Efficient Answer Retrieval System (EARS): Combining Local DB Search and Web Search for Generative QA

Nikita Krayko[1] **Ivan Sidorov**[1,2] **Fedor Laputin**[1,2] **Daria Galimzianova**[1]
Vasily Konovalov[3,4]
[1]MTS AI [2]HSE University [3]AIRI, Moscow, Russia
[4]Moscow Institute of Physics and Technology, Russia
n.kraiko@mts.ai

Abstract

In this work, we propose an efficient answer retrieval system (**EARS**): a production-ready, factual question answering (QA) system that combines local knowledge base search with generative, context-based QA. To assess the quality of the generated content, we devise comprehensive metrics for both manual and automatic evaluation of the answers to questions. A distinctive feature of our system is the Ranker component, which ranks answer candidates based on their relevance. This feature enhances the effectiveness of local knowledge base retrieval by 23%. Another crucial aspect of our system is the LLM, which utilizes contextual information from a web search API to generate responses. This results in substantial 92.8% boost in the usefulness of voice-based responses. **EARS** is language-agnostic and can be applied to any data domain.

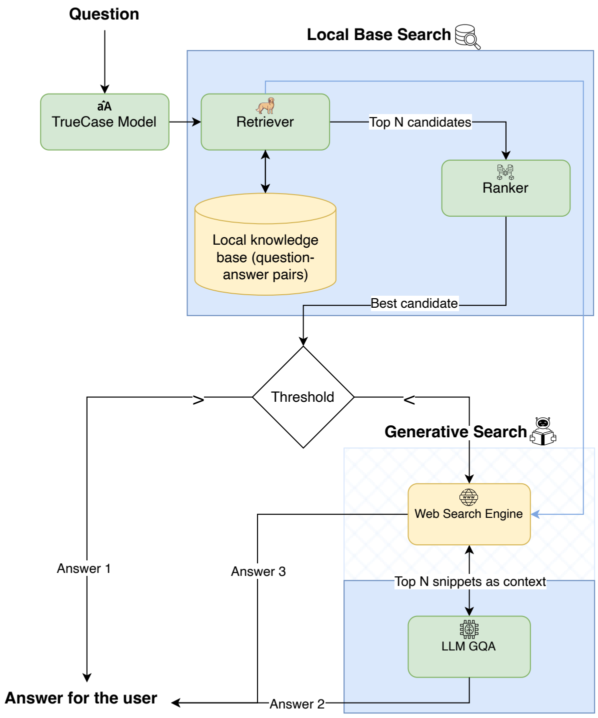


Figure 1: Combined search pipeline of EARS.

1 Introduction

Developing a virtual assistant is crucial for supporting clients as it provides 24/7 assistance, enhancing customer experience with instant responses and personalized interactions. It helps businesses scale their operations efficiently, reducing workload on human support teams and enabling them to focus on more complex issues. One of the essential components of a virtual assistant is a factual question-answering (QA) system. This system is capable of handling all user queries, providing answers to factual requests, whether domain-specific (related to the services of a particular company) or open-domain.

In this paper, we present a factual QA skill as one of the components of the virtual assistant, designed for the clients of Mobile TeleSystems (MTS) company[1]. The developed QA skill is proficient in addressing domain-specific inquiries about our services and products, along with open-domain questions. Our production-ready QA system seamlessly integrates two QA methodologies: knowledge base search and an LLM-based solution, enriched with context from a search engine. The system has been incorporated into both a chat-based interface and a voice interface.

Given that the skill accepts input from both chat and Automatic Speech Recognition (ASR), the query goes through extensive preprocessing (§3). After preprocessing, the user's query is processed by the Retriever (§4.1), which retrieves the top-n semantically relevant answer candidates from the knowledge base. Next, the Ranker (§4.2) selects the most appropriate answer from these candidates. However, if the Ranker lacks sufficient confidence

[1]More than 84 million subscribers

Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track, pages 1584–1594
November 12-16, 2024

in its choice, the question is passed to the Generative Search (§4.3). This component employs a Large Language Model (LLM) that is prompted to generate an answer based on relevant context from a web search API.

Our approach can be followed to develop your own factual QA system, capable of effectively assisting users with both domain-specific and open-domain inquiries. Our contribution consists of two key aspects:

1. We demonstrate how to effectively integrate knowledge base search with advanced LLM-based search techniques.

2. Our pipeline is language agnostic, enabling development of factual QA systems for any language.

2 Related work

Earlier QA systems were based on BM25 (Robertson and Zaragoza, 2009), a ranking function that estimates the relevance of documents to a given search query. Subsequently, BM25F (Pérez-Agüera et al., 2010) was developed, which enhances text relevance calculation by considering query terms across multiple specific fields. The BM25F class integrates the BM25 scores of the query term in these various fields.

The introduction of word embedding methods sparked the development of pipelines where proximity in the latent space serves as a metric for semantic retrieval. Cakaloglu et al. (2020) evaluated various text embeddings, including ELMo (Peters et al., 2018) and GloVe (Pennington et al., 2014), for question and paragraph embeddings. The Transformer encoder-based architecture (Vaswani et al., 2017) led to the introduction of multilingual E5 (Wang et al., 2024) text embeddings, which are trained through a multi-stage pipeline. When both the question and the context are provided (in reading comprehension), BERT-based models have brought about substantial improvements (Konovalov et al., 2020). In our pipeline, the local knowledge-based component relies on embeddings derived from the multilingual E5 model.

Knowledge Base Question Answering (KBQA) relies on knowledge graphs (KGs) to find the correct answer. KBQA involves applying two main approaches: semantic parsing (translating the question into an executable logical form) and retrieval-based methods (inferring answers from KG). The WQAqua (Diefenbach et al., 2017) pipeline starts with knowledge base grounding, after which the possible SPARQL queries are constructed that return non-empty answers when executed. Turganbay et al. (2023) outlines a generative model for QA that draws on textual content and knowledge graphs to uncover supportive information. Salnikov et al. (2023) proposed an algorithm for extracting subgraphs from a KG, based on question entities and answer candidates. Then the Transformer-based model is provided with linearized subgraph to generate response.

LLMs, when equipped with retrieval augmented generation (RAG), perform exceptionally well across a variety of tasks (Izacard et al., 2023; Shao et al., 2023). RAG improves the accuracy of LLMs' outputs by retrieving supplementary knowledge through specialized retrievers. This process enriches the prompts given to LLMs with relevant information from retrieved documents, enabling them to generate more precise and detailed content. Belikova et al. (2024) proposed a method to select a context for RAG-based system that is retrieved from different sources, including KGs. The selection method is based on Uncertainty Estimation (UE) techniques.

3 Preprocessing

The quality of search can be affected by the format of users' requests, which can stem from diverse sources such as chatbots, search bar widgets, or voice-based interfaces. These requests can be typed in lowercase, mixed case, or have capitalization errors. This is significant because the E5 (Wang et al., 2024) embedder model we employ is case-sensitive. The optimal performance is achieved when the user's input is orthographically correct. Therefore we develop the BERT-based (Devlin et al., 2019) **Truecase** component that fixes word casing. We train the model in token classification manner. For training, we utilize the tatoeba dataset (Tiedemann, 2020).

3.1 Local Base Search

A local database in our system is essential. It has been designed to meet customer requirements by providing pre-prepared answers in certain question domains, such as those containing advertising for other products of our company.

To address the need for up-to-date information in the knowledge base, we developed a pipeline for auto-updating data from Wikipedia and Wikidata (Vrandecic and Krötzsch, 2014). Wikipedia is the primary source of our knowledge base. It undergoes extensive preprocessing, cleaning, and question generation for each answer to be included in our KB. Questions are necessary because the retriever operates in a symmetric semantic search mode. We selected 300,000 most popular Wikipedia articles based on the last five years of page views to be incorporated into our local database.[2] The content is filtered and cleaned of special characters and editing artifacts, subsequently the questions are generated. In parallel, the Ranker features are gathered, including page views, categories and more.

With respect to Wikipedia, our methodology for generating questions for each article involves the following steps: Firstly, we employ Named Entity Recognition (NER) to identify and extract named entities from both the article titles and abstracts. Subsequently, we categorize these entities into various classes, including animate and inanimate objects, proper nouns, organizations, countries, and so on. This allows us to create questions that are tailored to the specific content of each article. With respect to Wikidata, every entity possesses structured properties, such as date of birth, citizenship, profession, and more. For each of these properties, we have devised specific question templates. As for other sources, the questions were crafted manually.

Beyond public sources, we include corporate sources guided by service needs. We also maintain an annotation management service that gathers data from alpha testing environment. Annotators utilize this platform to review and filter queries, manually annotate them, and generate accurate answers as needed. These annotations create distinct sources or tables within the databases, which are seamlessly integrated into the automatic update mechanism.

The final stage of the pipeline is merging all sources. We use the top 300,000 Wikipedia articles, extract data from Wikidata for 36,000 of them and include 4-5 additional smaller sources. Consequently, our knowledge base comprises around 400,000 entries and is updated several times a month.

[2]Based on the distribution of views and the system's performance, as well as to insure against the curse of dimensionality, we decided on the figure of 300,000 articles.

4 Combining Search Pipeline

General overview of our system can be found in Figure 1.

Once an input query (a question requesting a fact as an answer) is received by our system, it's processed in the following way:

1. The **Truecase** component recovers punctuation (including capitalization) that can occur in the query since the input can be received either from users typing it into the chatbot interface or from the ASR system. There are other ways to solve this problem, such as fine-tuning the **Retriever** or fine-tuning components in speech recognition block, but we decided to develop and implement a **Truecase** model.

2. The **Retriever** transforms the corrected query into vector representations with the E5 embedder. Then, the Retriever performs the Approximate Nearest Neighbor (ANN) search over the local database and if relevant answers are found, it returns top-n semantically close candidates. In case the local database lacks relevant information, the query is forwarded to the **Generative Search** module.

3. The retrieved candidates are sent into the **Ranker** model that improves the output from the Retriever and returns the best candidate based on its score.

4. Once a certain threshold (determined on validation) is reached, the answer retrieved from the local database is returned to the user (Answer 1).

5. If the local database lacks relevant answers, the **Generative Search** component is employed. The user query is forwarded to a web search API, which in turn provides the top-n (5 or 10) most relevant snippets. These snippets are short textual extracts from relevant web pages.

6. The snippets are concatenated into one string (context) which is passed to the LLM with the system prompt describing the task (Figure 4).

7. If the LLM returns *"No information"*, the context from the search API is returned to the user as the answer (Answer 3). Otherwise, the LLM generated answer is shown to the user (Answer 2).

4.1 Retriever

The Retriever component performs a search based on the local knowledge base. It consists of an embedder and an Approximate Nearest Neighbor (ANN) search. We incorporate both semantic search types: symmetric (query-query) and asymmetric (query-passage) that matches the concatenation of query, document and title to the input query.

We encode pre-compiled triplets (query-title-passage) from the local database using the multilingual E5 large embedding model (Wang et al., 2024). Next, the embeddings are indexed using Faiss (Johnson et al., 2021). We use the IndexFlatL2 method, which allows storing vectors in their original form without compression, as well as performing an accurate search. In our case, this is acceptable, since the database size is not huge (less than 1 million), and the search is performed on the GPU. According to the results of load testing of the our service, the index component executes for 8.17 ms on an A100 80 GB GPU. During inference, the user's input question is encoded by the same embedding model, and then passed to Faiss to fetch the top-10 semantically similar embeddings. If the top-1 score exceeds the threshold value, the system proceeds with the Ranker. Otherwise, it resorts to the Generative Search module.

4.2 Ranker

The Ranker's function is to identify the most accurate answer among the candidates retrieved by the Retriever. One of two implementations is usually used as a ranker model: (1) cross-encoder that gets two sentences as input and returns a value from 0 to 1 indicating the similarity between them; (2) a gradient boosting model trained with a tailored pairwise or listwise loss function like in Cao et al. (2007). Our system uses the second approach, as it's more flexible with the provision of additional features of a different nature.

A crucial aspect is that the system must return a single, most relevant document. To address this need, we use QuerySoftMax[3] loss function from CatBoost (Prokhorenkova et al., 2018) library.

Thus, using the gradient boosting model and tabular data representation allows to achieve improved performance by providing additional important information to the model (Appendix A provides a detailed description of each feature).

We evaluate our Ranker against other ranking techniques, including the performance metrics of the Retriever, the Ranker with basic features, the Ranker with all features, and a cutting-edge proprietary Cohere cross-encoder (rerank-multilingual-v3.0)[4]. The comparison is presented in the Table 2, which shows that even the base Ranker provides a quality boost of 16.5%. Our custom features add another 5.3% increase compared to the base Ranker, thus improving the overall retrieval quality by 22.7%.

4.3 Generative Search

When the answer chosen by the Ranker is insufficient or the question pertains to information not available in the local knowledge base, such as exchange rates, time-sensitive data, current news, real-time events, and similar, it is necessary to resort to external sources. The sign for insufficiency or absence of a response in the database is falling below a threshold. To respond to a question using external search engines, we need to navigate through two stages: (1) get the context relevant to the query; (2) provide the LLM with the request and context to answer the question. If the context lacks sufficient information for a response, the model should output *"No information"*.

To collect context, we use site snippets, namely, relevant pieces of documents from the first page of search engine API results. In the second stage, the compiled context and request are submitted to the LLM. The prompt can be found in Figure 4. It's crucial to indicate LLM to generate *"No information"* if there is no relevant information in the context, otherwise the LLM will hallucinate (Mallen et al., 2023). Thus, in the Generative Search component we employ RAG with relevant context retrieved from the internet via a search engine API.

We employ Mistral 7B (Jiang et al., 2023) in our pipeline as at the time of development this model provided the best quality and size trade-off for our case.

5 System Performance

We develop domain-specific validation sets that simulate probable user queries. Additionally, we have a comprehensive golden set comprising 1,600 factual questions, along with their corresponding

[3] https://catboost.ai/en/docs/concepts/loss-functions-ranking#QuerySoftMax

[4] https://cohere.com/blog/rerank-3

Model	Usefulness		*"No info"* proportion		Usefulness excl. *"No info"*	
	top-5	top-10	top-5	top-10	top-5	top-10
Mistral 7B	**59.58**	**65.43**	13.74	8.56	69.02	71.50
GPT-3.5	55.79	63.17	13.44	8.68	64.41	69.13
GPT-4o	50.09	50.00	**40.87**	**40.87**	**84.62**	**84.47**

Table 1: Usefulness on different LLMs evaluated on the golden set. Top-5 and top-10 indicate the number of search engine snippets passed to the model as context. The proportion of uninformative (*"No information"*) answers returned by the model is shown in *"No info"* proportion columns. The Usefulness metric calculated for only informative answers (excluding the uninformative) can be found in the last two columns.

Searcher	MAP@1
Retriever	0.6275
Retriever & base Ranker	0.7314
Retriever & all-features Ranker	**0.7705**
Retriever & Cohere Ranker	0.5874

Table 2: Retriever and Ranker metrics.

benchmark (ground-truth) answers crafted by humans. This set is highly representative of expected user questions (including the length and complexity).

The Usefulness metric is our key product metric for evaluating QA system responses. Although we define a "useful" answer simply as one that addresses the posed question, there are many nuances (some of the are highlighted in Appendix D which provides a detailed description of each usefulness value). The score can take values of 0, 0.5, or 1 for a single sample, and then these values are averaged over the validation set to get the final Usefulness.

Preliminary outcomes of the current system are depicted in Figure 2. The notable 92.8% improvement in voice-based answers, achieved through the optimal combination of +Ranker and +LLM, increased the overall Usefulness from 26.65% to 51.39%, taking into account the utility and contribution of each service component. The contribution of the Ranker is twofold: it reorders the Retriever's output and allocates the queries efficiently across the local and generative search. In our system, we assess the overall Usefulness of the entire service rather than each individual component. It's crucial to maximize the total final Usefulness by any means necessary, which renders the significance of individual components less critical.

In Figure 2b, one can observe the absence of the web search component's share, as we are unable to present web search results to the user through the audio channel; this is due to safety concerns and the lack of informativeness. Furthermore, it's important to note that, by definition, a QA system should respond to all posed questions. "*No Information*" cannot be considered a correct answer. Even in cases where a user asks an nonsensical question that cannot be answered properly, the LLM should provide a response indicating that the question does not have a definitively correct answer. Therefore, our objective is to minimize the share of "No Information" responses, as these responses offer no value or benefit to the service and are essentially useless.

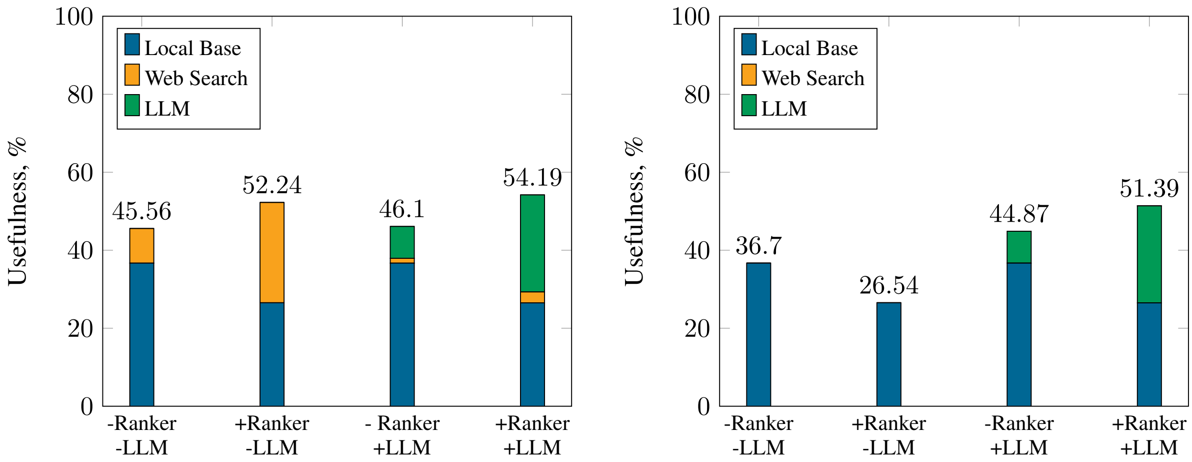
Upon optimizing the QA system validation, we focused on accelerating the service and migrating it to the Triton Inference Server[5]. By incorporating dynamic batching (Zha et al., 2019) for the local search module and continuous batching (Kwon et al., 2023) for the LLM, along with asynchronous external search engine queries for context, we achieved a 700% increase in RPS and a 500% reduction in response time. Currently, we are pursuing LLM quantization for further efficiency.

5.1 Human Evaluation

Human evaluation is capable of capturing a broad spectrum of elements and contextual aspects, such as cultural subtleties and the style of the text, which might be challenging for automated systems to fully grasp.

To speed up human annotation process, benchmark answers from golden sets were provided. As these benchmark answers need periodic updates to incorporate new information, a supplementary mechanism was implemented to automatically refresh items with updated data from knowledge base. The hourly rate paid to the annotators is approximately $2. The golden set has roughly 1,650 most

[5] https://developer.nvidia.com/triton-inference-server



(a) Distribution of Answers

(b) Distribution of Voiced Answers

Figure 2: The Usefulness measured across the Ranker and LLM components. In our virtual assistant, the bot answers are voiced by the text-to-speech model, and the answers obtained directly from the web search cannot be voiced. The histogram (a) includes Usefulness scores for all answers provided to the users. The histogram (b) excludes the web search component and only shows the scores for the answers our system was able to voice, which is the most desirable mode of operation. Both histograms show significant increase delivered by the Ranker and LLM components.

representative queries for system validation, this number allows us to optimize the validation process, both in terms of time spent and in terms of financial cost. By utilizing benchmark answers, the annotation time was decreased by approximately 30%. Furthermore, the inclusion of benchmark answers increased annotator consistency by an average of 6%.

We employ our primary golden set to assess two proprietary LLMs – ChatGPT 3.5 and GPT-4o – and evaluate their Usefulness side-by-side with open-source Mistral 7B (Jiang et al., 2023), which is integrated into the current system. The same set of hyperparameters is applied to all models (Table 4). Each model is evaluated with top-5 and top-10 context snippets received from the search engine API. The results can be found in the Table 1.

The average Usefulness for the open-source Mistral 7B answers in our setup is higher than any of the proprietary models. The refusal rate, which is the proportion of *"No information"* answers is the highest with GPT-4o (over 40% of all answers). If these uninformative answers are excluded from the overall Usefulness calculation, GPT-4o achieves significantly higher quality than any other model. This indicates that open-source LLMs might benefit from fine-tuning for context following, a strategy we plan to implement in future version of our system.

5.2 Automatic Evaluation

In addition, we implemented an automatic evaluation method, where the primary criterion was the answer's Correctness. In this type of assessment, the system-generated response is compared against the established ground truth (Es et al., 2024).

Two primary methods are employed for automated validation: (1) deterministic metrics, which don't incorporate stochastic components (such as Precision, Recall, F1-score, BLEU, ROUGE) in conjunction with Cosine Similarity, and (2) LLM-as-a-judge (Zheng et al., 2023). Both approaches are designed to correspond Usefulness metric.

The first approach often results in a weak correlation with human evaluation (Figure 3), while the LLM-as-a-judge, due to its nature, poses challenges to interpretation and can be resource-intensive, especially when utilizing proprietary models like GPT-4. Consequently, an intermediate framework was developed. It combines deterministic metrics into ensemble using Gradient Boosting Classifier (Prokhorenkova et al., 2018), providing a slightly lower performance compared to the LLM-as-a-judge yet being practically free in terms of resources and improving the use of certain metrics.

The ensemble acts as an extra quality control step prior human evaluation. This approach significantly improves the efficiency of the annotation process. It should be used as an additional stage, and not the main one, replacing the stage of human

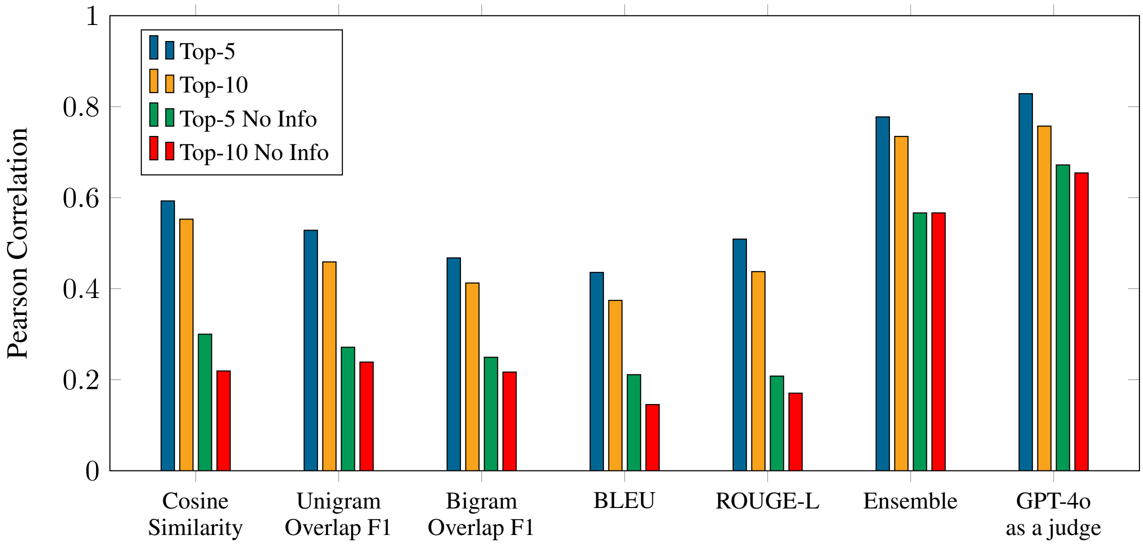


Figure 3: Metrics of both approaches evaluated on the golden set and compared using Pearson's correlation with human evaluation (Usefulness) labels. Top-5 and top-10 indicate the number of search engine snippets passed to the model as context. Variations with different refuse rates (No info) from Mistral 7B are included.

evaluation. It reduces the amount of data that requires human evaluation, concentrating the efforts and time of annotators only on the most complex and ambiguous cases.

6 Conclusion

In this work, we introduced **EARS**, a production-ready factual question answering system. Answers can be derived either from a local knowledge base or generated by an LLM using context obtained from the web search API. We outlined the comprehensive workflow for addressing factual questions, making it reproducible for any language. The cornerstone of the pipeline is the Ranker, which enhances retrieval quality by 23%. Incorporating the LLM boosts the quality of our chatbot's spoken answers by 92.8%. Furthermore, we developed a suite of automatic metrics to reduce reliance on human annotators.

Limitations

In spite of bringing a significant increase in usefulness in the task of factual question answering, our system has a few limitations that are planned to be covered in the future work.

One serious limitation is that the LLM that we deployed has not been fine-tuned to answer questions based on the context. Another point that could be covered in future research is extending the context. In our current deployment, the model is only tested on 5 and 10 top hits from the web search API, which can be increased to several pages of content, since the context window of latest LLMs allows that. Furthermore, we aim to integrate the results obtained from various web search APIs in order to improve both the accuracy and comprehensiveness of our results.

The human evaluation that we partly rely on to assess the performance of our system is a rather costly and lengthy process. It requires detailed instructions and proper training for annotators, which is not always feasible. One way to alleviate this is to use automatic metrics.

Our system might not be applicable for every domain of data. The questions that require long answers (e.g. food recipes) could be too difficult to cover with a local base, since each entry would require manual parsing. We are currently researching the types of LLM agents and building systems with multiple agents in order to handle cases that require constant data updates. For example, exchange rates, work schedules of organizations, routes and interesting events nearby that require third-party APIs and cannot be covered by the Web Search context.

Moreover, to enhance performance, it is advisable to investigate LLM quantization techniques.

References

Julia Belikova, Evegeniy Beliakin, and Vasily Konovalov. 2024. JellyBell at TextGraphs-17 shared task: Fusing large language models with external knowledge for enhanced question answering. In *Proceedings of TextGraphs-17: Graph-based Methods for Natural Language Processing*, pages 154–160, Bangkok, Thailand. Association for Computational Linguistics.

Ravish Bhagdev, Sam Chapman, Fabio Ciravegna, Vitaveska Lanfranchi, and Daniela Petrelli. 2008. Hybrid search: Effectively combining keywords and semantic searches. In *The Semantic Web: Research and Applications, 5th European Semantic Web Conference, ESWC 2008, Tenerife, Canary Islands, Spain, June 1-5, 2008, Proceedings*, volume 5021 of *Lecture Notes in Computer Science*, pages 554–568. Springer.

Tolgahan Cakaloglu, Christian Szegedy, and Xiaowei Xu. 2020. Text embeddings for retrieval from a large knowledge base. In *Research Challenges in Information Science*, pages 338–351, Cham. Springer International Publishing.

Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. 2007. Learning to rank: from pairwise approach to listwise approach. In *Machine Learning, Proceedings of the Twenty-Fourth International Conference (ICML 2007), Corvallis, Oregon, USA, June 20-24, 2007*, volume 227 of *ACM International Conference Proceeding Series*, pages 129–136. ACM.

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, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics.

Dennis Diefenbach, V. López, Kamal Deep Singh, and Pierre Maret. 2017. Core techniques of question answering systems over knowledge bases: a survey. *Knowledge and Information Systems*, 55:529 – 569.

Shahul Es, Jithin James, Luis Espinosa Anke, and Steven Schockaert. 2024. RAGAs: Automated evaluation of retrieval augmented generation. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*, pages 150–158, St. Julians, Malta. Association for Computational Linguistics.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2021. Deberta: decoding-enhanced bert with disentangled attention. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Gautier Izacard, Patrick S. H. Lewis, Maria Lomeli, Lucas Hosseini, Fabio Petroni, Timo Schick, Jane Dwivedi-Yu, Armand Joulin, Sebastian Riedel, and Edouard Grave. 2023. Atlas: Few-shot learning with retrieval augmented language models. *J. Mach. Learn. Res.*, 24:251:1–251:43.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de Las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *CoRR*, abs/2310.06825.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2021. Billion-scale similarity search with gpus. *IEEE Trans. Big Data*, 7(3):535–547.

Vasily Konovalov, Pavel Gulyaev, Alexey Sorokin, Yury Kuratov, and Mikhail Burtsev. 2020. Exploring the bert cross-lingual transfer for reading comprehension. In *Computational Linguistics and Intellectual Technologies*, pages 445–453.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles, SOSP 2023, Koblenz, Germany, October 23-26, 2023*, pages 611–626. ACM.

Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 9802–9822. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

José R. Pérez-Agüera, Javier Arroyo, Jane Greenberg, Joaquín Pérez-Iglesias, and Víctor Fresno. 2010. Using BM25F for semantic search. In *Proceedings of the 3rd International Semantic Search Workshop, SEMSEARCH '10, Raleigh, North Carolina, USA, April 26, 2010*, pages 2:1–2:8. ACM.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Liudmila Ostroumova Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. 2018. Catboost: unbiased boosting with categorical features. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 6639–6649.

Stephen E. Robertson and Hugo Zaragoza. 2009. The probabilistic relevance framework: BM25 and beyond. *Found. Trends Inf. Retr.*, 3(4):333–389.

Mikhail Salnikov, Hai Le, Prateek Rajput, Irina Nikishina, Pavel Braslavski, Valentin Malykh, and Alexander Panchenko. 2023. Large language models meet knowledge graphs to answer factoid questions. In *Proceedings of the 37th Pacific Asia Conference on Language, Information and Computation*, pages 635–644, Hong Kong, China. Association for Computational Linguistics.

Zhihong Shao, Yeyun Gong, Yelong Shen, Minlie Huang, Nan Duan, and Weizhu Chen. 2023. Enhancing retrieval-augmented large language models with iterative retrieval-generation synergy. In *Findings of the Association for Computational Linguistics: EMNLP 2023, Singapore, December 6-10, 2023*, pages 9248–9274. Association for Computational Linguistics.

Jörg Tiedemann. 2020. The tatoeba translation challenge - realistic data sets for low resource and multilingual MT. In *Proceedings of the Fifth Conference on Machine Translation, WMT@EMNLP 2020, Online, November 19-20, 2020*, pages 1174–1182. Association for Computational Linguistics.

Raushan Turganbay, Viacheslav Surkov, Dmitry Evseev, and Mikhail Drobyshevskiy. 2023. Generative question answering systems over knowledge graphs and text. *COMPUTATIONAL LINGUISTICS AND INTELLECTUAL TECHNOLOGIES"*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

Denny Vrandecic and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *Commun. ACM*, 57(10):78–85.

Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. 2024. Multilingual E5 text embeddings: A technical report. *CoRR*, abs/2402.05672.

Sheng Zha, Ziheng Jiang, Haibin Lin, and Zhi Zhang. 2019. Just-in-time dynamic-batching. *CoRR*, abs/1904.07421.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

A Ranker features

Below are some features employed by the Ranker boosting model:

Retriever similarity features significantly contribute to the ranker performance, virtually ensuring it doesn't degrade the retriever's initial ranking. This feature represents cosine similarity between the input question and the retrieved question as described in the Section 4.1.

Embedding features has been supported by CatBoost by using Linear Discriminant Analysis (LDA), vectors are converted into a single numerical column. At the first stage, all text information related to the retrieved document (query, title, passage), as well as input query, are encoded into embeddings using the multilingual E5 large embedding model (Wang et al., 2024). At the second stage, all emdeddings are reduced from 1024 to 512 components using Principal Component Analysis (PCA). And at the third stage, reduced embeddings are passed to the Ranker model as a features to take into account the text attributes of the input query, as well as the retrieved query, title and passage.

Natural Language Inference (NLI) based on DeBERTa (He et al., 2021) determines the relationship between the question and answer candidates. To be precise, we use concatenation of the title and the answer as a premise and the input query as a hypothesis. The resulting entailment, neutral, and contradiction logits are passed as a separate numerical features to the Ranker.

Named Entity Recognition (NER) extracts entities from the user's query and the discovered document. Entity classes are transformed into categorical features, while the entities themselves are encoded and passed to the ranker as an embedding feature.

BM25 (Robertson and Zaragoza, 2009) is used in addition to calculated at the retrieval stage similarity features, which are created by conducting a parallel document search. This technique is known as Hybrid Search (Bhagdev et al., 2008). We calculate BM25 score for the top-10 documents received by Retriever using the input query and concatenation of the retrieved query and the answer. It helps to take into account not the semantic, but the lexical relationship between the request and the document.

Document popularity based on access and viewing statistics of the local knowledge base (primarily Wikipedia and Wikidata), is incorporated into the feature vector. These features are particularly useful in cases of namesakes, homonyms, identical book or film titles, etc. We extract the views of all articles in our knowledge base over the past two years and then calculate various statistics such as sum, median, mean and standard deviation to pass them as a features to the Ranker.

Type	Features
Embedding	query, title, passage
Score	top-N, Retriever score, BM25 score
Text	NLI logits, NER entities similarity
Numeric	views: sum, median, mean, std

Table 3: Overview of the features of Gradient Boosting ranker.

B Generative Search prompt

{question}
{context}
You have been given a question and a context compiled from different sources, answer the question following strictly the context further. If you cannot answer the question, then write 'No information'. Answering a question out of context is prohibited.

Figure 4: LLM prompt for context question answering.

C LLM Inference Hyperparameters

Hyperparameter	Value
Temperature	0.01
Top-p	0.8
Max new tokens	128

Table 4: Inference hyperparameters for LLM.

D The Usefulness metric and annotators guidelines examples

Regarding voice channel, there are two potential sources for answers: a response derived from a local base search and a generative response produced by LLM. The system itself decides which component will provide the answer to a given query. A response is considered useful only if it directly addresses the question at hand. The usefulness of the system's responses can be either 0, 0.5, or 1, depending on the accuracy of the answer. A usefulness score of 0.5 indicates a partially useful answer, which typically occurs when the generative response is incomplete or fails to fully address the question. The following are several examples for each Usefulness score, accompanied by explanations and illustrations taken from the annotators' guidelines:

Usefulness is 0 (not useful response):

1. **Question:** *How many people live in Budapest?*

   Response: Local Base: *Budapest is the capital and largest city of Hungary. It is located in the northern part of the country on both banks of the Danube. It is the ninth most populous city in the European Union, the center of an agglomeration with a population of about 2.97 million people.*

   Usefulness: 0

   Explanation: It's necessary to check the fact about the population of Budapest. The correct answer from Wikipedia is: "The population of Budapest on January 1, 2023 is 1,671,004 people". The system's response differs from the actual information, so the Usefulness is 0.

2. **Question:** *Who invented the thermopot?*

   Response: LLM: *The thermopot was invented in Japan in 1992.*

   Usefulness: 0

   Explanation: The question did not pertain to the location or year of invention, but rather sought information about the specific individual who invented the thermopot. But this detail was absent from the system's response. A more accurate response would have been as follows: "The thermopot was invented in Japan in 1992. It is impossible to say exactly who invented it. Arthur Large came up with the idea of building a heating element into the kettle itself. And Russell Hobbs came up with the idea of making kettles automatic. All this influenced the creation of the thermopot."

3. **Question:** *Does a spider have teeth?*

   Response: LLM: *Yes, spiders have teeth.*

   Usefulness: 0

   Explanation: The system's response originated from LLM. Upon investigation on the Internet, we realize that the model's response was factually incorrect. The accurate answer is as follows: "Spiders don't have teeth, but they do have chelicerae — they contain the ducts of poisonous glands that secrete digestive juice."

Usefulness is 0.5 (partially useful response):

1. **Question:** *What is the name of the lead singer of Linkin Park?*

   Response: LLM: *Chester.*

   Usefulness: 0.5

   Explanation: Providing only a single name is insufficient; at the very least, the singer's last name should also be included. A more accurate response to the query would be: "The lead singer of the band "Linkin Park" is Chester Charles Bennington."

2. **Question:** *Name the fruits that are grown in Cyprus?*

   Response: LLM: *Cyprus grows oranges, peaches, cherries, kiwis and dates.*

   Usefulness: 0.5

   Explanation: It's evident that the list of fruits cultivated in this country is not exhaustive, as other produce such as figs and grapes are also grown there. If we had included phrases like "and so on" or "among others," the response would have received a full Usefulness of 1. However, since this was not done, the response is only rated at 0.5.

Usefulness is 1 (useful response):

1. **Question:** *Who is considered the founder of physics?*

   Response: LLM: *Newton is considered the founder of physics.*

   Usefulness: 1

   Explanation: The question is phrased in the singular form, and according to multiple sources, Sir Isaac Newton can indeed be regarded as the founding figure of classical physics. Although Galileo Galilei and René Descartes also made significant contributions to the field, based on the singular nature of the question, we conclude that Newton is the primary answer. Therefore, we consider this response to be a complete and accurate answer with a Usefulness of 1.

2. **Question:** *Who was the President of Italy in 2000?*

   Response: Local Base: *Carlo Azeglio Ciampi was an Italian statesman, the tenth President of the Italian Republic, and Chairman of the Council of Ministers of Italy.*

   Usefulness: 1

   Explanation: The answer was originated from a local base search. The sources confirm the correctness of the answer, so we rate the Usefulness at 1.