DIRECT: Directional Relevance in Conversational Trajectories

Anshuman Mourya[*]
Amazon
mouryaan@amazon.com

Rajdeep Mukherjee[*]
Amazon
rajdmukh@amazon.com

Prerna Jolly
IIT Hyderabad
prernajolly@alumni.iith.ac.in

Vinayak Puranik
Amazon
puranikv@amazon.com

Sivaramakrishnan Kaveri
Amazon
kavers@amazon.com

Abstract

Conversational agents have become ubiquitous across application domains, such as, shopping assistants, medical diagnosis, autonomous task planning etc. Users interacting with these agents often fail to understand how to start a conversation or what to ask next to obtain the desired information. To enable seamless and hassle-free user-agent interactions, we introduce Next Question Suggestions (NQS), which are essentially highly relevant follow-up question recommendations that act as conversation starters or discover-ability tools to capture non-trivial user intents, leading to more engaging conversations. Relying on LLMs for both response as well as NQS generation is a costly ask in latency-constrained commercial settings, with an added risk of handling potentially unsafe or unanswerable generated queries. A key component of building an efficient low-latency NQS experience is, therefore, *retrieval* (or embedding) models that fetch the most-relevant candidate questions from an offline pre-curated Question Bank (QB). Off-the-shelf embedding models cannot capture domain-specific nuances and more importantly the "directionality" inherent in follow-up question recommendations. In this work, we propose an end-to-end retrieval system, **DIRECT** that is optimized to model directional relevance. Given a user query, it produces a ranked list of highly relevant follow-up question recommendations within 1 sec. Our system also contains an **LLM-as-a-judge** component, tuned on proprietary user-agent interaction logs, to evaluate the end-to-end performance in terms of CTR.

1 Introduction

LLM-based Conversation Agents (or Assistants) have seen a rapid rise in popularity, driven by notable advancements in language generation abilities of modern Large Language Models (LLMs). They are increasingly being deployed across specialized domains such as healthcare, education, public service, etc., and in commercial setups such as e-commerce platforms and customer support to enable more accessible and personalized user interactions (Hu et al., 2024). Despite this growth, several challenges continue to limit the effectiveness of these systems. First, users often find it burdensome to manually type their queries, especially on mobile devices. Second, language barriers, especially in emerging markets, can hinder users, unfamiliar with English or formal written language, to interact with these agents. Third, and most importantly, users often struggle with discover-ability: they don't know what to ask or how to phrase it, which leads to incomplete or dead-end interactions. To address these issues, we introduce the task of Next Question Suggestion (NQS), where the goal is to suggest relevant follow-up questions to a user based on their current query and past conversation history, thereby reducing friction, guiding the conversation deeper, making it more productive, and improving the overall engagement.

Follow-up Question Generation has been widely studied (Meng et al., 2023; Hu et al., 2024), especially in specialized domains such as healthcare (Gatto et al., 2025), social media (Liu et al., 2025), conversation surveys (Ge et al., 2023), etc. However, commercial settings have strict latency and cost constraints, which makes it difficult to rely on LLMs both for response as well as NQS generation. Additionally, one needs to deal with the risk of handling potentially unsafe and/or unanswerable NQSs, given LLM generations are prone to hallucinations. These factors motivate us to model NQS as a low-latency *retrieval* problem, a largely unexplored area which establishes the significance of our study. Given a user query, the task, therefore, is to retrieve a set of highly relevant follow-up queries from an offline QB, and re-rank them based on their contextual relevance, strictly within the

[*]Equal contribution

Table 1: Types of Question Categories for Next Question Suggestion (NQS)

Category	Definition	Example
Similar	The suggestion conveys the same meaning or intent as the user's query, albeit phrased differently.	*Q1: How do I apply for credit card?* *Q2: What are the steps to apply for a credit card*
Follow-up	The suggestion is contextually related but introduces a new or more specific aspect of the topic. It typically follows the user's query in a natural conversational flow.	*Q1: How do I apply for credit card?* *Q2: Which credit card provides best benefit?*
Prior	The suggestion provides background or prerequisite information for understanding the user's query, and would usually precede it in a logical sequence.	*Q1: How do I apply for credit card?* *Q2: What is the purpose of having a credit card?*
Irrelevant	The suggestion is unrelated to the user's query, addressing a completely different topic with no meaningful connection.	*Q1: How do I apply for credit card?* *Q2: What is the capital of USA?*

time it takes for the agent/assistant to generate its response for the given user query.

Our proposed system is an IR framework consisting of a dual encoder-based retriever, and a cross encoder-based re-ranker. We additionally construct an **LLM-as-a-judge** (Li et al., 2024b) component, as part of the framework, for offline evaluation of our end-to-end system. Traditionally, retrievers or embedding models are trained with a binary notion of similarity - classifying pairs as either similar or dissimilar (Xu et al., 2025). However, our task requires a more nuanced understanding, where a relevant NQS should not be similar to the current user query as it introduces redundancy in the conversation flow. We therefore curate a *Proprietary Dataset* consisting of (query, suggestion) pairs labeled as either *similar*, *relevant*, or *irrelevant*. Definitions and examples for each category are reported in Table 1. The cosine similarity of *relevant* pairs is expected to lie between those of *similar* (highest) and *irrelevant* (lowest) pairs. To capture this hierarchy, we propose a two-step hierarchical fine-tuning strategy to train our *retriever* embeddings that effectively learns these boundaries.

Relevant questions can be further categorized into *follow-ups* and *priors*. As shown in Table 1, a follow-up question logically extends the original user query. A prior on the other hand provides background information for the current user query, and hence precedes it. Hence, we must avoid surfacing priors since they are not useful for the users. This introduces an asymmetry that is not found in standard retrieval tasks. This **directional relevance** is captured by our cross encoder while modeling the interaction between a user query, and a suggested candidate. It is fine-tuned on the *follow-up* vs. *prior* classification task to solve two purposes: a) to better detect relevant follow-ups, and b) to rank follow-ups higher than priors.

We name our proposed system **DIRECT**, as it captures the DIrectional RElevance in Conversational Trajectories. Our experimental results are reported on our *Proprietary Dataset*, and an open-source dataset, **FQ-Bank** (Richardson et al., 2023), suitably leveraged for our tasks. For the 3-way classification task, our proposed hierarchical fine-tuning strategy helps us to achieve a relative Precision@Recall improvements of up to 0.08 on our dataset, and up to 0.16 on *FQ-Bank* (Refer Sec. 4.4). For the follow-up vs. prior detection task, we observe Precision@Recall improvements of up to 0.12 on our dataset, and up to 0.1 on *FQ-Bank* (Refer Sec. 4.5). When our end-to-end systems are evaluated using our proposed LLM-as-a-judge framework, DIRECT achieves 6% and 4% CTR improvements over the Vanilla retriever (`BGE-Large` off-the-shelf), with and without the re-ranker respectively (Refer Sec. 4.6).

2 Related Work

Embedding models have become foundational for tasks like question answering, document retrieval, and dialogue systems. Early encoder-based models, such as BERT, RoBERTa, and T5, established strong baselines, while sentence-level adaptations like Sentence-BERT (Reimers and Gurevych, 2019) introduced siamese and triplet architectures to produce semantically meaningful sentence embeddings for efficient similarity search. Sentence-T5 (Ni et al., 2022) extended this to encoder-decoder models, achieving strong transfer performance with encoder-only methods. Recent innovations include proprietary models like Amazon's TitanV1 and TitanV2, along with multilingual models such as BGE-M3 (Chen et al., 2024), supporting over 100 languages. GISTEmbed (Solatorio, 2024) proposed a guided in-batch negative selection framework, leveraging a guide model to

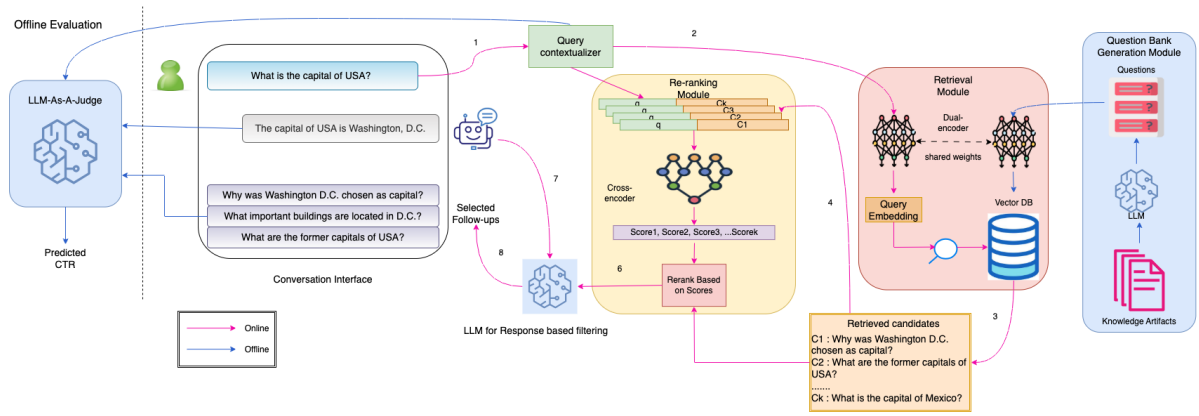

Figure 1: NQS System Architecture : System consists of Question Bank Generation Module, Query Contextualizer, Retriever Module, Re-ranking module, Response-based filtering and LLM-as-a-judge for offline evaluation. Sequence of steps during online call is represented as numbers on the arrows

reduce noise during contrastive training, thus improving embedding quality and enabling efficient fine-tuning of smaller models. Instruction-tuned models, such as INSTRUCTOR (Su et al., 2023) and BGE-ICL (Li et al., 2024a), have been developed to enhance generalization by aligning embeddings with task-specific instructions. More recently, NV-Embed (Lee et al., 2025) introduced a decoder-only architecture enhanced with bidirectional latent attention and contrastive instruction tuning, setting new state-of-the-art benchmarks across several tasks. In our paper, we focus on the capabilities of embedding models for question retrieval, which introduces its own set of challenges which include handling shorter word lengths, requiring a deeper semantic understanding of the domain, and distinguishing between questions that are similar, dissimilar, and those that lie in between as well as directionality of questions. Unlike traditional tasks where models are optimized for semantic similarity, our task involves a more nuanced three-tier classification. We aim to enhance the model's ability to differentiate between these three categories, improving its performance in question retrieval tasks.

3 Methodology

Here, we present our approach for efficient retrieval of highly relevant follow-up question suggestions for a given user query. Our model consists of a dual encoder for efficient retrieval of relevant queries, and a cross encoder for follow-up detection and re-ranking (Fig. 1). First, we pre-train our model on a large-scale proprietary corpus for domain adaptation (Sec. 3.2). Then, the retriever embeddings are trained to distinguish between similar, relevant, and irrelevant queries (Sec. 3.3). Finally, the cross encoder is trained to capture the directional relationship between a (query, suggestion) pair to distinguish between follow-ups and priors. (Sec. 3.4).

3.1 Problem Formulation

Given a conversational context $H = \{S_0, ..S_{j-1}\}$, the current user utterance S_j and an offline question bank $QB = \{Q_1, ...Q_M\}$, the task is to surface N most relevant NQSs to the user. We break down this task into two steps. First, we use our trained dual encoder (retriever) model to obtain the contextualized embedding s_j corresponding to (H, S_j). We use the same model to further obtain the embeddings for all $Q_b \in QB$. Given s_j, we retrieve the top-K candidates $C = \{C_1, ..., C_K\}$ from QB that satisfy certain cosine similarity thresholds. Please note here that the dual encoder model is trained on a sentence pair dataset $D = \{(q_{1i}, q_{2i}, y_i) \forall i \in [1, Z]\}$, where q_{1i} and q_{2i} respectively represent the query and suggested question in the i^{th} example, $y_i \in \{similar, follow-up, relevant, irrelevant\}$ and Z is the number of examples in the training data. In the second step, we train a cross encoder model on dataset D' such that we select examples from D where $y_i \in \{follow-up, prior\}$. The trained cross-encoder model will predict a score r_i for all $C_i \in C$. Finally, we select the top N candidates, based on these scores, to surface as NQSs to the user.

3.2 Domain Adaptation Pre-Training

General-purpose embedding models lack knowledge of specialized domains. In order to understand the nuanced contextual relationship between user

queries and candidate suggestions, possibly containing domain-specific jargon, we adopt BERT's Masked Language Modeling (MLM) technique (Devlin et al., 2019) to pre-train our model on a large-scale proprietary corpus using standard cross-entropy loss. Our innovation lies in the masking strategy adopted. Instead of masking tokens at random, we leverage Named Entity Recognition (NER) to identify entities and terms specific to our domain. Common or uninformative words such as prepositions, articles, and punctuations are filtered out. From the remaining domain-relevant tokens, we randomly select 25% for masking.

3.3 Hierarchical Training of Dual Encoder-based Retriever

For dense retrieval of relevant candidate suggestions for a given user query, we employ a standard dual encoder architecture built on a BERT-based backbone as our retriever. This backbone is first pre-trained for domain adaption. Here, both the query and candidate questions are processed in parallel through identical encoders, with their weights shared. Each input question is tokenized and passed through the encoder. `[CLS]` token output from the last layer gives us a fixed-length embedding representing the semantics of the question. We use *cosine similarity* as the distance metric to measure the similarity between two questions.

Most existing embedding models are trained with a binary notion of similarity—classifying pairs as either similar or dissimilar (Xu et al., 2025). Here, similar pairs are expected to have higher cosine similarity than dissimilar pairs. However, our task requires a more nuanced understanding, where a relevant follow-up suggestion, while sharing meaningful contextual overlap, should not be ideally similar to the question asked by the user. For this, we introduce a three-tier hierarchy of labels: *similar, relevant,* and *irrelevant* for a given user query with the following requirement: $\text{cos\_sim}(\text{similar}) > \text{cos\_sim}(\text{relevant}) > \text{cos\_sim}$ Accordingly, we fine-tune the model using a two-step hierarchical process, which we find to be more effective than a one-step approach where the loss is applied across all three labels simultaneously.

3.3.1 Similar vs Others

In the first step, we perform pairwise contrastive training, with *similar* questions treated as positive samples, and both *relevant* and *irrelevant* questions grouped together as negative samples. Formally, given two embeddings $\mathbf{u}, \mathbf{v} \in \mathbb{R}^d$, and a label $y \in \{0, 1\}$, the Online Contrastive Loss is defined as:

$$\mathcal{L}(\mathbf{u}, \mathbf{v}, y) = \begin{cases} (1 - \text{cos\_sim}(\mathbf{u}, \mathbf{v}))^2, & \text{if } y = 1 \quad \text{(positive pair)} \\ \max(0, m - (1 - \text{cos\_sim}(\mathbf{u}, \mathbf{v})))^2, & \text{if } y = 0 \quad \text{(negative pair)} \end{cases} \tag{1}$$

where cos_sim is the cosine similarity between the two embeddings and m is the margin hyperparameter for negative pairs, ensuring that negative pairs are at least m distance apart. While relevant pairs may share contextual overlap with the user query, we initially group them with irrelevant pairs to establish a strong contrast between truly similar pairs and all others. This allows the model to first tightly cluster highly similar candidates. However, relevant and irrelevant candidates are treated as equally dissimilar, which is not desirable for our setting.

3.3.2 Irrelevant vs Others

To address this, we perform a second round of pairwise contrastive training, using the same loss formulation as in the previous step, with *irrelevant* questions treated as negative samples, and *similar* and *relevant* questions grouped together as positive samples. This step encourages the model to bring relevant and similar candidates closer in the embedding space while pushing highly irrelevant candidates further away. Together, these two steps help our model learn the desired hierarchical structure in the embedding space as noted above.

Relevant queries could be further classified as *follow-ups* or *priors*. Our final goal is to retrieve not just relevant next question suggestions, but relevant *follow-up* suggestions which can potentially take the conversation deeper. Suggesting a query which, although relevant, is a *prior* to the given user query will only add redundancy in the conversation flow, as the latter assumes that the suggested *prior* is already understood by the user. We address this inherent asymmetry in our next task.

3.4 Cross Encoder Training for Follow-up Query Detection

Here, our goal is to distinguish between *follow-ups* and *priors*, and to rank the former higher than the latter. Please note here that while the relationship between a user query and a similar or irrelevant suggestion is symmetric, it is asymmetric when compared to a follow-up (or prior). Formally, $(cos\_sim(user\_query, follow - up) \neq cos\_sim(follow - up, user\_query))$. This directional relationship between a (query, suggestion) pair is not explicitly captured with dual encoders

Table 2: Precision@Recall, PRAUC, and Latency scores for Multiple Models, relative to `BGE-Large`, on the Proprietary Dataset with *similar* class as positive.

Recall	Snowflake L2 (Merrick et al., 2024)	Sentence-T5 Large (Ni et al., 2022)	Cohere_m (Kamalloo et al., 2023)	Instructor XL (Su et al., 2023)	GIST Large (Solatorio, 2024)	BGE-ICL (Li et al., 2024a)
0.5	0.01	0.04	0.01	0.07	-0.01	**0.1**
0.6	0.01	0.05	0.00	0.07	0.02	**0.11**
0.7	0.01	0.02	-0.01	0.05	-0.02	**0.11**
AUC	0.00	0.03	0.00	0.03	-0.01	**0.08**
Latency p90 (ms)	4.5	2.5	61.5	37.5	2.0	161.5

since the user query and candidate suggestion are processed in parallel. To capture their interaction, we employ a standard cross encoder architecture built on top of a BERT-based backbone. Similar to the dual encoder, we first pre-train this backbone, as detailed in Sec. 3.2, for domain adaption. We then train the model using *binary cross-entropy* loss for the task of classifying relevant suggestions into *follow-ups* and *priors*. A joint representation is learnt in the process that effectively captures the positional and directional context between the user query and suggested candidate. Learning this asymmetry not only helps the cross encoder to better distinguish between follow-up and prior candidates, but also generate an inherent ranking between the two sets with follow-ups ranked higher than priors.

4 Experiments and Results

4.1 Datasets

We perform our experiments on two datasets: a proprietary dataset and an open-source dataset modified for our task as detailed below:

Proprietary Dataset We source our data from multiple customer-facing resources, including help pages, video transcripts, blogs, and logs from our proprietary Customer Assistant chat bot. The latter includes customer queries, responses generated by the assistant, and suggested next questions. To construct a robust and diverse **Question Bank** (QB) to be used for NQS retrieval, we leverage *Anthropic*'s *Claude Sonnet 3.5*[1] to generate a wide range of around 90K potential customer questions grounded on the customer-facing resources mentioned above.

Our fine-tuning dataset consists of *(user query, suggested question)* pairs, with the former sourced from actual user interactions, and the latter sampled from the QB. Each pair is annotated by human reviewers and assigned one of the four categories: *similar*, *follow-up*, *prior*, and *irrelevant* (Refer to Table 1 for examples). For the hierarchical 3-way classification task, *follow-ups* and *priors* are grouped together under the *relevant* category, reflecting their contextual but directional relationship with the user query. Refer to Sec. A.1 for statistics.

Follow-up Query Bank (FQ-Bank) Proposed by Richardson et al. (2023), this dataset contains $\approx$ 14K multi-turn conversations, with each turn-level instance associated with a set of follow-up suggestions, out of which only one is *valid* and others are *invalid* candidates. In its original form, this dataset is only suited for the ranking task with an objective to rank the valid candidate higher than the invalid ones from a given set of questions. In order to make it suitable for our retrieval task, we apply some rules to map the *invalid* samples as either *similar*, *prior*, or *irrelevant*. Mapping rules and dataset statistics are reported in Sec. A.1.

4.2 Embedding Model Selection

We considered several top performing models (at the time of experiments) from the MTEB Leaderboard[2] and compared their off-the-shelf performance on the test set used for our hierarchical three-way classification experiments. This evaluation is, however, formulated as a binary classification task by designating one category as the positive class while combining the remaining categories as the negative class. For each model, we quantify the relationship between a given (query, suggestion) pair in terms of cosine similarity between their embeddings. Pairs exceeding a pre-defined upper similarity threshold were classified as similar, those below a lower threshold as irrelevant, and intermediate values as relevant. We compare the binary classification performance of all baselines by reporting the area under the Precision-Recall curve (**PRAUC**), and Precision@Recall scores relative to `BGE-Large` in Table 2 where *similar* class is considered as positive. **Precision@Recall** is defined

[1]https://www.anthropic.com/news/claude-3-5-sonnet

[2]https://huggingface.co/spaces/mteb/leaderboard

Table 3: Precision@Recall and PRAUC for fine-tuned BGE-Large variants for the 3-way classification task (*similar*, *relevant*, and *irrelevant*). Scores are reported for different binary classification scenarios. All the scores are reported relative to the Domain adapted pre-trained BGE-large

	On Proprietary Dataset				**On FQB Dataset**			
Recall	**Similar Vs. Others**		**Irrelevant Vs. Others**		**Similar Vs. Others**		**Irrelevant Vs. Others**	
	One-step	Hierarchical	One-step	Hierarchical	One-step	Hierarchical	One-step	Hierarchical
0.5	0.01	**0.08**	0.02	**0.06**	0.01	**0.03**	0.03	**0.13**
0.6	0.03	**0.08**	0.01	**0.04**	0.03	**0.04**	0.04	**0.16**
0.7	0.01	**0.05**	0.00	**0.04**	0.02	**0.05**	0.06	**0.16**
PRAUC	0.02	**0.07**	0.02	**0.05**	0.04	**0.07**	0.04	**0.10**

as the precision achieved when recall reaches a specific level, 0.5, 0.6 and 0.7, in our experiments. Relative `p90` latency of models are also reported. We selected `BGE-Large` as our backbone for subsequent experiments since it has the lowest latency with competitive accuracy values.

4.3 Pre-Training Experiments

We construct a pretraining dataset using customer help pages that were not included in our fine-tuning dataset. Document contents were segmented into chunks of 512 tokens, and 25% of them were randomly masked (refer Sec. 3.2). We then pretrain our `bge` models using the MLM objective for the task of predicting masked tokens. We trained `bge-large` for 5 epochs with a batch size of 8; `bge-small` and `bge-base` were trained for 8 epochs with a batch size of 32. For all models, 1×10^{-5} was used as the learning rate. Following Devlin et al. (2019), we compute *Accuracy* as the proportion of correctly predicted masked tokens, and observe **52%, 54%,** and **60%** improvements respectively for `bge-small`, `bge-base`, and `bge-large`, highlighting the efficacy of our approach for domain adaptation.

4.4 Hierarchical 3-way Classification

For both the fine-tuning steps, first to segregate *similar* samples from others, and next to segregate *irrelevant* samples from others (refer Sec. 4.4 for details), we hierarchically train `BGE-large` for 5 epochs with a batch size of 16 and a learning rate of 1×10^{-5}. We used similar settings while training the models respectively on the two datasets, *Proprietary* and FQB. Precision@Recall and PRAUC scores for our fine-tuned `BGE-Large` variants, relative to the domain adapted pre-trained BGE-Large model, are reported in Table 3 under two different binary classification scenarios, one where *similar* class is treated as positive while *relevant* and *irrelevant* classes are grouped as negative, and the other where the *irrelevant* class is negative, while *similar* and *relevant* classes are grouped as positive.

In Table 3, *One-step* represents the model trained in a single step jointly on all three categories using the *CoSENT* loss that maps cosine similarity values to class labels: 1 for similar, 0.5 for relevant, and 0 for irrelevant. *Hierarchical* represents the model trained using our proposed hierarchical fine-tuning strategy. *Hierarchical* consistently outperforms *One-step* (and hence pre-trained BGE-Large as well) on both datasets. Comparing with Table 2 on our *Proprietary* dataset, the PRAUC scores of the hierarchically fine-tuned `BGE-large` (330M) notably surpass larger models like `Instructor-XL` (1.5B) and are at par with `BGE-ICL` (7.1B).

4.5 Follow-up Detection

We train our cross-encoder on the follow-up vs. prior classification task. We experiment with two configurations, one without data augmentation and other where we augment the training data with reverse pairs, i.e. for a sentence pair $(x, y) = ((S_1, S_2), q)$ for $q \in \{0, 1\}$, with follow-up being the positive class, we augment a pair $(x', y') = ((S_2, S_1), 1 - q)$ to the dataset. Classification performance improves by +4% and +2% in terms of auPRC and auROC respectively while training with augmentation on our Proprietary dataset. We do not, however, observe any significant improvement with augmentation on the FQ-Bank dataset.

4.5.1 Threshold Detection

Once we have trained both the dual encoder as well as the cross encoder, we evaluate the performance after each stage on the (binary) follow-up detection task. For identifying the relevant follow-ups using our fine-tuned dual encoder model, first we compute the cosine similarities for the sentence pairs

Table 4: Best Precision@Recall scores with our proposed model variants on the Follow-up Detection task. All scores are relative to Vanilla (`BGE-Large`) retriever

Model	On the Proprietary Dataset			On the FQB Dataset		
Recall	0.5	0.6	0.7	0.5	0.6	0.7
DIRECT w/o re-ranker	0.081	0.076	0.071	0.024	0.030	0.016
DIRECT	**0.112**	**0.108**	**0.0.115**	**0.104**	**0.097**	**0.069**

Table 5: CTR Prediction using LLM-As-a-Judge Framework for our End-to-End model variants. All the scores are reported relative to Vanilla retriever model

Model	Predicted CTR	Latency p90 (ms)
DIRECT wo re-ranker	4%	1.30
DIRECT	**6%**	27.23

in our validation set and tune a pair of thresholds $t_{retrieval} = (t_{low}, t_{high})$, such that all the sentence pairs with cosine similarities between t_1 and t_2 are classified as *Follow-ups*, while others (with actual labels as either similar, irrelevant or prior) are classified as *Non Follow-ups*. We tune these thresholds to optimize for Precision@Recall, by identifying the threshold pair that provides the best precision at a given recall value. Once we have selected the threshold pairs using this process, we compute the final binary classification metrics on the test set.

4.5.2 Evaluation

We compare the follow-up detection performance of our proposed systems, DIRECT w/o re-ranker, and DIRECT (with re-ranker) in Table 4 by reporting the Precision@Recall scores relative to Vanilla (BGE-Large) retriever model. Please note that it is difficult to directly compare the two models, as the re-ranker is trained to classify between follow-ups and priors, whereas the dual encoder (retriever) was originally trained on the 3-way (similar/relevant/irrelevant) classification task. In Sec. A.2, we present our methodology to ensure a fair comparison between the two. From Table 4, we find that DIRECT, consisting of a hierarchically trained dual encoder, and a cross encoder (re-ranker) trained on priors as the hard-negatives, is the best performing model on both datasets.

4.6 LLM-As-a-Judge: End-to-End Evaluation

Given that we are still in the process of deploying DIRECT to production, we constructed an LLM-as-a-judge framework by employing *Claude Sonnet 4.5*[3] to evaluate the end-to-end performance of our model variants in terms of click through rate (CTR). This gives us a scalable offline mechanism to estimate the relative user engagement potential across models. We developed the prompt with good quality ICL (In Context Learning) exemplars obtained from our user-chat bot interactions that were not included in our Proprietary dataset. Each exemplar contains the conversation history, the current user query, the response generated by the chat assistant, the suggested follow-up candidates, and the customer click information (it is possible that none of the options were clicked if the user did not find the suggestions relevant enough).

Please refer to Sec. A.3 for details on how the flow varies between our system variants, DIRECT, and DIRECT w/o re-ranker, for obtaining the final set of NQSs from the offline QB given a user query. We report the comparative performance of our system variants in Table 5, and observe the highest CTR for DIRECT. Although the latency is expectedly more due to the re-ranking step, it is still within our acceptable limits of 1 second.

5 Conclusion

In this work, we address a novel challenge of recommending highly relevant follow-up question suggestions, unique to conversational experiences. High quality follow-up suggestions help in reducing dead-end conversations, making the experience seamless and more effective. Our research introduces a novel three-tier hierarchical fine-tuning methodology addressing a fundamental challenge in question sequence modelling. The experimental results on two datasets from different domains (a proprietary conversational dataset and public dataset of follow-up questions) comprehensively demonstrate the efficacy of our underlying methods – hierarchical fine-tuning and a separate cross-encoder based re-ranker. We also demonstrate that pre-training of embedding models on domain-specific data is highly effective in understanding nuanced terminologies and jargons to better establish contextual relationship between queries and questions. Through a combination of techniques we offer a robust and low-latency solution for a non-trivial problem of modelling temporal and logical relationships between question sequences. Our work aims to improve end-user metrics such as relevance and click-through rate for Next Question Suggestions in commercial conversational assistants, thus establishing relevance to a highly common industrial NLP application.

[3]https://www.anthropic.com/news/claude-sonnet-4-5

Limitations

Limitations of our work are as follows:

- *Reliance on annotation data*: Effectiveness of our techniques such as hierarchical fine-tuning and cross-encoder training hinges on the availability and quality of annotated data points. Further, low frequency examples belonging to follow-up or prior classes are expected to be well represented in the annotated data. While in the future work we plan to explore LLM-driven auto-labeling capabilities to generate labeled data automatically, currently variability in the label quality or coverage may impact the effectiveness of our approach to model the directionality.

- *Language adaptation*: We demonstrate our techniques by focusing on English language datasets. Support for non-English languages not only requires data availability to train the models, but further experimentation to support the extensibility of our technique to these languages. As conversation experiences become more widespread in their applications, the need to support local language of preferences gains importance.

- *Real-time generation*: While dynamic generation of follow-up questions in real-time using LLMs can lead to more naturally flowing conversational trajectories, we currently restrict our work to the approach of retrieval from a pre-curated question bank. This approach is also motivated by the fact that real-time generation risks hallucinations and demonstrates a significant increase in latency. Our future work explores leveraging real-time generations more effectively.

References

Jianlyu Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. 2024. M3-embedding: Multi-linguality, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 2318–2335, Bangkok, Thailand. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Joseph Gatto, Parker Seegmiller, Timothy E. Burdick, Inas S. Khayal, Sarah DeLozier, and Sarah M. Preum. 2025. Follow-up question generation for enhanced patient-provider conversations. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 25222–25240, Vienna, Austria. Association for Computational Linguistics.

Yubin Ge, Ziang Xiao, Jana Diesner, Heng Ji, Karrie Karahalios, and Hari Sundaram. 2023. What should I ask: A knowledge-driven approach for follow-up questions generation in conversational surveys. In *Proceedings of the 37th Pacific Asia Conference on Language, Information and Computation*, pages 113–124, Hong Kong, China. Association for Computational Linguistics.

Jiaxiong Hu, Jingya Guo, Ningjing Tang, Xiaojuan Ma, Yuan Yao, Changyuan Yang, and Yingqing Xu. 2024. Designing the conversational agent: Asking follow-up questions for information elicitation. *Proc. ACM Hum.-Comput. Interact.*, 8(CSCW1).

Ehsan Kamalloo, Xinyu Zhang, Odunayo Ogundepo, Nandan Thakur, David Alfonso-Hermelo, Mehdi Rezagholizadeh, and Jimmy Lin. 2023. Evaluating embedding apis for information retrieval. *Preprint*, arXiv:2305.06300.

Chankyu Lee, Rajarshi Roy, Mengyao Xu, Jonathan Raiman, Mohammad Shoeybi, Bryan Catanzaro, and Wei Ping. 2025. Nv-embed: Improved techniques for training llms as generalist embedding models. *Preprint*, arXiv:2405.17428.

Chaofan Li, MingHao Qin, Shitao Xiao, Jianlyu Chen, Kun Luo, Yingxia Shao, Defu Lian, and Zheng Liu. 2024a. Making text embedders few-shot learners. *Preprint*, arXiv:2409.15700.

Haitao Li, Qian Dong, Junjie Chen, Huixue Su, Yujia Zhou, Qingyao Ai, Ziyi Ye, and Yiqun Liu. 2024b. Llms-as-judges: A comprehensive survey on llm-based evaluation methods. *Preprint*, arXiv:2412.05579.

Jianyu Liu, Yi Huang, Sheng Bi, Junlan Feng, and Guilin Qi. 2025. From superficial to deep: Integrating external knowledge for follow-up question generation using knowledge graph and LLM. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 828–840, Abu Dhabi, UAE. Association for Computational Linguistics.

Yan Meng, Liangming Pan, Yixin Cao, and Min-Yen Kan. 2023. FollowupQG: Towards information-seeking follow-up question generation. 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), pages 252–271, Nusa Dua, Bali. Association for Computational Linguistics.

Luke Merrick, Danmei Xu, Gaurav Nuti, and Daniel Campos. 2024. Arctic-embed: Scalable, efficient, and accurate text embedding models. *arXiv preprint arXiv:2405.05374*.

Jianmo Ni, Gustavo Hernandez Abrego, Noah Constant, Ji Ma, Keith Hall, Daniel Cer, and Yinfei Yang. 2022. Sentence-t5: Scalable sentence encoders from pretrained text-to-text models. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 1864–1874, Dublin, Ireland. Association for Computational Linguistics.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Christopher Richardson, Sudipta Kar, Anjishnu Kumar, Anand Ramachandran, Zeynab Raeesy, Omar Khan, and Abhinav Sethy. 2023. Learning to retrieve engaging follow-up queries. In *Findings of the Association for Computational Linguistics: EACL 2023*, pages 2009–2016, Dubrovnik, Croatia. Association for Computational Linguistics.

Aivin V. Solatorio. 2024. Gistembed: Guided in-sample selection of training negatives for text embedding fine-tuning. *Preprint*, arXiv:2402.16829.

Hongjin Su, Weijia Shi, Jungo Kasai, Yizhong Wang, Yushi Hu, Mari Ostendorf, Wen-tau Yih, Noah A. Smith, Luke Zettlemoyer, and Tao Yu. 2023. One embedder, any task: Instruction-finetuned text embeddings. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 1102–1121, Toronto, Canada. Association for Computational Linguistics.

Zhichao Xu, Fengran Mo, Zhiqi Huang, Crystina Zhang, Puxuan Yu, Bei Wang, Jimmy Lin, and Vivek Srikumar. 2025. A survey of model architectures in information retrieval. *Preprint*, arXiv:2502.14822.

A Appendix

A.1 Dataset Statistics

We perform our experiments on two datasets: a proprietary dataset and an open-source dataset modified for our task as detailed below:

Proprietary Dataset We source our data from multiple customer-facing resources, including help pages, video transcripts, blogs, and logs from our proprietary Customer Assistant chat bot. The latter includes customer queries, responses generated by the assistant, and suggested next questions. To construct a robust and diverse **Question Bank** (QB) to be used for NQS retrieval, we leverage *Anthropic*'s *Claude Sonnet 3.5*[4] to generate a wide range of around 90K potential customer questions grounded on the customer-facing resources mentioned above.

Our fine-tuning dataset consists of *(user query, suggested question)* pairs, with the former sourced from actual user interactions, and the latter sampled from the QB. Each pair is annotated by human reviewers and assigned one of the four categories: *similar*, *follow-up*, *prior*, and *irrelevant* (Refer to Table 1 for examples). For the hierarchical 3-way classification task, *follow-ups* and *priors* are grouped together under the *relevant* category, reflecting their contextual but directional relationship with the user query. We obtain a set of total 5000 data points for training with the following distribution: 1460 *similar* samples, 1670 *relevant* samples (with 980 *follow-ups* and 690 *prior*), and 1870 *irrelevant* samples. For tuning and testing our models, we constructed a random validation and test sets from 8613 manually annotated question pairs: 4542 similar, 113 prior, 2090 follow-up, and 1868 irrelevant samples.

Follow-up Query Bank (FQ-Bank) Proposed by Richardson et al. (2023), this dataset contains ≈ 14K multi-turn conversations, with each turn-level instance associated with a set of follow-up suggestions, out of which only one is *valid* and others are *invalid* candidates. In its original form, this dataset is only suited for the ranking task with an objective to rank the valid candidate higher than the invalid ones from a given set of questions. In order to make it suitable for our retrieval task, first we tag the *valid* suggestions as *follow-ups*. We then apply the following rules to map the *invalid* suggestions as either *similar*, *prior*, or *irrelevant*: Suggestions with labels *irrelevant_entity*, and *irrelevant_question* are mapped to the *irrelevant* class, *paraphrase* is mapped to *similar*, and those tagged as *present_in_context* are mapped to *prior*. We used the exact same data split as proposed in Richardson et al. (2023).

[4]https://www.anthropic.com/news/claude-3-5-sonnet

A.2 Evaluating Follow-up Detection

We compare the follow-up detection performance of our proposed systems, DIRECT w/o re-ranker, and DIRECT (with re-ranker) in Table 4 by reporting the Precision@Recall scores relative to Vanilla (BGE-Large) retriever model. Please note that it is difficult to directly compare the two models, as the re-ranker is trained to classify between follow-ups and priors, whereas the dual encoder (retriever) was originally trained on the 3-way (similar/relevant/irrelevant) classification task. To ensure a fair comparison between the two models, we perform the following steps. First, we obtain the similarity threshold pair $t_{retrieval}$ from the previous step (refer Sec. 4.5.1) that was optimized for precision at recall k. Let the set of predicted follow-ups with this configuration be denoted as F_k. Next, we compute the cross encoder scores on the set F_k and tune another threshold t_{rerank} such that the recall on this set is 1 and the precision is maximized. Once both $t_{retrieval}$ and t_{rerank} are decided, we compute the Precision@Recall-k for different values of k and report them in Table 4. We find that DIRECT, consisting of a hierarchically trained dual encoder, and a cross encoder (re-ranker) trained on priors as the hard-negatives, is the best performing model on both datasets.

A.3 LLM-As-a-Judge: End-to-End Evaluation

We construct an LLM-as-a-judge framework by employing *Claude Sonnet 4.5*[5] to evaluate the end-to-end performance of our model variants in terms of click through rate (CTR). We developed the prompt with good quality ICL (In Context Learning) exemplars obtained from our user-chat bot interactions that were not included in our Proprietary dataset. Each exemplar contains the conversation history, the current user query, the response generated by the chat assistant, the suggested follow-up candidates, and the customer click information (it is possible that none of the options were clicked if the user did not find the suggestions relevant enough).

We randomly sample 500 user queries from our logs (not part of the Proprietary dataset) to create an evaluation set. For each embedding model (`BGE-Large`) variant reported in Table 5, we first construct a FAISS index on our QB. For each query, we search the index to retrieve the top 50 next question suggestions. We then apply the similarity thresholds tuned earlier (refer Sec. 4.5) to obtain the probable follow-up candidates. For the first two models (Table 5), we select the top 15 candidates based on cosine similarity scores. For the last one, we compute the cross encoder predictions and re-rank the candidates before selecting the top 15. These candidates are then passed through an LLM (*Claude Haiku 3.5*) call to filter out the ones already answered in the chat assistant response to the user query. We select a maximum of 3 suggestions from this filtered set and pass it through our proposed LLM-as-a-judge framework to calculate the CTR. We observe the highest CTR for DIRECT. Although the latency is expectedly more due to the re-ranking step, it is still within our acceptable limits of 1 second.

[5]https://www.anthropic.com/news/claude-sonnet-4-5