LLM Representations for Enhanced Asymmetric Error Classification

Isha Chaudhary [1] **Jason Vega** [1] **Mahmood Sharif** [2] **Nina Narodytska** [3] **Gagandeep Singh** [1]

Abstract

Large Language Models (LLMs) find applications in (binary) classification tasks such as question answering and response moderation. However, they can achieve suboptimal performance in asymmetric error classification tasks where the Type-I (false positive) error has a higher cost than the Type-II (false negative) error. In this work, we adopt an interpretability-guided approach to improve over LLM performance for asymmetric error classification scenarios in a cost-aware manner. Specifically, we design linear probes in the LLM representation space as *Neyman-Pearson classifiers* that predict the ground-truth classification, so that the classifier has the Type-I error constrained by a user-specified threshold while minimizing the Type-II error. We find that Neyman-Pearson linear classifiers on LLM representations from the residual streams can provide a better trade-off between the different errors than SOTA LLMs in medical question-answering, safety moderation, and schema linking asymmetric error classification tasks by achieving constrained Type-I and low Type-II errors.

1. Introduction

Large Language Models (LLMs) are used for classification tasks, where their output is constrained to a fixed set of possibilities. Popular examples are multiple choice question answering (QA) (Zhu et al., 2020; Pal et al., 2022; Hendrycks et al., 2021; Chaudhary et al., 2024), where few of a fixed set of answer options are correct; and safety moderation involving classification of given conversations between a model and its user as safe or unsafe. These tasks are important both as standalone problems (e.g., multiple choice QA) and as part of a larger LLM agent system (e.g., safety moderation). General classification tasks can be modeled as a union of several binary classification tasks, where each task is to identify whether a given class is a ground-truth class. Hence, we scope our discussion to binary classification.

Several practical instances (Zhu et al., 2020, QA), (Ji et al., 2023, safety moderation), (Talaei et al., 2024, schema linking) of binary classification tasks have asymmetric errors, where misclassification of samples from one class is costlier than those from the other. LLM performance in asymmetric error, binary classification tasks can be suboptimal (§3, Table 1). This is owing to the general purpose nature of LLMs that are not optimized for asymmetric-error classification tasks. Prior works (Jin et al., 2025; Lei and Cooper, 2025) recognize that due to their extensive training (Ibrahim et al., 2024) and emergent capabilities (Wei et al., 2022), LLMs extract important features for the final classification task and store them in their intermediate representations. However, they may not be able to properly combine the intermediate features to achieve good performance on their own. Our insight is to optimize on linearly combining the features to achieve good classification performance in asymmetric error scenarios. Ideally, we want misclassification with higher cost to be constrained by some user-defined tolerance threshold, while the other kind of misclassification is minimized. This would ensure optimal classifier performance with the high-cost error being constrained.

Key challenges. (1) Prior works on prompt tuning (Maharjan et al., 2024), fine-tuning (Hu et al., 2021), or training additional linear classifiers on LLM representations (Orgad et al., 2024) focus on accuracy as the primary optimization metric. There are no existing methods on textual inputs, to the best of our knowledge, that consider the differential costs of misclassification errors. (2) It is unclear which LLM intermediate representations can result in the most optimal classification performance.

Our approach. In this work, we develop a method to improve classification performance in asymmetric error classification tasks by using specialized linear classifiers in the LLM's intermediate representation space to predict the final classification results. We design these linear classifiers as *Neyman-Pearson* (NP) classifiers that threshold the high-cost misclassification error below a user-specified threshold while minimizing the other misclassification error. A benefit of using this approach is that the linear model remains the same for different user-specified thresholds, changing only the decision boundary by changing the threshold on the linear combination in the classifier. We train individual NP classifiers on the residual streams of all layers and identify that the residual stream of middle to final LLM layers can be used to train classifiers that achieve good performance.

Contributions.

- *Formulation*. We study linear classification in intermediate representation spaces of LLMs using the Neyman-Pearson framework. This approach aims to improve upon baseline LLM classification performance by first training a linear model, then adjusting its decision threshold to constrain the high-cost misclassification error while minimizing the other error.
- *Implementation*. We find that residual streams from the middle to the final layers are useful to achieve strong classification performance, and we use representations from these layers to train the classifiers. Our implementation is available at `https://anonymous.4open.science/r/llm-reps-0546`.
- *Findings*. The trained NP classifiers show a significant improvement over the baseline LLM performance in terms of lower Type-I errors (costlier). We generally achieve testing Type-I errors below 10%. The average Type-I error reduction from the original LLM as classifier is $\mathbf{34}\%$.

2. LLM representations for asymmetric error classification

The LLM residual stream (intermediate representations) encodes various features that crucially contribute to the final generation (Jin et al., 2025; Lei and Cooper, 2025). These features at each layer are formed using linear combinations of features in previous layers followed by non-linear transformations. General purpose LLMs are able to extract useful features due to their extensive training (Naveed et al., 2024; Ibrahim et al., 2024) and emergent capabilities (Wei et al., 2022). These features can be combined appropriately to form the final classification result, as observed by Lei and Cooper (2025). However, the LLMs themselves may not be able to correctly utilize the features to finally generate the correct answer to classification tasks. Hence, for the goal of improved downstream classification, we identify and effectively combine the intermediate features using linear models, similar to Orgad et al. (2024). We train linear models on LLM intermediate representations for task-specific prompts and specific layers.

However, simple linear models that minimize overall classification errors do not suffice for asymmetric classification errors, where one error has a higher cost than the other. Hence, we need specialized classifiers that take the asymmetric nature of the errors into account. Next, we define the two kinds of errors in asymmetric error binary classification for a classifier $\phi : \mathbb{R} \to \{0, 1\}$ that maps a $d-$dimensional ($d > 0$) real-valued space to either class-0 or class-1. Note that multiclass classification can be modeled as C (number of classes) binary classifications, where each task is to classify a given class as a ground truth class. Hence, we scope our method to binary classification.

Type-I error. Type-I error or false positives $R_0(\phi)$ occur when a sample X with ground truth (Y) class-0 is predicted as class-1 by ϕ.

$$R_0(\phi) \coloneqq \mathbb{P}(\phi(X) \neq Y \mid Y = 0) \tag{1}$$

Type-II error. Type-II error or false negatives $R_1(\phi)$ occur when a sample X with ground truth (Y) class-1 is predicted as class-0 by ϕ.

$$R_1(\phi) \coloneqq \mathbb{P}(\phi(X) \neq Y \mid Y = 1) \tag{2}$$

In this analysis, we consider the misclassification of class-0 or Type-I error as more costly than the Type-II error and hence constrain it to be below a user-specified tolerance threshold $\alpha > 0$. This choice is without loss of generality, as the class labels can be flipped to make class-0 the more important class in terms of misclassification. Among all classifiers ϕ' that satisfy $R_0(\phi') \leq \alpha$, the classifier with the minimum Type-II error $R_1(\phi)$ is desirable. Hence, to find the optimal classifier ϕ^* that has constrained Type-I error with minimum Type-II error, we solve the following optimization problem.

$$\phi^* \coloneqq argmin_{\phi | R_0(\phi) \leq \alpha} R_1(\phi) \tag{3}$$

To solve (3), we apply the umbrella Neyman-Pearson (NP) algorithm (Tong et al., 2018) on linear models on LLM intermediate representations for each layer to make them NP classifiers, as follows. Let f be a linear model that takes as input an intermediate representation vector, X and outputs a scalar that is normalized to a (classification) score between 0 and 1. The NP classifier derived from f is such that the true Type-I error is thresholded by a user-specified threshold $\alpha > 0$ with probability more than $1 - \delta, \delta > 0$, i.e., $Pr(R_0(\phi) \leq \alpha) \geq 1 - \delta$. The NP classifier is given by $\phi^*(X) = \mathbb{I}\{f(X) > \tau\}$, where τ is a constant threshold. To determine τ, the order statistics of the classification scores of f over a held-out validation set of class-0 samples are used. To ensure that $Pr(R_0(\phi) \leq \alpha) \geq 1 - \delta$, Tong et al. (2018) show that the k^{th} order statistic for a validation set of size $n > 0$ should be used as the threshold, where $k \coloneqq min\{k \in \{1, \ldots, n\} \mid v(k) \leq \delta\}$ and $v(k)$ is a violation upper bound given by $v(k) \coloneqq \sum_{j=k}^{n} \binom{n}{j} (1-\alpha)^j \alpha^{n-j}$.

3. Experiments

We show empirical evidence for performance improvement using our approach in the following evaluations. We conducted our experiments on 4 A100 GPUs with 40GB VRAM each. We obtain the LLM representations consisting of the

Table 1. Performance of baselines and our method. For methods using LLM intermediate representations, we report the test set performance of the classifiers with lowest Type-II error out of all classifiers with Type-I error $\leq \alpha$ trained on representations from individual LLM layers. Instances with Type-I error below α are highlighted. We also report the layers (alongside the total number of hidden layers in the model) whose representations were used to train the best classifiers, whose performance is shown.

Task	Model	LLM as classifier		Linear classifier			NP classifier		
		Type-I	Type-II	Type-I	Type-II	Layer/Total	Type-I	Type-II	Layer/Total
MedQA	Llama-3.2 (3B)	0.32	0.35	1.0	0.0	1/28	0.11	0.53	18/28
	Qwen-2.5 (3B)	0.70	0.08	1.0	0.0	1/36	0.11	0.53	33/36
	Llama-3.1 (8B)	0.40	0.20	0.997	0.004	32/32	0.11	0.48	31/32
	Qwen-2.5 (7B)	0.54	0.09	0.97	0.02	27/28	**0.10**	0.51	28/28
	Qwen-2.5 (14B)	0.36	0.19	0.96	0.04	41/48	0.11	0.41	48/48
SM	Llama-Guard-3 (1B)	0.51	0.03	0.48	0.48	5/16	**0.08**	0.38	14/16
	Llama-Guard-3 (8B)	0.46	0.04	0.49	0.53	4/32	**0.09**	0.29	14/32
SL	Llama-3.2 (3B)	0.23	0.45	1.0	0.0	1/28	0.14	0.26	28/28
	Qwen-2.5 (3B)	0.19	0.34	1.0	0.0	1/36	**0.05**	0.16	25/36
	Llama-3.1 (8B)	0.54	0.10	1.0	0.0	1/32	**0.08**	0.11	31/32
	Qwen-2.5 (7B)	0.62	0.04	1.0	0.0	1/28	**0.09**	0.11	28/28
	Qwen-2.5 (14B)	0.39	0.07	0.85	0.08	31/48	0.12	0.12	34/48

residual stream of each layer, *after* the generation of the classification token. We study the following asymmetric error classification tasks with SOTA LLMs. We give the specific prompts used for each task in Appendix A.

- **Medical QA** (MedQA). In a specific instance of medical question answering, we use the MashQA (Zhu et al., 2020) dataset. The classification task is—given a query and a sentence from the accompanying context, the model needs to classify the sentence as relevant or not for answering the query. This is an asymmetric error classification task, where misclassifying actually relevant sentences is costlier for downstream question answering than misclassifying actually irrelevant ones, which merely add noise. The training set for the classifiers comprises LLM representations derived from $35,182$ prompts generated during sentence relevance classification. These prompts were constructed from 800 queries sampled from the MashQA dataset. The test set consists of LLM representations from 8763 prompts derived from 200 queries. The resultant datasets of LLM representations annotated by the ground truth sentence relevance are imbalanced with the training and test sets each having 92% prompts about irrelevant sentences.

- **Safety Moderation** (SM). In this task, we study moderation tools for classifying given prompt and model responses as safe or unsafe using samples from the Beaver Tails dataset (Ji et al., 2023). This is also an asymmetric error classification task, where misclassifying unsafe responses has a higher cost than misclassifying safe responses. We construct the training and test sets for this task from 8000 and 2000 samples, respectively, of single-turn conversations consisting of user query and model response pairs from Beaver Tails. These data sets are balanced.

- **Schema Linking** (SL). SL is an important step in the text-to-SQL pipeline, where, for a given natural language query about a database and a column from the database's schema, the model needs to classify whether the column is relevant to write the equivalent optimized SQL query. In SL too, misclassification of relevant columns as irrelevant is costlier, as a pruned schema is made from the relevant columns and used in the subsequent steps of the pipeline. We show SL over queries from the challenging Formula-1 schema of the BIRD-SQL benchmark (Li et al., 2024). We derive LLM representations for the training and test set from 7520 and 1880 column relevance classification prompts derived from 80 and 20 natural language queries respectively, based on Formula-1 schema. This is an imbalanced classification task as well with 94% prompts about irrelevant columns.

We experiment with the following open-source LLMs, as we need intermediate representations to train our classifiers—Llama-3.2 (3B) Instruct, Llama-3.1-8B Instruct (Grattafiori and et al., 2024), and Qwen-2.5-Coder Instruct (3B,7B,14B) (Hui et al., 2024) for MedQA and SL and moderation LLMs Llama Guard 3 (1B,8B) (Inan et al., 2023) for SM. With this choice of models, we study a variety of model sizes and families to make our study extensive. We compare with the following classification baselines.

- **LLM as classifier**. We report the performance of the

target LLM, prompted for classification. The response is generated with greedy decoding.

- **Using simple linear classifiers**. We train simple linear classifiers on LLM representations with default decision thresholds. We use Scikit-Learn's (Pedregosa et al., 2011) implementation of Logistic Regression classifiers with default settings.

We report Type-I and Type-II errors on held-out test sets. For the Neyman-Pearson (NP) settings, we set the permissible threshold α for Type-I error as 10% and probability of Type-I error exceeding α, δ as 5%. We train Neyman-Pearson (NP) classifiers as Logistic Regression models on a subset of the training data. To implement the NP formulation, we set aside 50% of the class-0 samples as a held-out validation set, and exclude them from the classifier's training set. This validation set is then used to adjust the decision boundary to form the NP classifier. For each LLM and task, we train separate classifiers using the residual streams of each layer as input representations. We show the performance of the best performing classifier, selected based on held-out validation sets comprising 10% of the original training data for each layer. The remaining 90% of the data is used to train the classifiers. Table 1 presents our results.

Key observations.

- **NP effectively controls Type-I error**. The original LLM classification performance can show high Type-I error, which is undesirable. Similarly, the simple linear classifiers do not prioritize Type-I error reduction, hence, they are not well-suited for asymmetric error classification tasks. NP classifiers achieve testing Type-I error that is generally lower than α, with only minor violations. This is even in the imbalanced tasks of MedQA and SL, where simple linear classifiers perform suboptimally. The NP classifiers also show low Type-II error.
- **Best classifiers appear for middle to final layers**. We observe that the best performing NP classifiers are trained from the LLM residual streams for the middle to final layers for the studied tasks. Future works could investigate specific features that are extracted in these layers that lead to the best classification performance.

References

Isha Chaudhary, Vedaant V Jain, and Gagandeep Singh. Decoding intelligence: A framework for certifying knowledge comprehension in llms. *arXiv preprint arXiv:2402.15929*, 2024.

Aaron Grattafiori and et al. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding, 2021. URL https://arxiv.org/abs/2009.03300.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models, 2021. URL https://arxiv.org/abs/2106.09685.

Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Keming Lu, Kai Dang, Yang Fan, Yichang Zhang, An Yang, Rui Men, Fei Huang, Bo Zheng, Yibo Miao, Shanghaoran Quan, Yunlong Feng, Xingzhang Ren, Xuancheng Ren, Jingren Zhou, and Junyang Lin. Qwen2.5-coder technical report, 2024. URL https://arxiv.org/abs/2409.12186.

Adam Ibrahim, Benjamin Thérien, Kshitij Gupta, Mats L. Richter, Quentin Anthony, Timothée Lesort, Eugene Belilovsky, and Irina Rish. Simple and scalable strategies to continually pre-train large language models, 2024. URL https://arxiv.org/abs/2403.08763.

Hakan Inan, Kartikeya Upasani, Jianfeng Chi, Rashi Rungta, Krithika Iyer, Yuning Mao, Michael Tontchev, Qing Hu, Brian Fuller, Davide Testuggine, and Madian Khabsa. Llama guard: Llm-based input-output safeguard for human-ai conversations, 2023. URL https://arxiv.org/abs/2312.06674.

Jiaming Ji, Mickel Liu, Juntao Dai, Xuehai Pan, Chi Zhang, Ce Bian, Chi Zhang, Ruiyang Sun, Yizhou Wang, and Yaodong Yang. Beavertails: Towards improved safety alignment of llm via a human-preference dataset, 2023. URL https://arxiv.org/abs/2307.04657.

Mingyu Jin, Qinkai Yu, Jingyuan Huang, Qingcheng Zeng, Zhenting Wang, Wenyue Hua, Haiyan Zhao, Kai Mei, Yanda Meng, Kaize Ding, Fan Yang, Mengnan Du, and Yongfeng Zhang. Exploring concept depth: How large language models acquire knowledge and concept at different layers?, 2025. URL https://arxiv.org/abs/2404.07066.

Ge Lei and Samuel J. Cooper. The representation and recall of interwoven structured knowledge in llms: A geometric and layered analysis, 2025. URL https://arxiv.org/abs/2502.10871.

Jinyang Li, Binyuan Hui, Ge Qu, Jiaxi Yang, Binhua Li, Bowen Li, Bailin Wang, Bowen Qin, Ruiying Geng, Nan Huo, et al. Can llm already serve as a database interface? a big bench for large-scale database grounded text-to-sqls. *Advances in Neural Information Processing Systems*, 36, 2024.

Jenish Maharjan, Anurag Garikipati, Navan Preet Singh, Leo Cyrus, Mayank Sharma, Madalina Ciobanu, Gina Barnes, Rahul Thapa, Qingqing Mao, and Ritankar Das. Openmedlm: Prompt engineering can out-perform fine-tuning in medical question-answering with open-source large language models, 2024. URL https://arxiv.org/abs/2402.19371.

Humza Naveed, Asad Ullah Khan, Shi Qiu, Muhammad Saqib, Saeed Anwar, Muhammad Usman, Naveed Akhtar, Nick Barnes, and Ajmal Mian. A comprehensive overview of large language models, 2024. URL https://arxiv.org/abs/2307.06435.

Hadas Orgad, Michael Toker, Zorik Gekhman, Roi Reichart, Idan Szpektor, Hadas Kotek, and Yonatan Belinkov. Llms know more than they show: On the intrinsic representation of llm hallucinations, 2024. URL https://arxiv.org/abs/2410.02707.

Ankit Pal, Logesh Kumar Umapathi, and Malaikannan Sankarasubbu. Medmcqa: A large-scale multi-subject multi-choice dataset for medical domain question answering. In Gerardo Flores, George H Chen, Tom Pollard, Joyce C Ho, and Tristan Naumann, editors, *Proceedings of the Conference on Health, Inference, and Learning*, volume 174 of *Proceedings of Machine Learning Research*, pages 248–260. PMLR, 07–08 Apr 2022. URL https://proceedings.mlr.press/v174/pal22a.html.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Shayan Talaei, Mohammadreza Pourreza, Yu-Chen Chang, Azalia Mirhoseini, and Amin Saberi. Chess: Contextual harnessing for efficient sql synthesis, 2024. URL https://arxiv.org/abs/2405.16755.

Xin Tong, Yang Feng, and Jingyi Jessica Li. Neyman-pearson classification algorithms and np receiver operating characteristics. *Science Advances*, 4(2), February 2018. ISSN 2375-2548. doi: 10.1126/sciadv.aao1659. URL http://dx.doi.org/10.1126/sciadv.aao1659.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. Emergent abilities of large language models, 2022. URL https://arxiv.org/abs/2206.07682.

Ming Zhu, Aman Ahuja, Da-Cheng Juan, Wei Wei, and Chandan K. Reddy. Question answering with long multiple-span answers. In Trevor Cohn, Yulan He, and Yang Liu, editors, *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3840–3849, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.findings-emnlp.342. URL https://aclanthology.org/2020.findings-emnlp.342/.

A. LLM prompts for different tasks

Medical Question Answering. The prompt for classifying a given sentence as relevant to answering a given query, derived from the MashQA dataset (Zhu et al., 2020), is given in Prompt 1.

Medical Question Answering. We use the standard template[1] of Llama Guard (Inan et al., 2023) to form the prompt to classify a given user-assistant conversation, derived from the Beaver Tails dataset (Ji et al., 2023), as safe/unsafe.

Schema Linking. The prompt template for schema linking, which involves identifying the relevance of a given column to generate an equivalent SQL query from a given natural language query, given to all LLMs tested, is shown in 2. This template is borrowed with minor modifications from that used in column filtering in (Talaei et al., 2024).

You are a detail-oriented doctor tasked with evaluating the relevance of given sentence for answering a specific question.
Procedure:
1. Understand the question.
2. Carefully examine the provided sentence and its relevance to answering the question.
Question: {QUESTION}
Sentence: {SENTENCE}
Take a deep breath and provide your answer in the following json format:
“‘json
{{
“is_sentence_information_relevant”: “Yes” or “No” ,
“chain_of_thought_reasoning”: “One line explanation of why or why not the sentence information is relevant to the question.”
}}
”’
Only output a json as your response.

Prompt 1: Prompt for MedQA

[1] https://www.llama.com/docs/model-cards-and-prompt-formats/llama-guard-3/

You are a detail-oriented data scientist tasked with evaluating the relevance of database column information for answering specific SQL query question based on provided hint. Your goal is to analyze the provided database schema, comprehend the posed question, and assess whether the given column details are pertinent to constructing an SQL query to address the question informed by the hint. Label the column information as "relevant" if it aids in query formulation, or "irrelevant" if it does not.
Database Schema Overview:
{SCHEMA}
Procedure:

- Carefully examine the provided schema and column details.
- Understand the question about the database and its associated hint.
- Decide if the column details are necessary for the SQL query based on your analysis.

Here are some examples of how to determine if the column information is relevant or irrelevant to the question and the hint:
{Few shot examples}
Now, its your turn to determine whether the provided column information can help formulate a SQL query to answer the given question, based on the provided hint.
The following guidelines are VERY IMPORTANT to follow. Make sure to check each of them carefully before making your decision:

- You're given only one column's information, which alone isn't enough to answer the full query. Concentrate solely on this provided data and assess its relevance to the question and hint without considering any missing information.
- Read the column information carefully and understand the description of it, then see if the question or the hint is asking or referring to the same information. If yes then the column information is relevant, otherwise it is irrelevant.
- Look beyond mere keywords. Assess whether there is a meaningful, semantic connection between the column information and the needs of the question or hint. Mere word matches do not necessarily imply relevance.
- If the question refers to applying a logic on a data such as average, sum, max, min, or any other operation, and the column information is a part of that logic, then the column information is relevant.
- Pay attention to the provided 'Example of values in the column'. If you see a shared keyword between the example and the question or hint, then the column information is relevant. (VERY IMPORTANT)
- If you see the column name appeared in the hint, then it is definitely relevant. (VERY IMPORTANT)
- Note that it does not matter if the question is asking for other information not contained in the column, as long as this column's information is useful for crafting a SQL query answering the question, you should consider this column as relevant.

Column information:
{COLUMN_PROFILE}
Question: {QUESTION}
HINT: {HINT}
Take a deep breath and provide your answer in the following json format:
"'json
{{ "is_column_information_relevant": "Yes" or "No" ,
"chain_of_thought_reasoning": "One line explanation of why or why not the column information is relevant to the question and the hint." }} "'
Only output a json as your response.

Prompt 2: Prompt for Schema Linking