Efficient and Effective Conversational Search with Tail Entity Selection

Hai Dang Tran$^{1[0009-0002-8548-064X]}$, Andrew Yates$^{1,2,3[0000-0002-5970-880X]}$, and Gerhard Weikum$^{1[0000-0003-4959-6098]}$

^{1} Max Planck Institute for Informatics, Saarbrucken, Germany
`{hatran,weikum}@mpi-inf.mpg.de`
^{2} University of Amsterdam, Amsterdam, Netherlands
^{3} HLTCOE, Johns Hopkins University, Baltimore, Maryland, USA
`andrew.yates@jhu.edu`

Abstract. Questions in conversations can be highly informal, with the user intent becoming clear only with prior context. This is challenging when the conversation involves long-tail entities. Prior works addressed these issues with computationally expensive techniques; this work aims to improve efficiency while being competitive in effectiveness. To this end, we devise techniques to efficiently compute relatedness scores between questions and entity mentions in previous turns, and select a few mentions for question expansion. Answers are computed by a ranking pipeline, with candidate subsets of decreasing size as later stages incur higher cost. Experiments with two benchmarks show that our method outperforms all baselines on both efficiency and effectiveness.

1 Introduction

Motivation. Conversational IR (CIR) gives answers to questions in a multi-turn session [39]. Here, it is crucial to combine questions with context cues from previous turns. Questions can be about emerging entities, which may not exist in Wikipedia. A system answer would be an informative passage from news articles.

Consider the conversation on the left side of Figure 1; the example illustrates two problems. First, follow-up questions Q2 through Q4 are all but self-contained, and need to be *contextualized* with previous turns. Second, the conversation involves *long tail* entities. In Q4, “her team” refers to Annett Kaufmann and SV Boeblingen. However, Annett Kaufmann is not featured in Wikipedia, and “Boeblingen” would likely be a town rather than the sports club. Thus, entity linking is not an option in this case.

Limitations of Prior Works. Contextualizing a question from previous turns [4, 15, 21, 33] can rewrite the user input into a self-contained question, or expand the input with selected cues from the conversation history, most importantly, relevant entities and background from Wikipedia.

For coping with tail entities out of Wikipedia, the CONSENT method [33] leverages the relatedness between the question and previous mentions of named entities based on SBERT [29]. However, SBERT vectors are agnostic to the specific task of relating mentions to informal questions. Also, prior methods are computationally expensive due to making extensive use of techniques like cross-encoders, ILP or GNNs. The goal of this paper is to improve the *efficiency* of CIR with *long-tail entities*, while being competitive in *effectiveness*.

Approach. Our method, EECATS for Efficient and Effective Conversational search with tAil enTity Selection, reconciles efficiency and effectiveness. We adopt the idea of expanding questions with mentions from [33], but devise very different algorithms with much lower computational cost. EECATS has two stages:

1. **Contextualization:** For each user input, EECATS selects a few top relevant mentions of named entities from prior turns. We judiciously fine-tune an SBERT model for this task of question-mention relatedness. The selected mentions are added to the expanded question.
2. **Answer ranking:** EECATS feeds the expanded question to a search pipeline of BM25, ColBERT [31] and a cross-encoder [25], where each stage processes a decreasing number of answer candidates to tame the increasing costs. A key point is to avoid repeated computation of embeddings by using a fast embedding aggregation technique and an in-conversation cache.

Contributions. The salient contributions of this work are the following. 1) We propose a new way of contextualizing, by *fine-tuning* SBERT to the specific task of selecting relevant entities as informative cues. 2) We devise techniques for reducing the computational costs, including *efficient aggregation* of judiciously computed and cached embeddings. 3) Experiments show that EECATS outperforms the best prior baselines. For questions about tail entities, EECATS is significantly better in nDCG@3 than the best GPT competitor. For efficiency, EECATS improves latency by a factor of 37x.

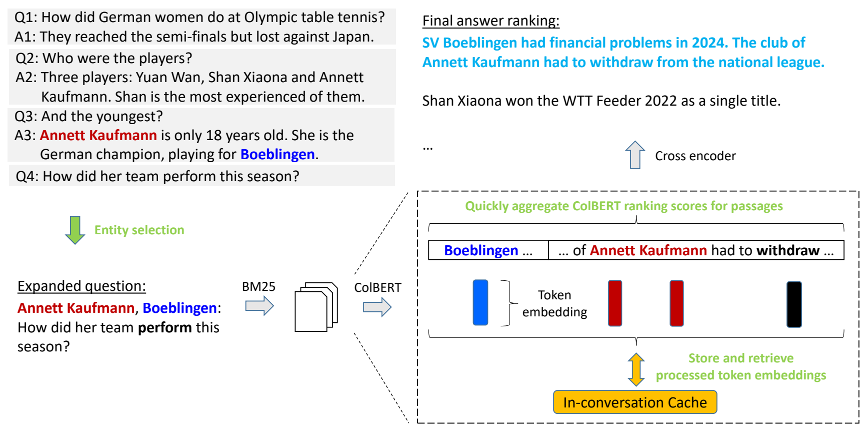
Q1: How did German women do at Olympic table tennis?
A1: They reached the semi-finals but lost against Japan.
Q2: Who were the players?
A2: Three players: Yuan Wan, Shan Xiaona and Annett Kaufmann. Shan is the most experienced of them.
Q3: And the youngest?
A3: Annett Kaufmann is only 18 years old. She is the German champion, playing for Boeblingen.
Q4: How did her team perform this season?
Entity selection
Expanded question:
Annett Kaufmann, Boeblingen: How did her team perform this season?
BM25
ColBERT
Final answer ranking:
SV Boeblingen had financial problems in 2024. The club of Annett Kaufmann had to withdraw from the national league.
Shan Xiaona won the WTT Feeder 2022 as a single title.
...
Cross encoder
Quickly aggregate ColBERT ranking scores for passages
Boeblingen ...
... of Annett Kaufmann had to withdraw ...
Token embedding
Store and retrieve processed token embeddings
In-conversation Cache

Fig. 1: Overview of the EECATS System Architecture

2 EECATS Methodology

Given a user question Q_i with prior conversation history $H = \{(Q_1, A_1), (Q_2, A_2), ..., (Q_{i-1}, A_{i-1})\}$, we aim to find the most relevant passage-level answers to Q_i from a collection of news and Wikipedia articles. Figure 1 gives an architecture overview of EECATS. As we focus on tail entities, we identify entity mentions in prior turns $(Q_j, A_j)(1 \leq j < i)$ in two categories: *i) in-KB entities* that can be linked to Wikipedia with high confidence, and *ii) out-of-KB entities* that either are not featured in Wikipedia at all or cannot be disambiguated with confidence.

EECATS selects the top most-relevant mentions and adds them to an expanded question. Later, EECATS runs the expanded question through a judiciously designed ranking pipeline of BM25, ColBERT and a cross-encoder. We need to compute a large amount of embeddings using ColBERT. To mitigate this cost, our design is to lower the GPU consumption, including light-weight aggregation of sentence-level embeddings, in-conversation caching and reuse of embeddings.

2.1 Relevant Entity Selection

At the i-th turn, we need to quantify the relatedness between the current question Q_i and each entity mention e in the history using the following concepts.

For entity e in the j-th turn of the history, the entity context $Con(e)$ is the j-th turn (i.e., the concatenation of the j-th question and the respective answer) if e is out-of-KB. If e is in-KB, $Con(e)$ is concatenation of first Wikipedia paragraph of e and turn j. We consider the in-KB first Wikipedia paragraph as a valuable background knowledge for entity selection (e.g., *"Shan Xiaona is a naturalised German table tennis player..."*). The turn text where the entity is mentioned can also provide strong cues like semantic types. E.g., in turn 3 of the example in Figure 1, we could infer that *"Boeblingen"* refers to a sports team, which is useful for interpreting Q_4. The knowledge representation of e is $Rep^K(e) = SBERT(e\ [SEP]\ Con(e))$, encoding knowledge about the entity.

To capture the flow of information need, we need to compute $Rep^F(e) = SBERT(Q_j\ Q_{j+1}\ ...\ Q_{i-1})$. The relatedness between e and Q_i is the inner product: $Rel(e, Q_i) = \langle Rep^K(e) + Rep^F(e), SBERT(Q_i)\rangle$. The top-$L$ most related mentions are prefixed to the current question, forming the expanded question that is fed into the retrieval stages. Our design captures both static background about entities and dynamic cues from the conversation for entity selection.

Training. The above computations could be carried out with standard SBERT, but this would be based on semantic similarity of sentences in general, whereas our goal is to capture relatedness with entity mentions. Therefore, we fine-tune SBERT for our specific task based on the following procedure for training data.

We use an LLM to automatically complete a training set of questions (e.g. *Q4*: How did the Boeblingen team of Annett Kaufmann perform this season?* of *Q4* in Figure 1), and use these as ground-truth annotations. We treat all mentions that appear in the completed question as being relevant for the interpretation of the original question, and others as irrelevant ones. With this approach, let E^+

and E^- denote positive entities and negative entities to a given question Q_i. We can fine-tune SBERT such that $Rel(e^+, Q_i)$ higher than $Rel(e^-, Q_i)$ for every pair (e^+, e^-) where $e^+ \in E^+$ and $e^- \in E^-$. We use the pairwise ranking loss [2] to train for entity selection, starting from initial SBERT parameters loaded from Hugging Face[4]. In total, we used $10,940$ training pairs (e^+, e^-). Note that this is a training-time method only. At inference time, no information about relevant entities is available upfront.

2.2 Passage Ranking

We feed the expanded question into a search pipeline. First, we use BM25 [30] to retrieve top-100 documents, and all passages in these documents are candidates to be processed in next ranking stages. Multiple passages in the same document could overlap on the same sentences. A bi-encoder computes representation vectors for the tokens in the expanded question and each candidate passage separately, aggregates these in a judicious manner, and computes scores for ranking passages. We use ColBERT [31] as a powerful choice for the bi-encoder.

The expanded question token representation $Rep^{EQW}(w)$ of a token w in the expanded question q' is the embedding of w when we feed q' through the bi-encoder. For the passages, we start with encoding tokens within their sentences. Later we aggregate their embeddings for ranking longer passages. For token w in a sentence s, the sentence token representation $Rep^{SW}(w)$ of w is the embedding of w when we feed s through the bi-encoder.

Since questions in a conversation are related to each other, it is likely that various sentences of the top documents from BM25 are repeated in different turns. We store sentence token representations in an in-conversation cache, and reuse them when a sentence re-appears in candidate passages of a future turn. This caching opportunity is another factor in our design for encoding sentences rather than entire passage, as the smaller granularity increases the chance of cache hits.

With the independently computed question token representations and sentence token representations, we can now define how these are aggregated into vectors at the passage level for scoring and ranking. For expanded question q' and sentence S with respective token sequences $w_1, w_2, ..., w_H$ and $v_1, v_2, ..., v_G$; the maximal similarity vector of S against q' is calculated as $Rep^S(S, q') = \{r_1, r_2, ..., r_H\}$ where $r_k = \max_{j=1}^{G} Cos(Rep^{EQW}(w_k), Rep^{SW}(v_j))(1 \leq k \leq H)$ and Cos is the cosine similarity. For passage A with consecutive sentences $S_1, S_2, ...,$ S_Z, the maximal similarity vector $Rep^P(A, q')$ of A against q' is $Rep^P(A, q') = \max_{k=1}^{Z} Rep^S(S_k, q')$ where max is element-wise maximum operator.

This is a light-weight technique, which allows us to efficiently compute maximal similarity vectors for passages of various lengths (with high amount of overlapping) by aggregating from the smaller granularity of their sentences. Consider a passage A with maximal similarity vector $Rep^P(A, q') = \{r'_1, r'_2, ..., r'_H\}$, the bi-encoder ranking score of A is $\sum_{k=1}^{H} r'_k$.

[4] https://huggingface.co/sentence-transformers/all-mpnet-base-v2

We employ a computationally heavier cross-encoder for the shortlist of top-100 candidates from prior steps. For each passage, we compute its relevance score against q' using cross-encoder [5] [25], this is the final ranking score of the passage.

3 Evaluation

3.1 Experimental Setup

We use two benchmarks: (i) a new dataset, CATS, focusing on tail entities, and (ii) the TREC CAsT 2019 and 2020 datasets [6, 7] with popular entities. We measure average nDCG@3, query latency and Tera FLOPS [6].

For the CATS dataset, we adapt the automatic benchmark construction from CONSENT [33] for passages as answers. Each question is assigned to one of three groups: out-of-KB, rare-in-KB, and uncommon-in-KB, based on popularity of the entity that the question centers on. CATS benchmark consists of a training set with 840 questions and a test set of 436 questions. The respective collection is news articles from 2018 [16] and Wikipedia articles from 2017. To judge system answers, we adapt the automatic judgment approach proposed by CONSENT [33], built on top of GPT-4 [27]. In CAsT, we use a collection of MS MARCO [5] and TREC CAR [8]; and CANARD [9] as a trainset. We use gold answers in the history, to ensure the history is identical for all methods.

We use the following baselines. *i) ConvDR*: This method [38] is a conversational dense retriever, obtaining the relevance score as the inner product of question and answer representations. *ii) CoSPLADE + T5*: CoSPLADE [15] extracts salient terms in the history for sparse retrieval, and monoT5 [26] is used as re-ranker. *iii) GPT-3.5*: GPT-3.5 Turbo [1] is asked whether a passage is relevant to the question under the conversation history. The probability of "yes" token is the final ranking score. *iv) CQR GPT-3.5*: This method uses GPT 3.5 Turbo to rewrite conversation questions into self-contained ones, which are issued also to GPT 3.5 Turbo (using probability of "yes" as in *GPT-3.5* for relevance score). *v) GPT-4o Anon*: This is the same as *GPT-3.5* baseline, except that we use GPT-4o [27] and consistently replace names in the test with anonymous names. *vi) CONSENT*: This is the cutting-edge CIR method for tail entities [33], which jointly contextualizes and ranks. For CoSPLADE + T5 and CONSENT and EECATS, we use CATS trainset for CATS test and CANARD for CAsT test.

3.2 Experimental Results

Effectiveness. Table 1 reports the results for the CATS and CAsT benchmarks. On almost all groups of CATS test, our method significantly outperforms the state-of-the-art method CONSENT with nDCG@3 being around 9 percentage points higher for All group, and even higher for the Out-of-KB and Rare groups. In nDCG@3, EECATS is significantly better than all GPT methods on All

[5] https://huggingface.co/cross-encoder/ms-marco-MiniLM-L-6-v2
[6] https://huggingface.co/spaces/MrYXJ/calculate-model-flops

Pipelines	**nDCG (CATS)**				**nDCG (CAsT)**		**Latency**	**TFLOPS**
	All	Out-of-KB	Rare	Uncommon	19	20	(seconds)	
ConvDR	0.064	0.049	0.068	0.075	0.466	0.340	**0.15**	**0.5**
CoSPLADE + T5	0.201	0.224	0.118	0.303	0.478	0.368	51.33	573.0
CONSENT	0.276	0.264	0.203	0.407	0.457	0.334	96.95	247.9
CQR GPT-3.5	0.279	0.310	0.204	0.358	-	-	66.96	-
GPT-3.5	0.303	0.311	0.218	0.426	-	-	50.18	-
GPT-4o Anon	0.311	-	-	-	-	-	101.52	-
EECATS	**0.363**†‡	0.358†	**0.335**†‡	0.414	0.482	0.379†	1.34	21.7
Zero	0.254	0.253	0.174	0.380	0.369	0.271	1.46	29.6
w/o ColBERT	0.362†‡	**0.360**†	0.322†‡	**0.426**	**0.492**	**0.394**†	8.88	46.0
only ColBERT	0.318†	0.328†	0.261†	0.395	0.373	0.298	1.23	21.1

Table 1: Performance on the two benchmarks. The † and ‡ show significant improvements over CONSENT and all GPT methods (paired t-test, $p<0.05$).

and Rare groups. Obviously, the LLMs struggle with tail entities that are not well covered in their training data. The highest contrast in nDCG@3 between EECATS and GPT methods is on the Rare group (around 12 percentage points higher), which consist of entities with very low popularities. This pattern could be seen again when other LMs such as CoSPLADE + T5 and especially dense retriver ConvDR do not perform well on tail entities either. Regarding CAsT with a focus on popular entities, Table 1 shows that our method EECATS achieves competitive nDCG@3 on this data as well.

Efficiency. The effectiveness of EECATS does not come at the cost of efficiency. EECATS is ca. 37 times faster in latency and ca. 11 times lower in TeraFLOPS than all baselines but ConvDR, which struggles to perform well with the limited training examples available in CATS. We discuss the efficiency on the CATS test set, with averages per turn. From the articles returned by BM25, we obtain 8,788 passages per turn. Many passages in an article overlap on the same sentences, each passage may have one to five consecutive sentences. After ranking these passages with ColBERT, we narrow the set down to the top 100 passages for re-ranking by the cross-encoder. In the default EECATS, ColBERT ranks 8,788 passages in 0.766 seconds, whereas it takes 8.39 seconds to rank all these passages with a cross-encoder. This speed difference is from the fact that ColBERT encodes each sentence in the top articles of the BM25 output once, and the maximal similarity vectors of passages are calculated efficiently with passages of consecutive sentences. Meanwhile, common texts in multiple overlapping passages could be processed multiple times if we use only cross-encoder. Overall, the efficiency of the EECATS pipeline comes from three factors: minimizing the number of times a cross-encoder is invoked, efficiently aggregating scores for ColBERT, and using in-conversation caching.

Ablation. We discuss the impact of two simplified pipelines: using only the final stage (like cross-encoder) to rank all passages from articles returned by BM25 (w/o ColBERT) and using only ColBERT to rank all these passages (only ColBERT). Table 1 gives results. Comparing EECATS and w/o ColBERT shows that nDCG@3 values are similar, but the default pipeline is 6.6 times faster. Comparing EECATS against only ColBERT, we observe that nDCG@3 for EECATS is much better, while both pipelines have similar latency. Besides, to verify the importance of fine-tuning SBERT, we compare EECATS against its Zero version, where the entity selection SBERT model is not fine tuned with our trainset. From the table, we can see that there is a big nDCG@3 gap between these methods, proving the importance of fine tuning SBERT for entity selection.

4 Related Work

Contextualization. In CIR [14,32,39], conversational questions are processed by *rewriting* into self-contained questions [3,9,10,17–20,24,34–37], or by *expansion* [4, 21, 23, 38]. A typical CIR method is CoSPLADE + T5 [15, 26], which follows the question expansion paradigm, utilizing relevant terms from the history. It adapts sparse retrieval methods SPLADE [11–13, 22] for conversations, and uses MonoT5 [26], a T5 [28] model fine-tuned on the MS MARCO, as a re-ranker. Popular method ConvDR [38] expands the current question with the prior history, then encode this into a representation for dense retrieval. Like other dense retrievers, ConvDR is very data-hungry and thus requires a lot of training examples for being effective.

CIR with Tail Entities. The state-of-the-art method for CIR with tail entities is CONSENT [33], which identifies relevant history entities using relatedness between the current question, prior turns, prior entities and answer candidates. The actual selection of entities is made by solving an integer linear program (ILP). This method selects relevant history entities and relevant answers in a joint manner. The CONSENT method is effective on a difficult benchmark with tail entities, but it is very slow.

Efficiency. The most effective CIR methods have high latency because they are based on computationally heavy techniques, like LLM (GPT), GNN (ExplaiGNN [4]) or ILP (CONSENT [33]). In contrast, a key point of our work is to design EECATS for low resource consumption and fast response times.

5 Conclusion

The presented method for conversational IR, EECATS, aims to optimize both effectiveness and efficiency. For efficient computation, our judiciously constructed pipeline pays the higher cost of a cross-encoder only for the final top-100 candidates. For effective answer ranking, in the presence of tail entities, our design for question-mention relatedness leverages a specifically fine-tuned light-weight LM. Experiments demonstrate that the design is successful on both dimensions, outperforming even LLMs like GPT-4o.

References

1. Brown, T.B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D.M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., Amodei, D.: Language models are few-shot learners. NIPS (2020)
2. Cao, Y., Xu, J., Liu, T.Y., Li, H., Huang, Y., Hon, H.W.: Adapting ranking svm to document retrieval. SIGIR (2006)
3. Chen, Z., Zhao, J., Fang, A., Fetahu, B., Rokhlenko, O., Malmasi, S.: Reinforced question rewriting for conversational question answering. EMNLP (2022)
4. Christmann, P., Roy, R.S., Weikum, G.: Explainable conversational question answering over heterogeneous sources via iterative graph neural networks. SIGIR (2023)
5. Craswell, N., Mitra, B., Campos, D., Yilmaz, E., Lin, J.: MS MARCO: Benchmarking ranking models in the large-data regime. SIGIR (2021)
6. Dalton, J., Xiong, C., Callan, J.: CAsT 2019: The conversational assistance track overview. TREC (2019)
7. Dalton, J., Xiong, C., Callan, J.: Cast 2020: The conversational assistance track overview. TREC (2020)
8. Dietz, L., Verma, M., Radlinski, F., Craswell, N.: Trec complex answer retrieval overview. TREC (2018)
9. Elgohary, A., Peskov, D., Boyd-Graber, J.: Can you unpack that? learning to rewrite questions-in-context. EMNLP-IJCNLP (2019)
10. Farzana, S., Zhou, Q., Ristoski, P.: Knowledge graph-enhanced neural query rewriting. WWW (2023)
11. Formal, T., Lassance, C., Piwowarski, B., Clinchant, S.: SPLADE v2: Sparse lexical and expansion model for information retrieval. arXiv:2109.10086 (2021)
12. Formal, T., Lassance, C., Piwowarski, B., Clinchant, S.: From distillation to hard negative sampling: Making sparse neural ir models more effective. SIGIR (2022)
13. Formal, T., Piwowarski, B., Clinchant, S.: SPLADE: Sparse lexical and expansion model for first stage ranking. SIGIR (2021)
14. Gao, J., Xiong, C., Bennett, P., Craswell, N.: Neural Approaches to Conversational Information Retrieval. Springer Cham (2022)
15. Hai, N.L., Gerald, T., Formal, T., Nie, J.Y., Piwowarski, B., Soulier, L.: Cosplade: Contextualizing splade for conversational information retrieval. ECIR (2023)
16. Hoffart, J., Milchevski, D., Weikum, G.: Stics: searching with strings, things, and cats. SIGIR (2014)
17. Jang, Y., Lee, K.i., Bae, H., Lee, H., Jung, K.: IterCQR: Iterative conversational query reformulation with retrieval guidance. NAACL (2024)
18. Ke, X., Zhang, J., Lv, X., Xu, Y., Cao, S., Li, C., Chen, H., Li, J.: Knowledge-augmented self-training of a question rewriter for conversational knowledge base question answering. EMNLP (2022)
19. Kim, G., Kim, H., Park, J., Kang, J.: Learn to resolve conversational dependency: A consistency training framework for conversational question answering. ACL-IJCNLP (2021)
20. Kostric, I., Balog, K.: A surprisingly simple yet effective multi-query rewriting method for conversational passage retrieval. SIGIR (2024)

21. Krasakis, A.M., Yates, A., Kanoulas, E.: Zero-shot query contextualization for conversational search. SIGIR (2022)
22. Lassance, C., Clinchant, S.: An efficiency study for splade models. SIGIR (2022)
23. Lin, S., Yang, J., Lin, J.: Contextualized query embeddings for conversational search. EMNLP (2021)
24. Mao, K., Dou, Z., Liu, B., Qian, H., Mo, F., Wu, X., Cheng, X., Cao, Z.: Search-oriented conversational query editing. ACL (2023)
25. Nogueira, R., Cho, K.: Passage re-ranking with BERT. arXiv:1901.04085 (2019)
26. Nogueira, R., Jiang, Z., Pradeep, R., Lin, J.: Document ranking with a pretrained sequence-to-sequence model. EMNLP (2020)
27. OpenAI: Gpt-4 technical report. arXiv:2303.08774 (2023)
28. Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., Liu, P.J.: Exploring the limits of transfer learning with a unified text-to-text transformer. JMLR (2020)
29. Reimers, N., Gurevych, I.: Sentence-bert: Sentence embeddings using siamese bert-networks. EMNLP (2019)
30. Robertson, S., Zaragoza, H.: The probabilistic relevance framework: Bm25 and beyond. Foundations and Trends in Information Retrieval (2009)
31. Santhanam, K., Khattab, O., Saad-Falcon, J., Potts, C., Zaharia, M.: Colbertv2: Effective and efficient retrieval via lightweight late interaction. NAACL (2022)
32. Thomas, P., Czerwinksi, M., Mcduff, D., Craswell, N.: Theories of conversation for conversational ir. TOIS (2021)
33. Tran, H.D., Yates, A., Weikum, G.: Conversational search with tail entities. ECIR (2024)
34. Vakulenko, S., Longpre, S., Tu, Z., Anantha, R.: Question rewriting for end to end conversational question answering. WSDM (2021)
35. Vakulenko, S., Voskarides, N., Tu, Z., Longpre, S.: A comparison of question rewriting methods for conversational passage retrieval. ECIR (2021)
36. Voskarides, N., Li, D., Ren, P., Kanoulas, E., de Rijke, M.: Query resolution for conversational search with limited supervision. SIGIR (2020)
37. Yu, S., Liu, J., Yang, J., Xiong, C., Bennett, P., Gao, J., Liu, Z.: Few-shot generative conversational query rewriting. SIGIR (2020)
38. Yu, S., Liu, Z., Xiong, C., Feng, T., Liu, Z.: Few-shot conversational dense retrieval. SIGIR (2021)
39. Zamani, H., Trippas, J.R., Dalton, J., Radlinski, F.: Conversational Information Seeking. Now Publishers (2023)