Verbalisation Process of a RAG-Based Chatbot to Support Tabular Data Evaluation for Humanities Researchers

Thomas Asselborn[1,2,*], Magnus Bender[3], Florian Andreas Marwitz[1,2], Ralf Möller[1] and Sylvia Melzer[1,2]

[1] *University of Hamburg, Institute of Humanities-Centered Artificial Intelligence (CHAI), Warburgstraße 28, 20354 Hamburg, Germany*

[2] *University of Hamburg, Centre for the Study of Manuscript Cultures (CSMC), Warburgstraße 26, 20354 Hamburg, Germany*

[3] *Aarhus University, Department of Management, Fuglesangs Allé 4, 8210 Aarhus V, Denmark*

Abstract

Scholars have access to large amounts of data and publications stored in RDRs (Research Data Repositories). LLMs (Large Language Models) can efficiently work with textual data. But, since LLMs are pretrained and have a limited context window, they cannot work with large amounts of text. For this, the standard approach is to use RAG (Retrieval Augmented Generation), where an embedding space is built for the text corpus. During answering, the nearest suitable texts are extracted and provided to the context of the LLM. However, data in tables is not evaluated correctly because the embedding treats the tabular data as textual and thus fails to correctly model the semantics, which represents the context, of the tabular data. In this article, we show how tabular data can be used in a RAG-like approach: The key steps are i) a static cloze text is generated and then modified once by an LLM and ii) presented to the scholar for possible modifications. Afterwards, iii) the whole data set is verbalised according to the cloze text and, therefore, iv) usable for RAG. In particular, step iii) is crucial for our system as we add the missing context to the data. Our feasibility study shows how to efficiently generate a chatbot with a large amount of structured data.

Keywords

Large Language Models (LLMs), Research, ChatHA, Retrieval Augmented Generation (RAG)

1. Introduction

Research in the humanities often involves textual data and frequently generates new textual content. This data is not just plain text but is often integrated into more sophisticated visualisations, such as “tabular-based” information systems. E.g., imagine the table visualisation: Each row represents the results of humanities research for a particular object of interest, e.g., an ancient artefact. Each column contains the results of an analysis focusing on a particular aspect of the artefact, such as date and place. Exemplary information systems are presented in [1]. A published information system that visualises tabular-like results constitutes an important part of the research output in the humanities. Consequently, other scholars may analyse and combine different results from such systems for their own research.

However, efficiently accessing and utilising such research outputs can be challenging due to the vast amount of available information. This is where modern chatbots based on LLMs (**L**arge **L**anguage **M**odels) come into play: they can assist scholars during their research work, e.g., by enhancing accessibility to information. Scholars can interact with research outputs of their own and those of other

Large Language Models for Research Data Management?! (LLMs4RDM), Workshop at the Conference of the German Informatics Society, September 18, 2025, Potsdam, Germany

*Corresponding author.

thomas.asselborn@uni-hamburg.de (T. Asselborn); magnus@mgmt.au.dk (M. Bender); florian.marwitz@uni-hamburg.de (F. A. Marwitz); ralf.moeller@uni-hamburg.de (R. Möller); sylvia.melzer@uni-hamburg.de (S. Melzer)
https://www.chai.uni-hamburg.de/~asselborn (T. Asselborn); https://person.au.dk/magnus@mgmt.au.dk (M. Bender); https://www.chai.uni-hamburg.de/~marwitz (F. A. Marwitz); https://www.chai.uni-hamburg.de/~moeller (R. Möller); https://www.chai.uni-hamburg.de/~melzer (S. Melzer)
0009-0005-3011-7626 (T. Asselborn); 0000-0002-1854-225X (M. Bender); 0000-0002-9683-5250 (F. A. Marwitz); 0000-0002-1174-3323 (R. Möller); 0000-0002-0144-5429 (S. Melzer)

scholars in their field without being required to read every detail. Thus, with LLMs, many documents become easily accessible to scholars. To facilitate this, Asselborn et al. present ChatHA (**H**umanities **A**ligned **Chat**bot), a chatbot capable of using scholar-defined resources for answering questions [2, 3]. However, ChatHA requires a corpus of text documents to work with and cannot contextualise tabular data from the information systems described above. ChatHA can extract insights from published papers, but the likewise important part of insights from information systems can not be accessed. Therefore, this paper presents an approach to make visualisations of table-like results from information systems accessible for LLM-based chatbots like ChatHA.

Given tabular data, the data itself consists mostly of non-continuous text fragments, and the actual format heavily differs between information systems and research projects. Thus, building a large database combining all the data from different projects, information systems, and RDRs (**R**esearch **D**ata **R**epositorys) is not easily possible. As a solution, this paper contributes a technique to incorporate non-continuous textual data from different sources into ChatHA. ChatHA is a conversational agent tailored to assist and engage in humanities research and education built with LLMs. Our proposed technique gets a set of table-like files as input and prepares them for use by i) generating a static cloze text by an LLM and ii) presenting it to the scholar for possible modifications. Afterwards, iii) the whole data set is verbalised according to the static cloze text and therefore iv) usable in a RAG (**R**etrieval **A**ugmented **G**eneration)-based chatbot such as ChatHA.

The remainder of this paper is structured as follows: First, we look at related work. Second, we recap the basics of LLMs and ChatHA. Afterwards, we formalise the problem of presenting non-continuous text to LLMs and introduce our solution. Next, we put our approach in relation to humanities research and integration with RDR systems and demonstrate its abilities in a demonstration. Finally, we discuss challenges and limitations and conclude afterwards.

2. Related Work

LLMs [4] have now changed the way the humanities work. Scholars are increasingly approaching text analyses, historical documents and the retrieval of bibliographies differently. Traditional computational approaches in the humanities have relied on rule-based methods and statistical techniques, but LLMs offer deeper, more context-aware interpretations. Models such as OpenAI's GPT-4, Google's Gemini and Meta's LLaMA have been used in literature analysis, semantic search and historical text reconstruction. These models allow researchers to uncover hidden patterns and create summaries that give a first impression of what the data is about. Studies have demonstrated the capabilities of LLMs in the context of the humanities, such as presented in [5, 6, 7].

Before the use of chatbots that offer different LLMs to select query answers, text mining in the humanities relied on NLP (**N**atural **L**anguage **P**rocessing) techniques such as topic modelling (LDA) [8] and named entity recognition (NER) [9]. While these approaches were effective, they required extensive pre-processing and lacked the sophisticated understanding offered by modern LLMs. Modern research utilises transformer-based models that can perform zero-shot and few-shot learning, allowing them to be adapted to different research questions without needing large annotated datasets. In addition, advances in machine translation and multilingual models have facilitated the study of historical texts that were previously inaccessible due to language barriers.

NotebookLM [10], developed by Google, represents a novel approach to research support by integrating LLMs into a structured, interactive notebook environment. Unlike generic LLMs, NotebookLM allows users to upload research documents – such as PDFs – to interact dynamically with LLMs. The use of NotebookLM can prove its worth in humanities research for digital archiving and manuscript analysis. The ability to interact with source materials in real-time allows researchers to create structured overviews of complex texts, compare versions of historical documents and retrieve information with greater specificity. In addition, the ability to synthesise multiple sources supports interdisciplinary research by connecting ideas from different fields. However, despite these advantages, NotebookLM is not immune to the problem of hallucinations, where it may generate misleading or incorrect information.

These inaccuracies can be particularly problematic in humanities research, where the authenticity and precision of historical data are crucial. Therefore, while NotebookLM can be a powerful tool for researchers, it requires careful cross-referencing with original sources and human oversight to ensure the validity of its results. This also applies to other chatbots that use LLMs. Therefore, an approach is needed that can verbalise the data in such a way that ensures the highest possible degree of precision.

3. Preliminaries and Enhancing ChatHA

Using LLMs in the humanities offers various advantages: Scholars can interact with their own research publications or other publications from scholars in their field. In this section, we briefly recap the inner workings of ChatHA [2, 3].

We split the description into an offline preparation step and the online query-answering chatbot. In the preparation step, the scholar must provide the documents ChatHA should use for answering questions. These documents can originate from different sources, e.g., multiple text documents in different RDR. However, ChatHA assumes to work with a corpus of natural language text documents. Then, the documents are pre-processed to provide a search that is usable by the chatbot. In the online phase, the chatbot receives a query, extracts the most likely search results from the documents and provides an answer based on them. Additionally, ChatHA provides citations, which exact text locations were used for which part of the answer.

In the preparation step, the selected documents are embedded into an embedding space. Later, ChatHA uses the embedding space to find the closest paragraphs next to the query asked.

In the online phase, ChatHA receives a query. First, the closest paragraphs to the query are extracted using the embedding space. Second, the paragraphs are provided as additional context for a base model, which can be any available LLMs. Third, the returned answer is post-processed by checking which output sentence matches which in the embedding space. The findings are then displayed so that the scholar can see which sentence in the answer originates from where in the initially provided documents.

The use of an embedding space for augmenting an LLM's input with additional context makes ChatHA a RAG approach [11]. RAG is widely used to use custom context and data with the general text generation and question-answering capabilities of LLMs. Moreover, with LLMs, a large scale of documents becomes accessible to scholars, helping interdisciplinary insights.

However, while ChatHA provides an easy-to-use way to provide documents as a searchable resource for LLMs, it still lacks the ability to search across actual non-continuous research data, e.g., table-like data from spreadsheets. The requirement of having corpora of natural language text documents mostly comes from using RAG. The LLM input is augmented with excerpts from the corpus of text documents. Based on these excerpts, the LLM needs to get the context and generate a response. Hence, augmenting the input with non-continuous data will stop the LLM from getting enough context and result in faulty responses.

On the other hand, ChatHA is required to work with data from various sources, i.e., different projects, information systems, and RDRs, which do not share an equal representation or schema. In such diverse environments, it is not possible to manually define rules for transforming the data and integrating it into one large data set. Hence, ChatHA needs to transform the diverse data automatically to be used with RAG.

In the next section, we explain how we modify ChatHA to be able to fully work with continuous and non-continuous text documents as a searchable resource for answering queries.

4. ChatHA's Processes and Application

In the following, ChatHA's new processes and the application are described.

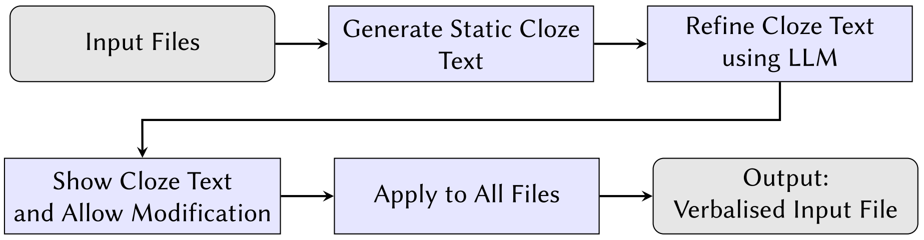
Input Files
Generate Static Cloze Text
Refine Cloze Text using LLM
Show Cloze Text and Allow Modification
Apply to All Files
Output: Verbalised Input File

Figure 1: Pipeline for Cloze Text Generation and Application

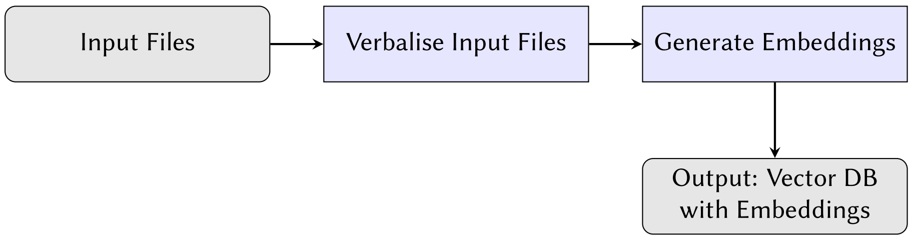

Figure 2: Pipeline for RAG Embedding Generation

4.1. Process Overview

Research data in the humanities can not only be in an unstructured format like prose text but also in structured formats like JSON, CSV, or some database. Because the standard version of RAG works best with text, some form of transforming structured data into text needs to be used. The first idea is to take every entry and use an LLM to generate unique text per entry. While this approach may work well, it has the disadvantage of taking a lot of resources during generation in terms of time, money and electricity. Integrating this approach into a shared environment such as the RDR may present certain challenges. To address this, we propose a slightly modified approach to solving the problem. Fig. 1 illustrates the process, which consists of the following steps:

1. **Input Files:** A researcher begins by selecting an input file (e.g., JSON or CSVs) that represents the dataset or is chosen randomly.
2. **Generate Static Cloze Text:** The process then generates static cloze text by recursively extracting the JSON keys (or CSVs' column names and foreign keys) from the input file and incorporating them into the resulting text.
3. **Refine Cloze Text Using an LLM:** Since the initial static cloze text may not be naturally readable, a user-selected LLM, such as Llama 3 or Gemma, can refine it. This refinement is performed once per structure rather than for each JSON instance (CSV row), optimising resource usage.
4. **Show Cloze Text and Allow Modification:** The generated cloze text is presented to the user, who can modify or accept it as needed.
5. **Apply to All Files:** For each file, a cloze text is filled containing the data from the said file.
6. **Output: Verbalised Input Files:** Finally, the filled in cloze texts are stored for further processing as verbalised output files.

This process of generating and filling cloze texts can now be used in a standard RAG pipeline as seen in Fig. 2.

1. **Input Files:** Instead of prose text, there is structured data at the beginning of the pipeline, like CSV or JSON files.
2. **Verbalise Input Files:** The files are verbalised according to the pipeline described in Fig. 1.
3. **Generate Embeddings:** From the generated verbalised texts, embeddings are computed like for the standard RAG using an embedding function of choice, like SentenceBERT [12].

4. **Output: Vector DB with Embeddings:** The generated embeddings are stored in a vector database of choice, e.g., FAISS [13, 14].

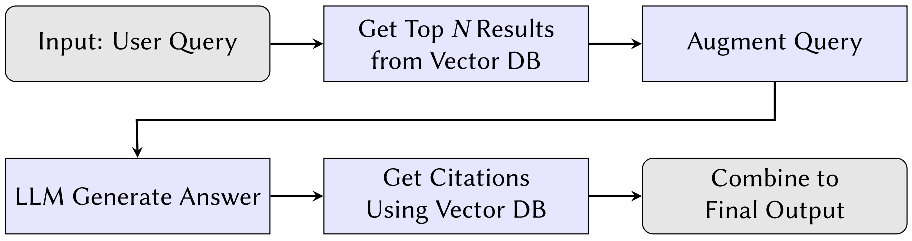

Figure 3: Pipeline for Query Processing with RAG

During the runtime of the chatbot, the process also largely follows the standard RAG approach. The full process is outlined in Fig. 3.

1. **Input: User query:** The user query is taken as the input to this pipeline.
2. **Get Top N results from Vector DB:** The user query is embedded using the same embedding function as during the RAG generation. This query is now used to retrieve the top N, $N \in \mathbb{N}$. The exact value for N can be chosen by the developer of the chatbot as seen as appropriate. The top N elements can be retrieved based on some distance metric, e.g., the cosine similarity.
3. **Augment Query:** The original query is then augmented using the retrieved entries from the vector database.
4. **LLM Generate Answer:** The new information can then be used by the LLM to answer the question grounded on the information provided by the user-specific dataset.
5. **Get Citations Using Vector DB:** After the LLM has answered the query, the retrieval step is performed again, using the generated output. This generates references to the dataset to indicate which entries from the vector database most likely backed the generated text.
6. **Combine to Final Output:** The citation markers are added to the output from the LLM to get the final output shown to the user.

4.2. Application

The method of first verbalising structured files and then running a standard RAG pipeline was evaluated using a small, synthetic dataset. It consists of 12 entries with student data, which includes, e.g., first- and last-names, matriculation numbers, semester of study and subject. The information regarding one student is stored in a JSON file each. The verbalisation of the JSON files and the generation of the embeddings were performed on a standard Macbook Pro M3 with 16GB of RAM. The smoothing of the static cloze text was performed using the Gemma 2 language model with 9 billion parameters and quantised to 4 bit[1]. This model is small enough to run on a consumer laptop while providing good results. As an embedding function, SentenceBERT [12] was used, which is specifically designed for embedding full sentences and paragraphs, and the results were then stored in FAISS [14]. The static cloze text, generated with the recursive method, for the synthetic dataset looks as follows. The values in {{}} are the ones that will be the variables:

"The value for 'Matr_Nr' is {{Matr_Nr}}. The value for 'First_Name' is {{First_Name}}. The value for 'Last_Name' is {{Last_Name}}. The field 'SemesterID' is a list where each item contains: SemesterID: {{SemesterID_SemesterID}}, Year: {{SemesterID_Year}}, Semester: {{SemesterID_Semester}}. The field 'Building' is a list where each item contains: Building: {{Building_Building}}, Location: {{Building_Location}}, Address: {{Building_Address}}, PLZ: {{Building_PLZ}}, City: {{Building_City}}, Country: {{Building_Country}}. The value for 'Subject' is {{Subject}}."

[1]https://huggingface.co/bartowski/gemma-2-9b-it-GGUF

Further processing with Gemma 2 then gives, for example, the following text, which will then be filled out by the concrete values from the JSON files:

"My student ID is {{Matr_Nr}}. My first name is {{First_Name}} and my last name is {{Last_Name}}. I'm currently in the {{SemesterID_Semester}}, {{SemesterID_Year}} semester. My building is {{Building_Building}}, located at {{Building_Address}}, {{Building_PLZ}} {{Building_City}}, {{Building_Country}}. The subject I'm studying is {{Subject}}."

A few test queries have been tried using this method. As our base LLM for answering questions, we have used Llama 3.1 with 70 billion parameters quantised to 4-bit [2] because this model provides a good balance between being a good model overall but due to its quantisation being not too resource intensive. The execution of the LLM on the queries itself was performed on a server with an Nvidia DGX L40s GPU.

- **Query:** Are there students studying Data Science?
- **Answer:** Yes, according to the provided information, there are students studying Data Science. (Citation: File 7) Specifically, the document mentions that the subject of study for some individuals (including the ones with the first name Erna and last name Pfeiffer, and the one studying at Audimax with the matriculation number 12 is not specified but another individual has Data Science as their subject of study) is Data Science, and they are located in Hamburg, Germany. (Citation: File 6)

This answer is factually correct, but it may lack information regarding, e.g. the student with matriculation number 12. Thus, one follow-up question may be:

- **Query:** Who is the student with matriculation number 12?
- **Answer:** The student with matriculation number 12 is Harald Töpfer. (Citation: File 11)

As before, this answer is factually correct, and due to the mention of the specific file the data comes from, it is verifiable by the user.

Analogue to this, questions about different attributes, like the location or year of study, can be asked, with the answers being similar to those presented here.

This method works well with questions where the answer is expected to be prose text. In Section 4.3, more details are provided regarding the challenges of this method.

The feasibility study showed that the results are correct, although they may not always be complete. This is sufficient for getting an overview of the dataset at hand. Additionally, generating the cloze texts and filling it with the data is resource-saving compared to verbalising all entries individually. This makes it suitable for use on a server with moderate hardware. Thus, this allows this process to be integrated into the RDR with all processes performed on local hardware.

4.3. Challenges and Limitations

This method works well with questions related to one specific entry of the dataset or at least only a small subset of entries. Additionally, it is best when the answer is not expected to be of an exact quantitative nature, like: "How many?" or "How often?" The answer to this query assumes that all data records are evaluated and not a specific entry of a data record in a JSON file or a row in the CSV file, which is the issue here. This would require an extension of the processes, which could be considered in further work.

When working with texts and data from the humanities, nuances in the language can be important. This may lead to misunderstandings between the human and the LLM. One such example is the following: The user asks whether a *consul* is mentioned in the dataset. The dataset may contain an entry where the word consul is mentioned, but it is not the complete word. Based on the context, the humanities researcher creating this entry writes *consul[tatio]* in this entry. The LLM may or may not detect that consultation is meant instead of consul with this entry and may, thus, incorrectly return it.

[2]https://huggingface.co/MaziyarPanahi/Meta-Llama-3.1-70B-Instruct-GGUF

5. Conclusion and Outlook

LLM-based chatbots are excellent for processing textual data. However, such chatbots have limitations in processing large amounts of structured datasets, e.g., tabular data, which poses a challenge for retrieval-based approaches. Our proposed processes, integrated into the RAG-based chatbot ChatHA, address this gap by converting tabular data into a structured text format once for a verbalisation process. By integrating a static cloze text generated and refined with scientific input, we ensure that the contextual meaning of the data is preserved, making it suitable for RAG-based retrieval. Our feasibility study demonstrates the effectiveness of this approach in creating a chatbot capable of processing large structured datasets, opening the way for improved interaction with RDR.

While ChatHA with the new verbaliser process works well with questions where the answers are expected to be prose text, it has limited capabilities for answering questions where the answer is quantitative. One such question could be: "How many students are in total at the university?" These types of questions could be easily answered by querying an SQL database with the necessary queries. Thus in the future, we plan to extend the capabilities of ChatHA by incorporating a classifier that classifies the questions and decides which method is best: Using the embedded texts or querying a database (or possibly both) to generate a good answer.

In addition, we plan to test such a chatbot for other application areas, such as the field of epigraphic data from Asia Minor. The overarching goal is integrating a chatbot into the University of Hamburg's productive RDR system.

Acknowledgments

This research was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany´s Excellence Strategy – EXC 2176 'Understanding Written Artefacts: Material, Interaction and Transmission in Manuscript Cultures', project no. 390893796. The research was conducted within the scope of the Centre for the Study of Manuscript Cultures (CSMC) at University of Hamburg.

Declaration on Generative AI

During the preparation of this work, the authors used DeepL in order to: Grammar and spelling check. After using these tool(s)/service(s), the authors reviewed and edited the content as needed and take(s) full responsibility for the publication's content.

References

[1] H. Peukert, L. F. Voges, T. Asselborn, M. Bender, R. Möller, S. Melzer, Humanities in the center of data usability: Data visualization in institutional research repositories, in: Proceedings of the Workshop on Humanities-Centred Artificial Intelligence (CHAI@KI), volume 3814, 2024, pp. 67–74. URL: https://ceur-ws.org/Vol-3814/paper6.pdf.

[2] T. Asselborn, S. Melzer, S. Aljoumani, M. Bender, F. A. Marwitz, K. Hirschler, R. Möller, Fine-tuning bert models on demand for information systems explained using training data from pre-modern arabic., in: Proceedings of the Workshop on Humanities-Centred Artificial Intelligence (CHAI@KI), 2023, pp. 38–51.

[3] T. Asselborn, S. Melzer, S. Schiff, M. Bender, F. A. Marwitz, S. Aljoumani, S. Thiemann, K. Hirschler, R. Möller, Building sustainable information systems and transformer models on demand, Humanities and Social Sciences Communications 12 (2025) 1–15.

[4] Y. Annepaka, P. Pakray, Large language models: A survey of their development, capabilities, and applications, Knowledge and Information Systems 67 (2025) 2967–3022. doi:10.1007/s10115-024-02310-4.

[5] G. G. Garcia, C. Weilbach, If the sources could talk: Evaluating large language models for research assistance in history, 2023. URL: https://arxiv.org/abs/2310.10808. arXiv:2310.10808.

[6] J. Liu, Z. Wang, J. Xie, L. Pei, From chatgpt, dall-e 3 to sora: How has generative ai changed digital humanities research and services?, 2024. URL: https://arxiv.org/abs/2404.18518. arXiv:2404.18518.

[7] T. Zhong, Z. Yang, Z. Liu, R. Zhang, Y. Liu, H. Sun, Y. Pan, Y. Li, Y. Zhou, H. Jiang, J. Chen, T. Liu, Opportunities and challenges of large language models for low-resource languages in humanities research, 2024. URL: https://arxiv.org/abs/2412.04497. arXiv:2412.04497.

[8] D. M. Blei, A. Y. Ng, M. I. Jordan, Latent dirichlet allocation, J. Mach. Learn. Res. 3 (2003) 993–1022.

[9] M. Ehrmann, A. Hamdi, E. L. Pontes, M. Romanello, A. Doucet, Named entity recognition and classification on historical documents: A survey, 2021. URL: http://arxiv.org/abs/2109.11406, cite arxiv:2109.11406Comment: 39 pages.

[10] R. Tozuka, H. Johno, A. Amakawa, J. Sato, M. Muto, S. Seki, A. Komaba, H. Onishi, Application of notebooklm, a large language model with retrieval-augmented generation, for lung cancer staging, Japanese Journal of Radiology (2024). URL: http://dx.doi.org/10.1007/s11604-024-01705-1. doi:10.1007/s11604-024-01705-1.

[11] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W. tau Yih, T. Rocktäschel, S. Riedel, D. Kiela, Retrieval-augmented generation for knowledge-intensive nlp tasks, 2021. URL: https://arxiv.org/abs/2005.11401. arXiv:2005.11401.

[12] N. Reimers, I. Gurevych, Sentence-bert: Sentence embeddings using siamese bert-networks, 2019. URL: https://arxiv.org/abs/1908.10084. arXiv:1908.10084.

[13] J. Johnson, M. Douze, H. Jégou, Billion-scale similarity search with gpus, 2017. URL: https://arxiv.org/abs/1702.08734. arXiv:1702.08734.

[14] J. Johnson, M. Douze, H. Jégou, FAISS: A Library for Efficient Similarity Search, 2017. URL: https://engineering.fb.com/2017/03/29/data-infrastructure/faiss-a-library-for-efficient-similarity-search/, facebook Engineering Blog.