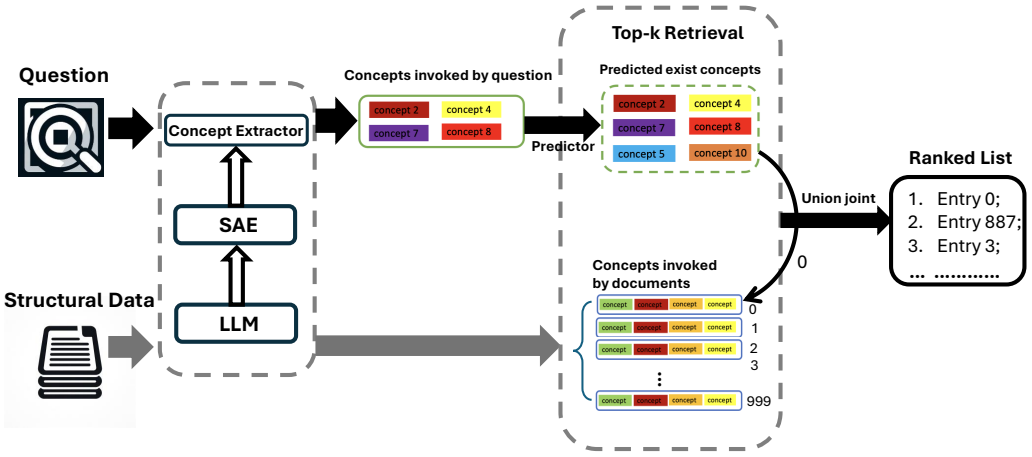


Figure 4: Tool calling framework based on missing concept prediction in ambiguous questions.

and corresponding SQL queries (2,965 in total), spanning 846 multi-table databases across 16 distinct domains. It includes three types of ambiguity: scope ambiguity, attachment ambiguity, and vagueness, and goes beyond earlier datasets that assume a single "correct" query, offering a rigorous evaluation yardstick for models that must both detect ambiguity and enumerate all valid SQL programs.

We first prompt LLMs to generate interpretations for the ambiguous questions in the dataset. Specifically, we use LLama-3.3-70B-Instruct [19] for this task (see Appendix B.1 for a prompting example). For each ambiguous question, we generate two interpretations, i_1 and i_2. These interpretations are then treated as unambiguous questions and are each further prompted to generate their own interpretations. As a result, for both ambiguous and unambiguous questions, we obtain two interpretations each, forming a triplet (q, i_1, i_2).

Next, we compute the distances between the original question and its interpretations on AMBROSIA: $\mathrm{D}(q, i_1)$, $\mathrm{D}(q, i_2)$, and $\mathrm{D}(i_1, i_2)$. These distances are calculated using both our path kernel-based method and traditional dense vector-based methods. For comparing with both Embedding model and Generation models, we use SFR-Embedding-Mistral [23] and LLama-3.3-70B-Instruct to generate dense embeddings, with distances computed as follows:

$$\mathrm{D}(\mathbf{x}, \mathbf{x}') \;=\; 1 - \frac{\mathbf{E}(\mathbf{x}) \cdot \mathbf{E}(\mathbf{x}')}{\|\mathbf{E}(\mathbf{x})\| \, \|\mathbf{E}(\mathbf{x}')\|} \tag{10}$$

We further analyze the computed distances using the following two ways:

1. We compute the average of the three distances (by Equation 7) - $\mathrm{D}_1(q, i_1)$, $\mathrm{D}_1(q, i_2)$, $\mathrm{D}_1(i_1, i_2)$) - and plot the distribution of these mean values to show patterns.
2. We normalize the distances (by Equation 8) using the ratios $\mathrm{D}_2(q, i_1)/\mathrm{D}_2(i_1, i_2)$ and $\mathrm{D}_2(q, i_2)/\mathrm{D}_2(i_1, i_2)$, and plot these normalized values to reveal potential patterns.

The results from dense vector-based methods serve as baselines for comparison.

Results. Figure 5 presents the results using our path kernel-based method (with SAE), as well as two dense vector-based methods: one using SFR-Embedding-Mistral and the other using LLama-3.3-70B-Instruct. The horizontal axis shows the average distance assigned to each sample, computed as $\overline{(\mathrm{D}_1(q, i_1), \mathrm{D}_1(q, i_2), \mathrm{D}_1(i_1, i_2))}$, for both ambiguous questions and unambiguous questions in the AMBROSIA dataset. Moving along the x-axis from left to right corresponds to increasing average distance. The vertical axis represents the absolute frequency, i.e., the raw number of observations falling into each of the 40 equal-width histogram bins. Superimposed on the histogram bars are

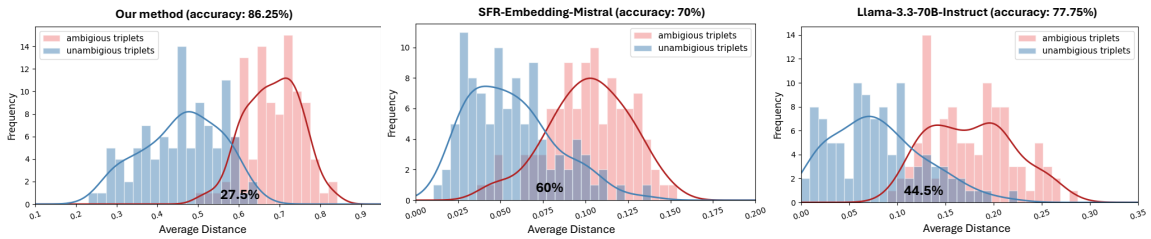


Figure 5: Distribution of average distances calculated using the path kernel method with SAE, and dense vector-based methods with SFR-Embedding-Mistral and LLama-3.3-70B-Instruct. A smaller overlapping area indicates a stronger ability to distinguish ambiguous from unambiguous questions.

kernel density curves, scaled so that their peaks align with the same frequency units. This allows for a direct visual comparison between the smooth density estimates and the discrete histogram counts.

As shown in the figure, our method results in fewer overlapping samples (27.5%) between ambiguous and unambiguous questions compared to the dense vector-based methods. Specifically, when using the x-coordinate of the intersection point of the red and blue density curves as a threshold for distinguishing ambiguous from unambiguous questions, the detection accuracies are as follows: Path kernel-based method (with SAE): 86.25%, Dense vector method with SFR-Embedding-Mistral: 70%, and Dense vector method with LLama-3.3-70B-Instruct: 77.75%. As a comparison, the Zero-shot accuracy of the Llama3-70B is 46.31%. We also visualise the distance relationship in Appendix B.2.

4.2. Agentic Tool Calling

Experiment Settings. We evaluate our tool-calling framework (Figure 4) on the Gorilla dataset [12]. This multi-faceted benchmark contains about 1.6K ML-oriented API call templates sourced from HuggingFace, TorchHub, and TensorHub. The dataset includes training and test sets, as well as API collections that support retrieval-augmented generation (RAG). We analyzed the API call results (including both the API calls and their domains) and found that ambiguity is a major factor contributing to performance degradation. See Appendix C.1 for an illustrative example.

We evaluate our framework on the Gorilla dataset using several baselines: the Gorilla base model (7B), the Gorilla fine-tuned model (fine-tuned on the TensorFlow Hub API dataset), and fine-tuned Gorilla with BM25 and GPT-based retrievers. As the Gorilla base model is relatively small, for fair comparison, we also use a 7B model, Mistral-7B[24] with its sparse-autoencoder[25] to implement our method. Additionally, we include SFR-Embedding-Mistral [23] as a baseline [1].

To predict the missing concepts in queries, we train a LightGBM model. We then evaluate the performance of our framework on the test data by using the predicted concepts to retrieve relevant API calls from the API collections. Considering the extra computational cost introduced by the sparse autoencoder (SAE), we do not retrieve all the concepts activated by the query. Instead, we select the top 50%, 30%, and 20% of the activated concepts, ranked by their activation values.

Results. Figure 6 shows the performance of our concept retrieval method compared to the baselines on the Gorilla TensorFlow Hub API bench. We evaluate the accuracy of identifying the correct API domains and retrieving the correct API calls. The red highlight shows the performance of our method, demonstrating that when using the top 50% of activated concepts, our approach achieves the highest accuracy in retrieving API calls. Accuracy of retrieving the correct domain is only slightly lower than Fine-tuned 0-shot Gorilla. We note that even when using only the top 20% of concepts (reduce the computational cost introduced by SAE), our method still outperforms all retrieval based baselines.

[1] SFR-Embedding-Mistral is ranked among the top 5 models on the MTEB leaderboard [26].

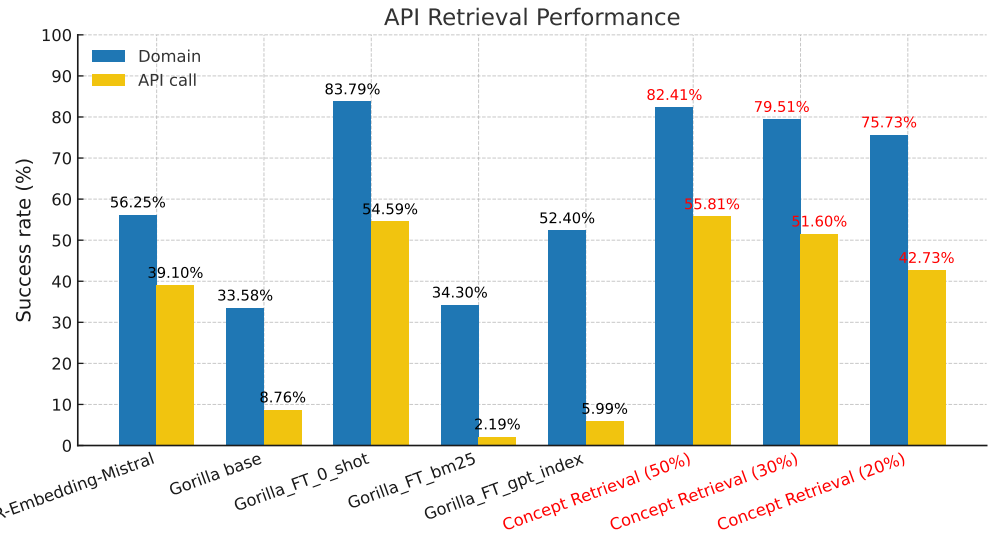


Figure 6: API bench Retrieval Results. Our methods are highlighted in red.

5. Conclusion

In this paper, we designed a novel concept-based method for ambiguity resolution in LLMs. Our method distilled concepts from ambiguous utterances and their associated interpretations, inferred the pattern of their difference in the latent space and leveraged the difference for ambiguity resolution. We demonstrated that out method outperformed baselines on the text-to-SQL task. We also gave a new method to improve LLMs' agentic tool calling performance through missing concept prediction. The method outperformed the SOTA in APIBench.

References

[1] Sewon Min, Julian Michael, Hannaneh Hajishirzi, and Luke Zettlemoyer. Ambigqa: Answering ambiguous open-domain questions. *arXiv preprint arXiv:2004.10645*, 2020.

[2] Ivan Stelmakh, Yi Luan, Bhuwan Dhingra, and Ming-Wei Chang. Asqa: Factoid questions meet long-form answers. *arXiv preprint arXiv:2204.06092*, 2022.

[3] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023.

[4] Irina Saparina and Mirella Lapata. Disambiguate first parse later: Generating interpretations for ambiguity resolution in semantic parsing. *arXiv preprint arXiv:2502.18448*, 2025.

[5] Gaurav Kamath, Sebastian Schuster, Sowmya Vajjala, and Siva Reddy. Scope ambiguities in large language models. *Transactions of the Association for Computational Linguistics*, 12:738–754, 2024.

[6] Irina Saparina and Mirella Lapata. AMBROSIA: A benchmark for parsing ambiguous questions into database queries. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2024. URL `https://openreview.net/forum?id=IlFk5U9cEg`.

[7] Elias Stengel-Eskin, Kyle Rawlins, and Benjamin Van Durme. Zero and few-shot semantic parsing with ambiguous inputs. *arXiv preprint arXiv:2306.00824*, 2023.

[8] Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick SH Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. In *EMNLP (1)*, pages 6769–6781, 2020.

[9] Trenton Bricken, Adly Templeton, Joshua Batson, Brian Chen, Adam Jermyn, Tom Conerly, Nick Turner, Cem Anil, Carson Denison, Amanda Askell, Robert Lasenby, Yifan Wu, Shauna Kravec, Nicholas Schiefer, Tim Maxwell, Nicholas Joseph, Zac Hatfield-Dodds, Alex Tamkin, Karina Nguyen, Brayden McLean, Josiah E Burke, Tristan Hume, Shan Carter, Tom Henighan, and Christopher Olah. Towards monosemanticity: Decomposing language models with dictionary learning. *Transformer Circuits Thread*, 2023. https://transformer-circuits.pub/2023/monosemantic-features/index.html.

[10] Adly Templeton, Tom Conerly, Jonathan Marcus, Jack Lindsey, Trenton Bricken, Brian Chen, Adam Pearce, Craig Citro, Emmanuel Ameisen, Andy Jones, Hoagy Cunningham, Nicholas L Turner, Callum McDougall, Monte MacDiarmid, C. Daniel Freeman, Theodore R. Sumers, Edward Rees, Joshua Batson, Adam Jermyn, Shan Carter, Chris Olah, and Tom Henighan. Scaling monosemanticity: Extracting interpretable features from claude 3 sonnet. *Transformer Circuits Thread*, 2024. URL `https://transformer-circuits.pub/2024/scaling-monosemanticity/index.html`.

[11] Pedro Domingos. Every model learned by gradient descent is approximately a kernel machine. *arXiv preprint arXiv:2012.00152*, 2020.

[12] Shishir G Patil, Tianjun Zhang, Xin Wang, and Joseph E Gonzalez. Gorilla: Large language model connected with massive apis. *Advances in Neural Information Processing Systems*, 37: 126544–126565, 2024.

[13] Nelson Elhage, Tristan Hume, Catherine Olsson, Nicholas Schiefer, Tom Henighan, Shauna Kravec, Zac Hatfield-Dodds, Robert Lasenby, Dawn Drain, Carol Chen, et al. Toy models of superposition. *arXiv preprint arXiv:2209.10652*, 2022.

[14] Jonathan Pilault, Xavier Garcia, Arthur Bražinskas, and Orhan Firat. Interactive-chain-prompting: Ambiguity resolution for crosslingual conditional generation with interaction. In *Proceedings of the 13th International Joint Conference on Natural Language Processing and the 3rd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2023. URL `https://aclanthology.org/2023.ijcnlp-main.31/`.

[15] Alisa Liu, Zhaofeng Wu, Julian Michael, Alane Suhr, Peter West, Alexander Koller, Swabha Swayamdipta, Noah Smith, and Yejin Choi. We're afraid language models aren't modeling ambiguity. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 2023. URL `https://aclanthology.org/2023.emnlp-main.51/`.

[16] Hyuhng Joon Kim, Youna Kim, Cheonbok Park, Junyeob Kim, Choonghyun Park, Kang Min Yoo, Sang-goo Lee, and Taeuk Kim. Aligning language models to explicitly handle ambiguity. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, 2024. URL `https://aclanthology.org/2024.emnlp-main.119/`.

[17] Weiwei Sun, Hengyi Cai, Hongshen Chen, Pengjie Ren, Zhumin Chen, Maarten de Rijke, and Zhaochun Ren. Answering ambiguous questions via iterative prompting. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2023. URL `https://aclanthology.org/2023.acl-long.424/`.

[18] Fangwen Mu, Lin Shi, Song Wang, Zhuohao Yu, Binquan Zhang, ChenXue Wang, Shichao Liu, and Qing Wang. Clarifygpt: A framework for enhancing llm-based code generation via requirements clarification. *Proc. ACM Softw. Eng.*, 1(FSE), 2024. URL `https://doi.org/10.1145/3660810`.

[19] Meta AI. Llama-3.3-70b-instruct. https://huggingface.co/meta-llama/Llama-3.3-70B-Instruct, December 2024. Accessed: 2025-05-12.

[20] Goodfire. Llama-3.3-70B-Instruct-SAE-l50. https://huggingface.co/Goodfire/Llama-3.3-70B-Instruct-SAE-l50, 2025. Accessed: 2025-05-14.

[21] Lorenz Kuhn, Yarin Gal, and Sebastian Farquhar. Semantic uncertainty: Linguistic invariances for uncertainty estimation in natural language generation. *arXiv preprint arXiv:2302.09664*, 2023.

[22] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, 30, 2017.

[23] Rui Meng, Ye Liu, Shafiq Rayhan Joty, Caiming Xiong, Yingbo Zhou, and Semih Yavuz. Sfr-embedding-mistral:enhance text retrieval with transfer learning. Salesforce AI Research Blog, 2024. URL https://www.salesforce.com/blog/sfr-embedding/.

[24] The Mistral AI Team. Mistral-7B-v0.1. https://huggingface.co/mistralai/Mistral-7B-v0.1. Accessed: 12 May 2025.

[25] Tyler Cosgrove. Mistral-7B-sparse-autoencoder-layer16. https://huggingface.co/tylercosgrove/mistral-7b-sparse-autoencoder-layer16. Accessed: 12 May 2025.

[26] Niklas Muennighoff, Nouamane Tazi, Loïc Magne, and Nils Reimers. Mteb: Massive text embedding benchmark. *arXiv preprint arXiv:2210.07316*, 2022.

A. Algorithm: Measuring Semantic Entropy by LLM

Definition A.1 (Semantic Entropy)**.** Given a prompt x, let $\mathcal{S}$ be the set of possible sequences (texts) and let $\mathcal{C}$ be a partition of $\mathcal{S}$ into semantic equivalence classes (meanings) $c \in \mathcal{C}$. The language model induces a distribution $p(s \mid x)$ over sequences $s \in \mathcal{S}$, which pushes forward to a distribution over meanings,

$$p(c \mid x) \;=\; \sum_{s \in c} p(s \mid x).$$

The *semantic entropy* of the model at x is the Shannon entropy of this meaning distribution:

$$\mathrm{SE}(x) \;=\; -\sum_{c \in \mathcal{C}} p(c \mid x)\, \log p(c \mid x) \;=\; -\sum_{c \in \mathcal{C}} \left(\sum_{s \in c} p(s \mid x) \right) \log \left(\sum_{s \in c} p(s \mid x) \right). \tag{11}$$

At a high level the semantic entropy estimation involves three steps (Algorithm 1):

1. **Generation.** Given a prompt x, sample N sequences $s_1, \ldots, s_N \sim p(\cdot \mid x)$ from the LLM.
2. **Clustering.** Embed each sequence $z_i = \mathcal{E}(s_i)$ and group the embeddings into semantic equivalence classes $C = \{C_1, \ldots, C_{|C|}\}$ via a clustering method $\mathcal{C}$ (cosine + agglomerative with threshold). Let ℓ_i denote the class label of s_i and $C_k = \{i : \ell_i = k\}$.
3. **Entropy estimation.** Estimate the class masses $p(C_k \mid x)$ and compute entropy over meanings.

Algorithm 1: Measuring Semantic Entropy (Monte-Carlo class-average)

Input: Prompt x; num of samples N; generator $\mathcal{G}$; embedder $\mathcal{E}$; clustering procedure $\mathcal{C}$
Output: semantic entropy H_{sem}, cluster probs $\{p_k\}$, labels $\{\ell_i\}$

`// 1) Generate` N `samples and (optional) scores`
for $i \leftarrow 1$ **to** N **do**
  $y_i, s_i \leftarrow \mathcal{G}(x)$; `//` y_i`: text,` s_i`: (avg) sequence log-prob`
`// 2) Embed and cluster by meaning`
$Z_i \leftarrow \mathcal{E}(y_i)$ for $i = 1..N$;
$\ell_1, \ldots, \ell_N \leftarrow \mathcal{C}(Z_1, \ldots, Z_N)$; `// e.g., cosine + agglomerative`
`// 3) Estimate cluster probabilities`
Let K be the number of distinct clusters among $\{\ell_i\}$;
if *using counts* **then**
  $n_k \leftarrow \sum_{i=1}^{N} \mathbf{1}[\ell_i = k]$; $p_k \leftarrow \frac{n_k}{N}$ for $k = 1..K$;
else
  `// probability-weighted`
  `// Stabilize weights with log-sum-exp`
  $m \leftarrow \max_i s_i$; $\tilde{w}_i \leftarrow \exp(s_i - m)$; $w_i \leftarrow \frac{\tilde{w}_i}{\sum_{j=1}^{N} \tilde{w}_j}$;
  $p_k \leftarrow \sum_{i=1}^{N} w_i \cdot \mathbf{1}[\ell_i = k]$ for $k = 1..K$;
`// 4) Ensure numerical safety (clip and renormalize)`
$\varepsilon \leftarrow 10^{-12}$;
$p_k \leftarrow \max(p_k, \varepsilon)$; $p_k \leftarrow \frac{p_k}{\sum_{j=1}^{K} p_j}$ for $k = 1..K$;
`// 5) Calculate semantic entropy`
$H_{\text{sem}} \leftarrow -\sum_{k=1}^{K} p_k \log_b p_k$ (default $b = 2$ for bits);
return H_{sem}, $\{p_k\}$, $\{\ell_i\}$;

B. Ambiguity Detection: Prompt examples and Additional Results

B.1. Few-shot Examples used in Prompts to Inject Concepts

Here we provide an example showing how concept-embodied examples can guide an LLM to generate diverse interpretations (Figure 7).

```
Example_1 = "question: What brands of agricultural machinery are available in each machinery
store\n
**interpretations**:
1. Which brands of machinery are equally available in all agricultural machinery stores?\n
2. For each agricultural machinery store, show which brands of machinery are available?\n"

Example_2 = "question: List the price of products sold in every duty-free shop.\n
**interpretations**:
1. For each duty-free shop, list the prices of all the products they sell.\n
2. What is the price of each product that is sold in all duty-free shops.\n"

instruction = f"Below question is ambiguous, and it has 2 interpretations. Please you generate
these 2 interpretations for this question. Here are two examples: Example 1: {Example_1}
Example 2: {Example_2} Now please generate 2 interpretations for below question. Don't answer
it is ambiguous or not, only answer the 2 interpretations. \n**Question**: " + question +
"\n**interpretations**:\n1. "
```

Figure 7: An example to prompt LLM to generate diverse interpretations.

B.2. Distance Relationship

In Figure 8, we normalize the distances (Equation 8) for the distances between q, i_1, i_2 using the ratios $\mathrm{D}_2(q, i_1)/\mathrm{D}_2(i_1, i_2)$ and $\mathrm{D}_2(q, i_2)/\mathrm{D}_2(i_1, i_2)$, and plot these normalized values to reveal potential patterns. In this case, no concept mask is applied during distance calculation. Our goal is to examine distance patterns when using equation 8 with all activated concepts valid. For clarity, we visualize 100 samples for each case. We find that compared to unambiguous questions, interpretations for ambiguous questions are more concentrated and more symmetrically distributed in their distances to the questions.

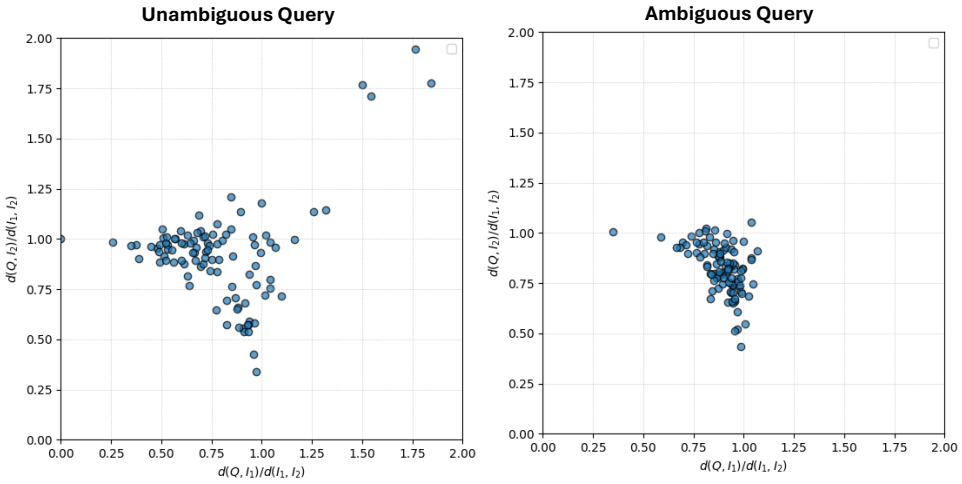
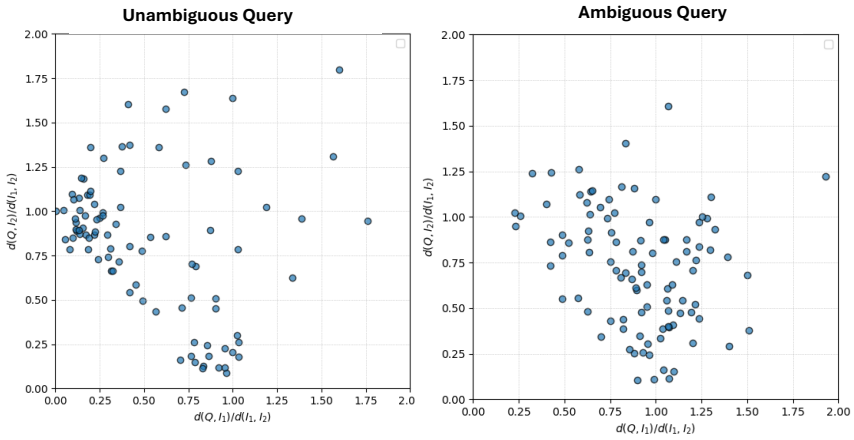
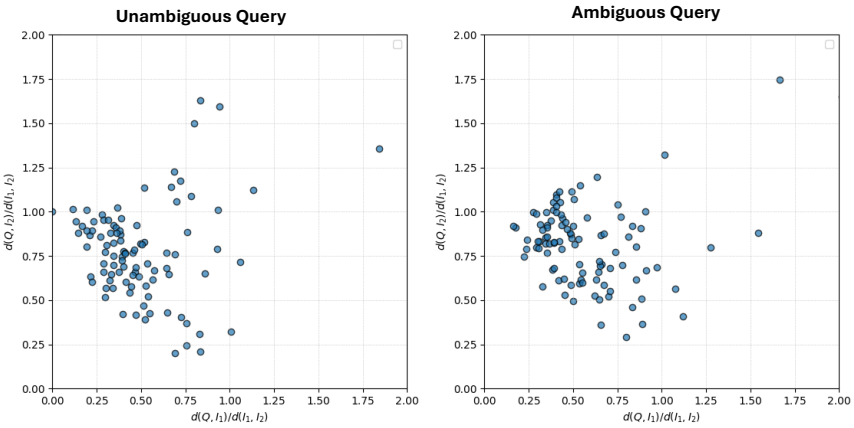


Figure 8: Normalized distances (Equation 8) for the distances between questions and their interpretations using the ratios $\mathrm{D}_2(q, i_1)/\mathrm{D}_2(i_1, i_2)$ and $\mathrm{D}_2(q, i_2)/\mathrm{D}_2(i_1, i_2)$, compared to unambiguous and ambiguous questions' distance triplets cluster to the center of the map.

Figure 9 shows the results of the baselines, we can see that distance calculations with dense vectors generated by both generation and embedding models cannot show the symmetric pattern of ambiguous questions and their interpretations. As such, we can not distinguish each data point is ambiguous or not by their measurements.



(a) Llama-3.3-70B-instruct model



(b) SFR-Embedding-Mistral model

Figure 9: Normalized distances for the baselines.

B.3. Ambiguity sensitivity improvement of LLMs with missing concept addition

To demonstrate the connection between targeted concepts and ambiguity resolution, we conducted experiments with LLaMA3-70B on questions involving scope ambiguity. Below is an example (an ambiguous question and its interpretations):

> Ambiguous question: "What brands of machinery are available in each machinery store?";
> Interpretation 1: "Which brands of machinery are equally available in all machinery stores?";
> Interpretation 2: "For each machinery store, show which brands of machinery are available."

As the missing concepts were found to correlate with tokens like "For," "each," "Which," "What," "all," "?", and "common", we first identified the concepts invoked by these tokens, and then manually increased the activation values of these concepts to 1.0. As results in table 1 show, we found that the accuracy of ambiguity detection on unambiguous questions increased from 39.8% to 60.5%. In contrast, when the same number of random concepts were activated instead, the accuracy dropped to just 0.2%. Although this will result in a 16.4% decrease in the ambiguous question detection accuracy, the overall accuracy will increase from 46.31% to 54.61%. And as unambiguous questions and ambiguous questions in dataset are unbalanced, the unambiguous accuracy increasing indicates that target activation significantly improves the model's false positive rate, rather than simply misclassify

most queries as ambiguous. And this result also shows that only targeted concept activation helps identify ambiguity, whereas randomly activating concepts only bring interference.

Table 1: Effect of concept activation on ambiguity detection and overall accuracy.

Metric	Baseline	Targeted activation	Random activation	vs. baseline
Unambiguous accuracy	39.8%	60.5%	0.2%	+20.7 pp (targeted)
Ambiguous accuracy	59.3%	42.8%	—	−16.4 pp
Overall accuracy	46.31%	54.61%	—	+8.30 pp

Notes. "pp" = percentage points. Unambiguous questions and ambiguous questions in dataset are unbalanced. (N_unambiguous : N_ambiguous = 2 : 1)

C. Agentic Tool Calling: Prompt and Concept Matching Examples

C.1. Ambiguous Prompt Examples

Instruction: "Find out what's in the image taken by a wildlife photographer, so we can determine the ***main subject*** of the picture.\n###Input: An image taken by a wildlife photographer."

The API assistant searched:


```
{"domain": "Image object detection",
 "framework": "TensorFlow Hub",
 "functionality": "Detect objects in images",
 "api_name": "model_id",
 "api_call": "hub.load('https://tfhub.dev/tensorflow/ssd_mobilenet_v2/2')",
 "api_arguments": ["model_id"],
 "python_environment_requirements": ["tensorflow", "tensorflow_hub"],
 "performance": {"dataset": "COCO", "accuracy": "0.320"},
 "description": "A pre-trained TensorFlow Hub model for detecting objects in images using the Single Shot
MultiBox Detector (SSD) architecture with MobileNet V2 as the base network."}
```


Instruction: "Find out what's in the image taken by a wildlife photographer, so we can determine the ***object*** of the picture.\n###Input: An image taken by a wildlife photographer."

The API assistant searched:


```
{"domain": "Image feature vector",
 "framework": "TensorFlow Hub",
 "functionality": "Feature extraction",
 "api_name": "model_id",
 "api_call":
"hub.KerasLayer('https://tfhub.dev/google/imagenet/mobilenet_v2_100_224/feature_vector/4')",
 "api_arguments": {"model_id": "string", "input_shape": "tuple", "trainable": "boolean"},
 "python_environment_requirements": ["tensorflow", "tensorflow_hub"],
 "performance": {"dataset": "ImageNet", "accuracy": "Top-1 accuracy"},
 "description": "A pre-trained image feature vector model for image classification and transfer
learning, based on MobileNetV2 architecture."}
```


Ground truth


```
{"domain": "Image classification",
 "framework": "TensorFlow Hub",
 "functionality": "Image classification using pre-trained model",
 "api_name": "imagenet_mobilenet_v2_100_224_classification",
 "api_call":
"hub.KerasLayer('https://tfhub.dev/google/imagenet/mobilenet_v2_100_224/classification/4')",
 "api_arguments": {"url":
"https://tfhub.dev/google/imagenet/mobilenet_v2_100_224/classification/4"},
 "python_environment_requirements": {"tensorflow": ">=2.0.0", "tensorflow_hub": ">=0.12.0",
"numpy": ">=1.19.5", "PIL": ">=8.3.2"},
 "performance": {"dataset": "ImageNet", "accuracy": "71.8%"},
 "description": "A pre-trained image classification model using MobileNetV2 architecture on
ImageNet dataset with 100% depth and 224x224 input size."
```


Figure 10: An example illustrating ambiguity in agentic tool calling (from the Gorilla dataset, using the Gorilla model as the API assistant). The red highlight marks differences in the instructions. Minor changes to the instruction can steer the LLM's answer, and may even shift the domain of the returned API.

C.2. A Concept Matching Example

Figure 11 illustrates how the concepts that involve which activated by the input question and which predicted by pre-trained predictor are matched to those in the structured API document through the union joint operator.

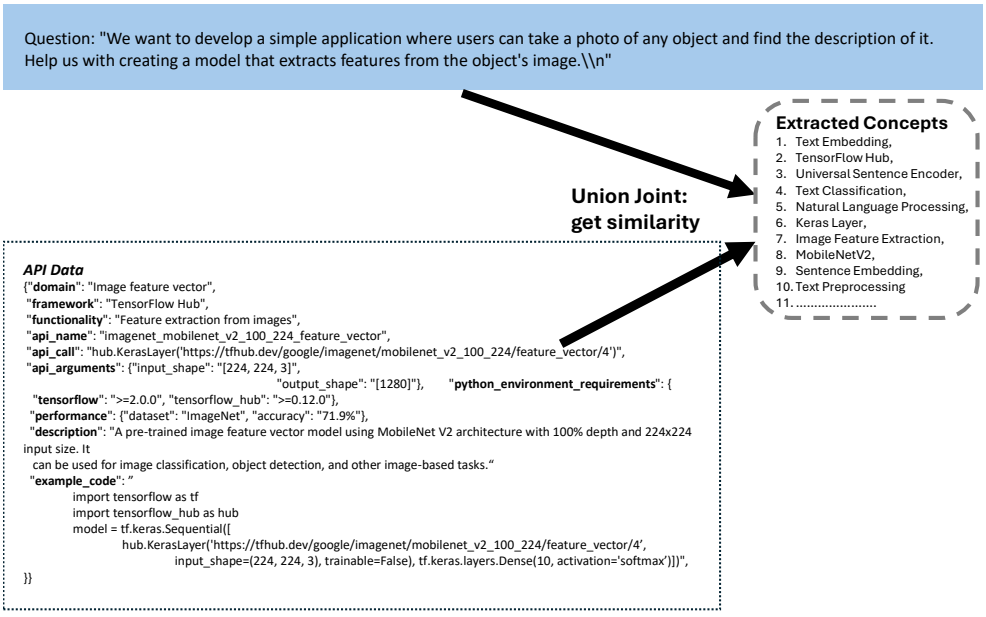


Figure 11: An example for getting similarity for extracted concepts by union joint.

D. Limitations

Our method was evaluated on limited datasets. While results on both ambiguity and API datasets demonstrate its effectiveness, these datasets cover only a subset of known ambiguity scenarios, leaving it unclear whether our interpretation-generation method generalizes to other types of ambiguity in natural language. Investigating this question is left for future work.

E. Computing Resources

Our experiments were conducted on four NVIDIA H100 GPU node, each with 96GB memory.