# Conversational Geographic Question Answering for Route Optimization: An LLM and Continuous Retrieval-Augmented Generation Approach

Jose Tupayachi
jtupayac@vols.utk.edu
University of Tennessee, Knoxville
Knoxville, Tennessee, USA

Xueping Li
Xueping.Li@utk.edu
University of Tennessee, Knoxville
Knoxville, Tennessee, USA

## ABSTRACT

We present a pilot study exploring the potential of Large Language Models (LLMs) to interface with application programming interfaces through logical instructions, specifically within the domain of Geographic Question Answering for route optimization. This study employs a Continuous Retrieval-Augmented Generation approach combined with fine-tuned LLMs, featuring customized node-based storage and vector search retrieval. We also provide a comparative analysis of the method's effectiveness and adaptability in handling diverse textual queries.

## KEYWORDS

Question Answering, Retrieval Augmented Generation, Geographical Information, Large Language Models

**ACM Reference Format:**
Jose Tupayachi and Xueping Li. 2024. Conversational Geographic Question Answering for Route Optimization: An LLM and Continuous Retrieval-Augmented Generation Approach . In *17th ACM SIGSPATIAL International Workshop on Computational Transportation Science GenAI and Smart Mobility Session (IWCTS'24), October 29-November 1 2024, Atlanta, GA, USA.* ACM, Seattle, WA, USA, 4 pages. https://doi.org/10.1145/3681772.3698217

## 1 INTRODUCTION

Recent advancements in Natural Language Processing (NLP) and Large Language Models, known as transformers, have been adapted across various fields [2]. Knowledge Graphs (KGs) aid in converting complex data into organized network entities, thereby enabling users to query and extract information effectively. The integration of KGs can improve search engines by analyzing both semantic and structured data. Relationships and semantic connections excel at representing complex domains linking information through ontologies [9]. However, as the scale and diversity of information grows, there is an increasing need for more flexible retrieval methods. Vectorized databases represent a significant evolution in information storage and retrieval when paired with solutions like vector search [12]. Encoding data into high-dimensional vectors, enables advanced search capabilities beyond keyword matching, allowing for similarity-based retrieval using embedding [17, 20]. This shift handles unstructured data effectively, suitable for topic-specialized chatbots that can access vast amounts of information retrieving facts based on similarity. Question Answering (QA) is an emerging field with the support of Large Language models (LLMs), A type of transformers [2] are trained on extensive and diverse corpora of text and perform language generation, whose goal is to generate automated answers for questions in natural language [24]. Yet, spatial information poses higher challenges such as the calculation of spatial proximity between two locations [15]. Geographic QA (GQA) focuses on the development of systems that can respond to questions containing geographical information with high accuracy [14]. Utilizing REST API [1] allows to interact with back-end applications via URLs and methods for data operations. [5, 6] This serves as a knowledge distribution that grants access to vast pools of specialized operations. However, effectively leveraging this information often requires adherence to rigid rules. The interactive use of LLMs introduces flexibility, thanks to their capabilities. Despite their potential, most solutions still rely on pre-trained extensively labeled fixed data, which can restrict applicability. Retrieval-augmented generation addresses the limitations of pre-trained models, which often demand substantial hardware resources. By leveraging external domain knowledge sources, RAG reduces these dependencies, enabling more flexible knowledge. Our approach enables non-technical users to explore and obtain insights into the delivery network by using a semi-automated approach by employing specialized information sourced from API endpoints. The Continuous RAG (CRAG) lies on uses a question and instruction logic file, that can be updated impacting the knowledge base. The conversational process is explained under Algorithm 1. Source files are available under https://github.com/jtupayachi/CRAG_PAPER

## 2 METHOD

Specialized knowledge demands extensive training in specific domains, along with advanced reasoning capabilities [8]. Techniques like RAG can provide domain-specific responses. The work by [21], has demonstrated innovative approaches to question-answering GIS data. We leverage external databases through a REST API and incorporate a set of instructions for output generation. [7] employs a multi-format data pipeline. Unstructured text data is leveraged to develop reasoning based on a set of API documentation endpoints through Swagger [19] pair with human input questions and

This research is supported by ARPA-E under award number DE-AR0001780.


*IWCTS'24, October 29-November 1 2024, Atlanta, GA, USA*

ACM ISBN 979-8-4007-1151-0/24/10…$15.00
https://doi.org/10.1145/3681772.3698217

[1] Representational State Transfer Application Programming Interface

instructions as presented under 1. The work of [23] focuses on the fine-tuning of domain-specific initial knowledge sources, which limit the flexibility of updates, hindering its adaptability to evolving information within the domain.

The logical component in Figure 1, which encompasses API documentation, questions, and instructions, enables the generation of specialized responses for each query based on generic instructions. In Figure 1, the initial source of information consists of a set of proceeded geographical data optimized for shortest path and route calculations using GeoPandas for GIS parsing and NetworkX for shortest path determination and neighboring location. For this research, two endpoints LCA (Life Cycle Analysis) and CRD (Points of Interest ) are deployed. Endpoint information and generic instructions to retrieve data are included in this document in JSON format. This information is encoded and later stored. CRAG allows for continuous updates of the mapping or pairing questions.

Upon user request the approach utilizes the conversational pipeline in Algorithm 1 to determine, first, the correct endpoint. Vector search maps the prompt to the questions and then retrieves the instructions. The question prompts are ranked based on cosine similarity [13] by applying embeddings on the user prompt. Questions along with instructions are retrieved to be followed by the LLM. This ensures dynamic, domain-oriented responses but also maintains a well-defined structure. To achieve a tailored output, fine-tuning is necessary. However, we maintain generic instructions, that can be reused among the different endpoints, for shell scripting within our logical component to ensure flexibility and adaptability in various contexts.

The bottom section of Figure 2 shows the point of update for the CRAG approach upon executable and verified shell code users decide to insert different types of prompts and their retrieval instructions. This feature enables to enhancement of question-instruction mapping, particularly when training is selected enhancing flexibility in user interactions. This is less resource-intensive compared to the initial fine-tuning process, making it accessible for a wider range of applications.

## 2.1 Storage & Retrieval:

In order to encode using embeddings we use the `all-MiniLM-L6-v2` model. This uses 6 layers and small hidden sizes. Carries a self-attention mechanism in computing a weighted sum of representations of all words in a sentence. Input tokens are converted into dense vectors through multiple layers of self-attention and feed-forward networks. Each layer applies a linear transformation and non-linear activation functions. Contextualized embeddings for each token generate a single embedding for the entire sentence. Here, $\mathbf{e}_i$ represents the embedding of the $i$-th token, and $N$ is the total number of tokens in the sentence. The semantic similarity between text elements is determined by measuring distances or similarities between their vector representations. Information is indexed based on their embeddings. When user input is made, its embedding is compared to the embeddings of documents in the index to find the most similar question on the logical component. We employ pgvector [11] and it is set to the nearest neighbor search algorithm providing perfect recall. Let $\mathbf{E} \in \mathrm{R}^d$ represent the given query embedding. Let $\mathbf{e_i} \in \mathrm{R}^d$ represent the stored embedding for the i-th record in the table. The cosine distance D($\mathbf{E}$, $\mathbf{e_i}$) between $\mathbf{E}$ and $\mathbf{e_i}$ is defined as: $D(\mathbf{E}, \mathbf{e_i}) = 1 - \frac{\mathbf{E} \cdot \mathbf{e_i}}{\|\mathbf{E}\| \|\mathbf{e_i}\|}$ where $\mathbf{E} \cdot \mathbf{e_i}$ is the dot product between the query embedding E and the stored embedding $\mathbf{e_i}$, and $\|\mathbf{E}\|$ and $\|\mathbf{e_i}\|$ are the magnitudes of the embedding. The query retrieves the records $r_i$ from the database table, calculating the cosine distance D(E, $\mathbf{e_i}$) for each record. The records are then sorted by their cosine distance D(E, $\mathbf{e_i}$) in ascending order, with the top `limit` records being selected. If `metadata_filters` are provided, an additional filter condition is applied to the query to restrict the results based on the metadata. The query retrieves up to `limit` records $r_i$ such that the cosine distance D(E, $\mathbf{e_i}$) is minimized.

## 2.2 Fine-Tuning

The upper section of Figure 2 illustrates the fine-tuning process for a base Gemma 2 model, which begins with the instructions dataset. We utilize 900 training observations and 100 test observations. The training phase employs the tokenizer from the base model to process the input data. To optimize the training under constraints of GPU memory and model size, we incorporate Flash Attention [3] and QLoRA [4]. Flash Attention enhances memory efficiency by allowing for faster computation, enabling the model to focus on relevant parts of the input without excessive memory usage. Meanwhile, QLoRA facilitates efficient fine-tuning by leveraging quantization methods that reduce the model's memory footprint. Upon completing the training of the adapter, it is seamlessly integrated into the main model, resulting in a GUFF file designated as Gemma 2 CRAG [10]. This process is computationally intensive.

# 3 COMPUTATION & EXPERIMENTATION

We use the fine-tuned model based on Gemma 2 9B IT and endpoint information packed in the logical component to generate shell code based on a set of instructions that iteratively the LLM follows. Testing is performed by the use of Selenium web driver [18] following an automated procedure that queries the UI retrieves the response and compares it with the reference response. This follows the conversational pipeline in Algorithm 1. The setup includes four instances comprising CRD and LCA data. The CRD instances contain coordinates and information about delivery modes accepted at each location. The LCA instances capture details such as fuel type, vehicle type, and transportation mode, and calculate the $CO_2$ equivalent emissions. Testing is conducted by evaluating 1-2 parameters for the CRD endpoint and 2-3 parameters for the LCA endpoint. The framework performance evaluation uses quantitative metrics such as BERTscore [2] [22], METEOR [3] [1], and BLEU [4] [16] to rank the error analysis. We define a base scenario as one that does not include instruction-based fine-tuning, relying solely on an RAG approach for mapping and sharing information. In contrast, a CRAG scenario incorporates instruction-based fine-tuning with general-purpose instructions to enhance the mapping and sharing process.

Figure 3 presents a detailed comparison of performance across scenarios and approaches. Increasing the number of parameters appears to complicate both the instructional query and retrieval

[2] Bidirectional Encoder Representations from Transformers
[3] Metric for Evaluation of Translation with Explicit ORdering
[4] Bilingual Evaluation Understudy

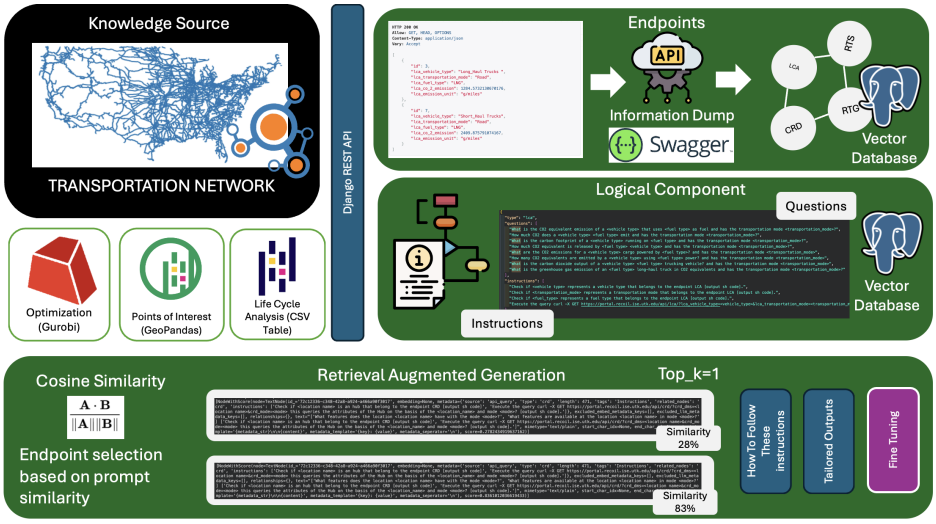



**Figure 1: Architecture Overview.**

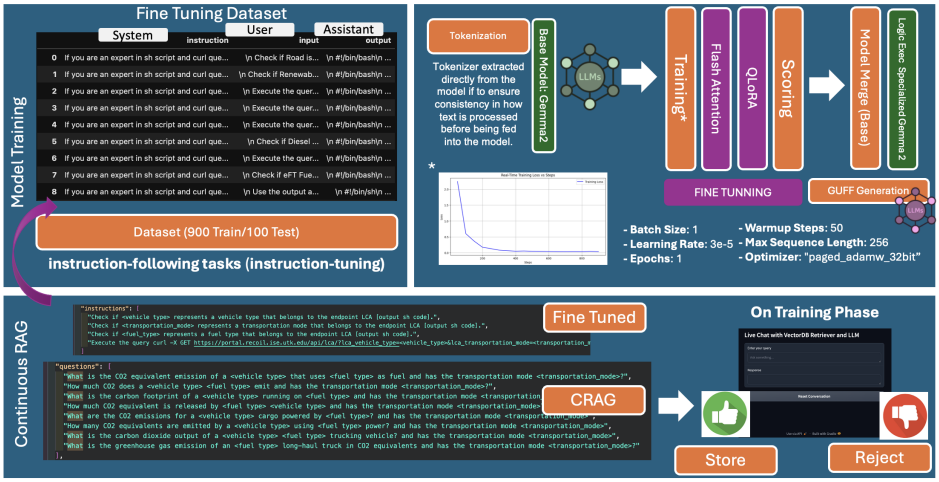
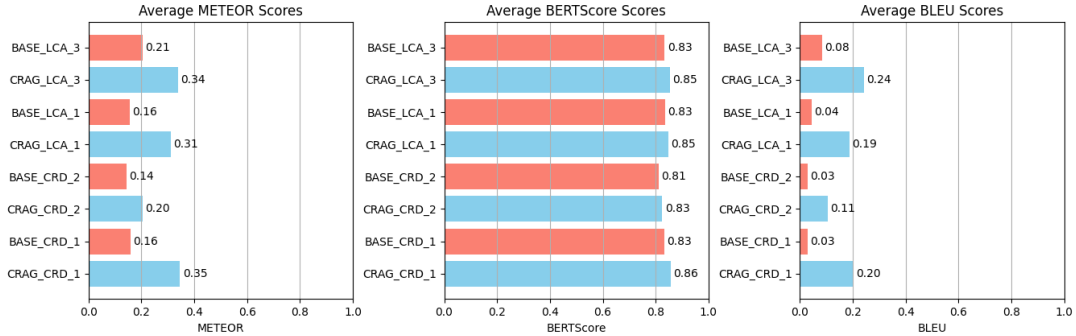



**Figure 2: Fine tuning process model and continuous RAG.**

**Figure 3: Performance Metrics for CRD and LCA.**




processes, making these interactions more complex to interpret. Notably, BERTscore remains stable for cases where CRAG instances show favorable differences compared to BASE instances. Significant variations, however, are observed in METEOR scores, while BLEU penalizes instances heavily by counting the overlapping n-grams between predicted and gold-standard translations. Overall, the base scenario shows a notable decline in comparison metrics during LCA testing. METEOR scores indicate stronger performance for both

CRD and LCA models independently across most configurations. The experimental setup maintained a temperature of 0.3, generated up to 128 new tokens, used a context window of 1024, and leveraged embeddings with a dimensionality of 384.

## 4 CONCLUSIONS & FUTURE DIRECTIONS

The proposed approach, when combined with the CRAG, aims at ensuring accurate responses. Our workflow leverages the integration of LLM and RAG creating a powerful platform capable of executing API requests across various instances. The use of the logical component based on questions and instructions allows the use of domain knowledge to handle the interaction of the API endpoints. Future directions will focus on extensive comparison considering a higher complexity request to evaluate the impact on the methodology.

**Algorithm 1** Conversational Pipeline: CO2 Emission Calculation

**Input:** User query $Q$ (e.g., `What is the CO2 equivalent emission of a` $< vehicle\_type >$ `that uses` $< fuel\_type >$ `as fuel from` $< Origin >$ `to` $< Destination >$ `and has the transportation mode road?`)
**Step 1: User Request**; $Q = f(\text{Input})$, where $Q \in \mathcal{D}$ and $\mathcal{D}$ represents the domain of possible user queries.
**Step 2: Execute Vector Search**;
$R_{\text{raw}} = \texttt{VectorSearch}(Q)$ ▷ Retrieve relevant information based on the user query using vector search
**Step 3: Query Validation and Instructions**;
**if** RAG results indicate no similar prompt **then**
    `Invalid prompt provided`
**else**
    `Extract logic instructions from RAG results`
**end if**
**Step 4: Backend Processing**; Provide queried data $R_{\text{raw}}$ to the Fine-tuned LLM model to obtain shell code: $R_{\text{refined}} = h(R_{\text{raw}}, \mathcal{M})$, where $\mathcal{M}$ is the RAG model.
**Step 5: Follow Instructions**;
$E = R_{\text{refined}}$ ▷ e.g CO2 emissions calculated based on the refined results procedure tailored by LLM and instructed by CRAG logic.
**Step 6: Machine Action (Response Generation)**; Generate the response $R$ based on $E$: $R = \text{GenerateResponse}(E)$.
**Step 7: Response Validation**
**while** $R$ does not meet expected criteria (e.g., completeness, correctness) **do**
    Return to Step 2 and repeat the process.
**end while**
**if** $R$ meets expected criteria **and** CRAG training mode is active **then**
    Store $R$ within logic.
**else**
    Reject $R$ from storage.
**end if**
**Output:** Final response $R$ (e.g. `The CO2 equivalent emission for the vehicle is X kg for the journey from` $< Origin >$ `to` $< Destination >$.)

## REFERENCES


[1] Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*. 65–72.

[2] Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, et al. 2023. A survey on evaluation of large language models. *ACM Transactions on Intelligent Systems and Technology* (2023).

[3] Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness. arXiv:2205.14135 [cs.LG] https://arxiv.org/abs/2205.14135

[4] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. QLoRA: Efficient Finetuning of Quantized LLMs. arXiv:2305.14314 [cs.LG] https://arxiv.org/abs/2305.14314

[5] Django REST framework. n.d.. Django REST framework. https://www.django-rest-framework.org Accessed: 2024-10-28.

[6] Roy Thomas Fielding. 2000. *Architectural Styles and the Design of Network-based Software Architectures*. Ph.D. Dissertation. University of California, Irvine, Irvine, CA, USA. Dissertation Committee: Professor Richard N. Taylor, Chair, Professor Mark S. Ackerman, Professor David S. Rosenblum.

[7] Gihan Gamage, Nishan Mills, Daswin De Silva, Milos Manic, Harsha Moraliyage, Andrew Jennings, and Damminda Alahakoon. 2024. Multi-Agent RAG Chatbot Architecture for Decision Support in Net-Zero Emission Energy Systems. In *2024 IEEE International Conference on Industrial Technology (ICIT)*. IEEE, 1–6.

[8] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, and Haofen Wang. 2023. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997* (2023).

[9] Shaoxiong Ji, Shirui Pan, Erik Cambria, Pekka Marttinen, and S Yu Philip. 2021. A survey on knowledge graphs: Representation, acquisition, and applications. *IEEE transactions on neural networks and learning systems* 33, 2 (2021), 494–514.

[10] Jose Tupayachi. 2024. gemma-2-9b-it-crag (Revision 85f6eb4). https://doi.org/10.57967/hf/3360

[11] Linnakangas Heikki Kane Andrew. 2024. pgvector: Open-source extension for vector similarity search in PostgreSQL. https://github.com/pgvector/pgvector. https://github.com/pgvector/pgvector GitHub repository.

[12] Timo Kersten, Viktor Leis, Alfons Kemper, Thomas Neumann, Andrew Pavlo, and Peter Boncz. 2018. Everything you always wanted to know about compiled and vectorized queries but were afraid to ask. *Proceedings of the VLDB Endowment* 11, 13 (2018), 2209–2222.

[13] Patrick Lewis, Ethan Perez, Aleksandra Piktus, et al. 2020. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*.

[14] Gengchen Mai, Krzysztof Janowicz, Ling Cai, Rui Zhu, Blake Regalia, Bo Yan, Meilin Shi, and Ni Lao. 2020. SE-KGE: A location-aware knowledge graph embedding model for geographic question answering and spatial semantic lifting. *Transactions in GIS* 24, 3 (2020), 623–655.

[15] Gengchen Mai, Krzysztof Janowicz, Rui Zhu, Ling Cai, and Ni Lao. 2021. Geographic question answering: challenges, uniqueness, classification, and future directions. *AGILE: GIScience series* 2 (2021), 8.

[16] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics* (Philadelphia, Pennsylvania) *(ACL '02)*. Association for Computational Linguistics, USA, 311–318. https://doi.org/10.3115/1073083.1073135

[17] 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*. Association for Computational Linguistics. https://arxiv.org/abs/1908.10084

[18] SeleniumHQ. 2024. Selenium WebDriver. https://www.selenium.dev/documentation/webdriver/ Accessed: [09-06-2024].

[19] SmartBear. 2024. Swagger: API Documentation Design Tools. https://swagger.io. Version 3.0.0.

[20] Haowen Xu, Xueping Li, Jose Tupayachi, Lian Jianming, and Femi Omitaomu. 2024. Automating Bibliometric Analysis with Sentence Transformers and Retrieval-Augmented Generation (RAG): A Pilot Study in Semantic and Contextual Search for Customized Literature Characterization for High-Impact Urban Research. (Oct 2024). Preprint, submitted on October 8, 2024.

[21] Linhao Ye, Zhikai Lei, Jianghao Yin, Qin Chen, Jie Zhou, and Liang He. 2024. Boosting Conversational Question Answering with Fine-Grained Retrieval-Augmentation and Self-Check. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2301–2305.

[22] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. 2019. Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675* (2019).

[23] Tianjun Zhang, Shishir G Patil, Naman Jain, Sheng Shen, Matei Zaharia, Ion Stoica, and Joseph E Gonzalez. 2024. Raft: Adapting language model to domain specific rag. *arXiv preprint arXiv:2403.10131* (2024).

[24] Yuchen Zhuang, Yue Yu, Kuan Wang, Haotian Sun, and Chao Zhang. 2024. Toolqa: A dataset for LLM question answering with external tools. *Advances in Neural Information Processing Systems* 36 (2024).