ChatPapers: An AI Chatbot for Interacting with Academic Research

Max Dean
School of Computing
Ulster University
Belfast, Northern Ireland
MaxDean140@Gmail.com

Raymond R. Bond
School of Computing
Ulster University
Belfast, Northern Ireland
RB.Bond@Ulster.ac.uk

Michael F. McTear
School of Computing
Ulster University
Belfast, Northern Ireland
MF.Mctear@Ulster.ac.uk

Maurice D. Mulvenna
School of Computing Ulster University
Belfast, Northern Ireland
MD.Mulvenna@Ulster.ac.uk

2023 31st Irish Conference on Artificial Intelligence and Cognitive Science (AICS) | 979-8-3503-6021-9/23/$31.00 ©2023 IEEE | DOI: 10.1109/AICS60730.2023.10470521

***Abstract* – A growing and significant number of computer science related papers are being published; hence it is challenging to keep up with the latest research. This paper describes the development of a large language model (LLM) augmentation chatbot and user interface that provides responses to research queries in the domain of computer science. Around 200,000 computer science research papers from arXiv were** embedded**, resulting in ~11 million vectors (based on 'chunks' from the papers). Each vector is comprised of 384 numbers/dimensions. Technologies used include Langchain, a Vector Database, and Semantic Searching with document / query embeddings. The** chatbot **was tested using 30 sample questions that could be asked by computer science students across several topics and from different education levels (i.e., BSc, MSc and PhD level). The responses from this chatbot were compared with those from GPT-4. The responses with and without prompting were also compared. Readability metrics (Flesch-Kincaid and Coleman-Liau) were used to compare the responses from this LLM with GPT-4. Retrieval Augmented Generation Assessment (RAGAS), a novel LLM self-evaluation method was used to evaluate the system. We observed that the developed system provides more suitable responses to the user based on the readability level at which the questions were asked.**

Keywords—large language model, chatbot, retrieval augmented generation, Langchain, vector database, semantic search, GPT-4, Retrieval Augmented Generation Assessment, Readability metrics, Flesch-Kincaid, Coleman-Liau

I. INTRODUCTION

This paper discusses the creation of a new large language model (LLM) powered chatbot system designed to assist users in learning from and researching computer science research papers. The system utilizes the latest emerging technologies, including agent abstraction to provide users with relevant information. The intention is to ease the process of understanding research papers, making them more accessible to students who are completing research assignments or PhD students who are exploring literature. Under the ***Methodology*** section we discuss both the engineering / technological methods that are used to produce the desired system, and the scientific methods that are used to assess whether the system is fit for purpose. Key elements of system design are discussed under Methodology > Engineering; including the system architecture and the technology stack, the frameworks and abstractions used[1]. The aim of the research described here is to create and test an LLM and chatbot system that provides responses to research queries in the domain of computer science, and to contribute to the literature in the emerging field of *Retrieval Augmented Generation*.

II. EXISTING WORK

Although LLMs display a remarkable ability to generate human-like text, they are limited in that they can only answer queries accurately that relate to content learned from their training data up to the date when the training took place. In order to address this, researchers are exploring ways to augment LLMs with external knowledge in the form of embeddings. In this way the user's query is used to retrieve a relevant document and the LLM uses the document to generate a fluent and accurate response to the query.

The implementation of a Research Paper Recommendation system that used a mechanism known as 'Weighted Citation Network' which used word and document embeddings of research papers in conjunction with a graph network of (unidirectional) citations is described in [1]. This mechanism was then used to accept a new document (research paper) and recommend similar papers, returning a similarity score. Another paper, [2], discusses the training of a model that *jointly* uses the Title and Abstract of research papers to create a vector embedding / representation for the papers via a fine-tuned BERT (Bidirectional Encoder Representations for Transformers) model in conjunction with a custom GRU (Gated Recurrent Unit) based model. A 2019 paper [3] released by Microsoft employees outlines the use of 'Microsoft Academic' which is a semantic search database with 200+ million indexed research papers. Their

[1] The chatbot system described in this paper is hosted at https://chatpapers.io

recommendation system is based on word embedding vectors in combination with TF-IDF (Term Frequency - Inverse Document Frequency) weightings to create a vector representation of a full paper. This study used user surveys to evaluate the quality of recommendations from the system. Another study [4] proposes a recommendation system where the user can input text of their choosing and have it vectorised by SciBERT and uses K-Means clustering to classify and recommend similar papers.

Once the embeddings have been created, the next step is to use them to retrieve a relevant document and generate an appropriate response. Retrieval-Augmented-Generation (RAG) is a new AI framework for improving the quality of LLM-generated responses through the use of external knowledge sources. Using RAG ensures that the responses of the LLM are accurate and up-to-date while taking advantage of the LLM's ability to generate fluent natural language text [5].

III. METHODOLOGY

A. Scientific Methods

There are a range of common place methods used to assess the performance of LLMs, both as a loss function during the training process and to produce metrics by which different LLMs can be compared. Some conventional practices include the Perplexity and BLEU (Bilingual Evaluation Understudy) scores (which measure the likelihood of the next output token and compare to a dataset / corpus) as well as MMLU (Massive Multitask Language Understanding) and others which measure a model's aggregate task specific performance. Human focused tests can also be used such as Standardised Acceptance Tests, Bar examinations, or even coding challenges (*Leetcode*). [6]

However, these methods *1)* are not entirely useful in the examination of document retrieval systems, and *2)* have already been used to compare / evaluate the underlaying model in the system implemented here. Thus, it is necessary that we devise our own methods of quantifying key aspects of the systems performance. These aspects are:

1. The system's ability to discern the level at which the users' input has been written.
2. The system's ability to retrieve appropriate documents via semantic search.
3. The system's ability to return responses at the same readability level as the input / chat history.

Given the domain in context (research literature in computer science) - the method used to test the system has been limited to queries that might be asked by 'BSc', 'MSc', and 'PhD' students. A list of 10 questions that may be asked of the proposed system were devised with each question being asked at 3 different difficulty levels ('BSc', 'MSc', and 'PhD'). The questions cover a range of topics from Deep Learning to Computer Networking, that we could expect a student of Computer Science to ask of a chatbot. See evaluation questions [7].

1) Readability Metrics

The Flesch-Kincaid (FK) and Coleman-Liau (CL) readability metrics were used to assess the system inputs and outputs. See equations E1 and E2. Both metrics aim to produce a *grade level* at which an individual needs to be in order to *comprehend* the given text. Although grade-levels stop at 12, these metrics can be extended indefinitely, and we can define university grade levels as follows:

- 'BSc' = 15.5
- 'MSc' = 16.5
- 'PhD' = 17.5

$$FK(text) = 206.835 - 1.015\left(\frac{total\ words}{total\ sentences}\right) - 84.6\left(\frac{total\ syllables}{total\ words}\right) \quad (1)$$

$$CL(text) = 0.588 * L - 0.296 * S - 15.8$$
$$L = avg\ letters\ per\ 100\ words$$
$$S = avg\ sentences\ per\ 100\ words \quad (2)$$

2) RAGAS Evaluation

RAGAS (Retrieval Augmented Generation Assessment) is a recent evaluation framework for assessing the performance of retrieval augmented LLMs. The RAGAS metric is the harmonic mean of three scores (that may also be considered individually): 1. Answer Relevancy 2. Context Relevancy 3. Faithfulness.

The answer relevance score is generated by using an LLM to reverse generate questions for each answer in the dataset, then an *embedding* model is used to vectorise both the real questions and the generated questions. A cosine similarity between the real/generated pairs is taken and the mean across the dataset is calculated. This is illustrated in figure 1. The context relevancy metric uses an LLM as a *Cross-Encoder*, measuring the similarity between each context and its corresponding question. The faithfulness metric is a simple mechanism that generates a *statement* about each question-answer pair using an LLM, and then calls the LLM again to ask if the statements are *supported* by the contexts. [8]

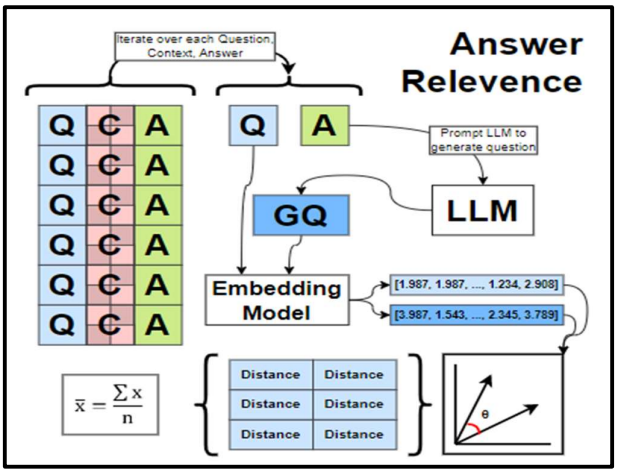

Fig. 1. RAGAS Answer Relevance flow.

$$Harmonic\ Mean = \frac{3}{\left(\frac{1}{AR} + \frac{1}{CR} + \frac{1}{F}\right)} \quad (3)$$

$$AR = Answer\ Relevence$$
$$CR = Context\ Relevence$$
$$F = Faithfulness$$

The harmonic mean is taken of the three scores as it weights outliers heavily. So, if our RAG pipeline outputted answers and contexts that were relevant but the answers were false, the RAGAS score would be closer to 0 than 2/3. [8]

B. Engineering Tools & Techniques

1) Software Architecture

The software architecture of the current chatbot follows the client-server paradigm. A Single Page Application developed with FlutterFlow is the *front-end* or *client* where the user can log in, create, and delete chats, and interact with the chatbot through a chat interface. The generation of response messages occurs on a Google Cloud Function, this must be done server-side as the client machine may not be computationally powerful enough to run the embedding model. The chatbot is built with the LangChain framework and uses a ZeroShotReact Agent, accompanied by seven tools. The ReAct agent paradigm uses an LLM to make a series of notes and develop its understanding, then with the help of an ***output parser*** it chooses a tool from a list provided to enable it to take the appropriate action to satisfy the user's query.

2) Indexing / Ingestion

During the vectorisation process of a paper - when new papers are indexed - a script downloads the research paper (PDF) and the associated metadata, then extracts all the text from the file. The text content of the file is split into *chunks* of roughly 1000 characters, and these chunks are vectorised using a *transformer encoder*. The 'all-MiniLM-L6-V2' model from HuggingFace sentence-transformer library is used for this and all other vectorisations in the bot's operation. This model was chosen for its high acclaim in the NLP (Natural Language Processing) community, and its small file size (~90MB) allows it to make efficient inferences. The training data for this model includes the S20RC dataset from *Semantic Scholar* – making up some 200+M of the 1.2B Data items in the training set. Though testing is outside the scope of this study, we may assume that this contributes to the model's ability to encode scientific papers / writing.

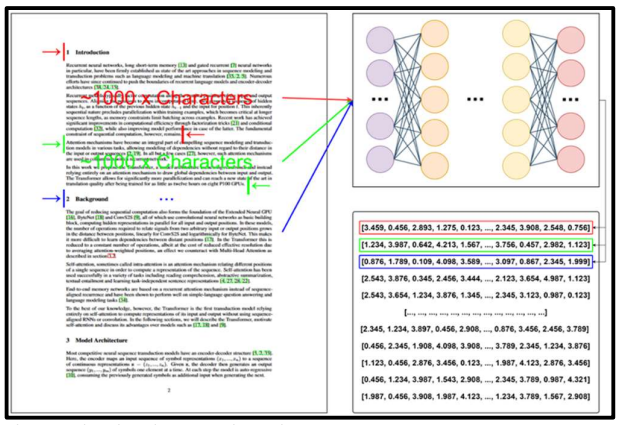

Fig. 2. Showing document ingestion process.

The ingestion script takes the 384-dimensional vector encoding of each chunk and stores them in a vector database. Postgres and the pgVector plugin were used for this via a SaaS database product – Supabase – which also stores the necessary metadata that some of the Agent tools utilise. We had the option during the text splitting process to include some overlap in the chunks but this was decided against in the interests of keeping the total size of the database produced smaller.

3) Querying / Searching

When the agent uses a tool that searches the vector database, it first vectorizes its query, for example, "Deep learning network architectures". This search term is vectorized with the same model (all-MiniLM-L6-V2) that was used for the text chunks from the research papers. This provides a 384-dimensional vector that can be used for a 'similarity' or a 'semantic' search within the database. To achieve this, pgVector takes the input vector and gathers the k nearest vectors in the database using cosine similarity. The value for k can be set by the agent, that is, the agent can choose how much relevant information to gather from the search. pgVector then returns the text chunk and associated metadata to the agent tool that made the search, and the tool returns that data to the agent's *observation* step. This means that those chunks of text from the similarity search are inserted into the agent's context window. [9]

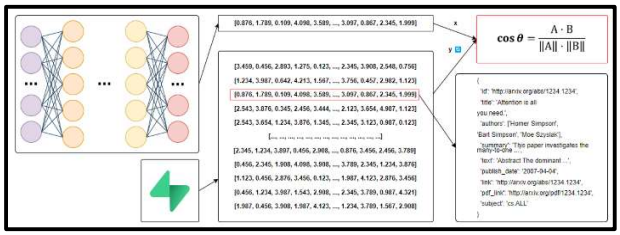

Fig. 3. S howing document query process.

The Euclidean distance may also be used in a vector similarity search such in the current work. However, Cosine distance is used because the search term (which is decided by the agent) may be less complex than the chunks of text we seek to have returned.

4) Agent

There are several agent design paradigms available when using the LangChain framework; Plan-and-Execute, Self-Ask, and ReAct. These are of course all just abstractions achieved through prompt engineering, though their use is necessary to give the chatbot structure and governance. Plan-and-Execute agents are best suited to tasks that require multiple planned steps to achieve its goal, while Self-Ask is good for reasoning without the use of a tool. A ReAct (Reason + Act) agent was chosen in this situation because of its ability to interact with tools to provide responses with accurate information. The ReAct Agent takes a series of sequential steps that may or may not require the use of a tool. In the LangChain framework there are some variations in the designs for the ReAct agent: Zero-Shot - the most basic kind (does not use conversation history), Structured Input – which allows more complex tool argument schemas, and Conversational – which can incorporate a prior chat history. For this application a combination of structured input and *conversational* ReAct agent was used. Some of the agent's search tools need to be able to query with search terms and set the k parameter; thus, structured input is necessary. And, as the application is a chatbot, the chat history should also be included in the agent's prompt. [10] [11] [12]

The operation of a ReAct agent can be understood in terms of the thought-action-observation cycle seen in figure 4. A new agent is instantiated to respond to each new human message in a chat, its first step is to accept the user message and prepare its first LLM prompt and then make the first LLM call i.e., the '**thought'** stage. An *output parser* searches the response for either a *stop condition* + *final answer* or an ***action*** + *action input* (if these aren't present an error is caused which must be handled). Some logic allows the agent to call a tool method with the appropriate arguments if that path is chosen, and the tool's output becomes the '**observation'**. The LLM is then prompted for another thought and the cycle either begins again, or the stop condition is met, and the answer is returned to the user and written into the chat history database. [10]

Figure 5 shows a basic example of the output at each of the three stages, where the agent was able to a discern enough information to satisfy the user after just one cycle. In this cycle it uses the getRecentPapers tool and passes the string ***"IoT in care applications"*** as the search query.

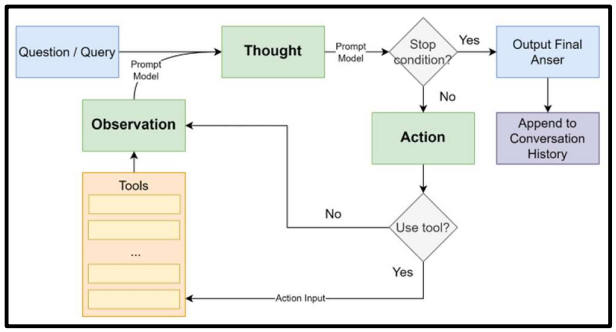

Fig. 4. ReAct agent though process.

```
Question: Tell me about the latest research relating to IoT in care applications.
Thought: I should use the getRecentPapers tool to search for papers relating to IoT in care applications.
Action: getRecentPapers
Action Input: IoT in care applications
Observation: "[{'id':'0708.0624v1','title':'ADS-Directory…',…},…{'id':'0804.3500','title':'Natural Pseudo…',…}]"
Thought: I know the final answer.
Final Answer: There have been many recent developments in...
```

Fig. 5. ReAct agent thought process in agent prompt 'scratch pad'.

5) Tools

Seven tools were implemented for the agent to use, table 1 names these along with the 'description' provided to each. These descriptions become part of the prompt during the Thought → Action LLM query. Thus, the LLM uses the description to decide which tool to use next (or that no tool may be used) to achieve its goals. The purpose of each tool can be categorised as **exploration** or **understanding;** both *search* and *getRecentPapers* are exploration tools. This means that they are used by the agent to help the user discover new research in an area, by semantically searching it with vector similarity. The other tools serve either to build the user's understanding of research papers and areas, or to help the agent gather more information to respond to the user as such. *getFullPaperSummary* is used to summarise entire research papers, while *askPaperQuestion* can retrieve specific pieces of information from a research paper.

TABLE I. AGENT TOOLS

Name	Description
search	Searches top-k research papers related to a text query. Returns information about the papers including ID, title, authors, release date, and summary which is an overview of the paper that does not go into details (same as getRecentPapers tool). The output of this tool is often enough to answer a question.
getFullPaper Summary	Given an ID (FORMAT: [REDACTED] or [REDACTED]), this tool returns a custom summary of the full paper. NOTE: This tool is very expensive to use, it must only be used if the user specifically asks for a summary.
getRecent Papers	Searches recent research papers related to the user's query. Returns information about the including title, authors, release date and summary which is an overview of the paper that does not go into details (same as search tool).
askPaper Question	This tool should be used to answer all questions about specific research in a paper. If correct information is not found, we can increase k and try again.
Get Information FromPaper	This tool should be used to get information from specific research in a paper. If correct information is not found, we can increase k and try again.
getPrior Context	This tool can retrieve the preceding chunk to any given chunk of text that isn't the first. Useful for when a chunk of text starts in the middle of an important piece of information. Given a context ID (format = an integer, e.g. 12345678).
Get Subsequent Context	This tool can retrieve the succeeding chunk to any given chunk of text that isn't the first. Useful for when a chunk of text ends in the middle of an important piece of information. Given a context ID (format = an integer, e.g. 12345678).

6) Prompts

The agent scratchpad (figure 5) is part of the LLM prompt, with every cycle of the thought process it is sent to the LLM and updated. The prompt however, must also contain other important information: 1) A system message to tell the LLM *who it is* and what tools it has access to. 2) A history of the

chat messages that have been exchanged already. 3) The latest message from the user in the chat. 4) An instruction set, explaining how to take steps in the thought process. 5) And lastly the scratchpad area – figure 5. In the system message we can prompt the agent to take a prior step, in which it decides what type of user it is speaking too. Thus, we can encourage the agent to make its final output at the understanding level of a BSc, MSc, or PhD level student. However, as this is persisted in the agent prompts, it will also influence the inputs passed to the tools. That is, if the agent decides that is being spoken to at a BSc level of understanding, it will use more basic search terms when it calls its tools and searches the vector database.

IV. RESULTS

A. Readability

After collecting a sample of answers (and corresponding contexts) we can visualise how well GPT4 and ChatPapers bot were able to match the user question readability level. Figures 6 and 7 show box plots for both the Flesch-Kincaid and Coleman-Liau readability metrics for both the current chatbot system, and GPT-4.

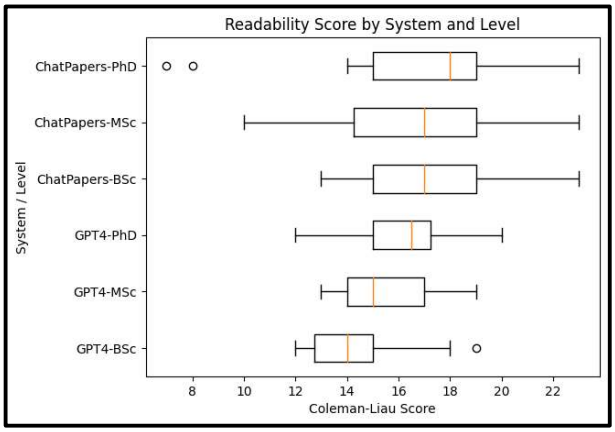

Fig. 6. Flesch-Kincaid boxplots.

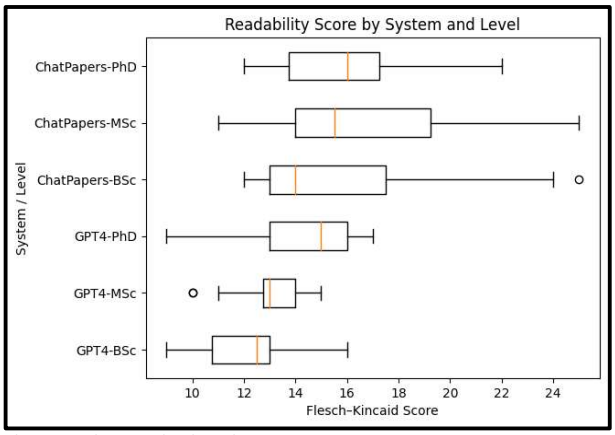

Fig. 7. Coleman-Liau boxplots.

B. RAGAS

Table 3 shows the RAGAS scores and their individual components for the ChatPapers chatbot, calculated from the contexts and outputs from the sample questions. We can see this as a bar chart in figures. The RAGAS results seem to trend upwards as the sub-set of sample questions becomes more difficult.

TABLE II. CHATPAPERS RAGAS SCORES

Question Level	RAGAS	Answer	Context	Faithfulness
ChatPapers				
BSc	0.081	0.925	0.029	0.724
MSc	0.088	0.871	0.031	0.875
PhD	0.104	0.931	0.038	0.773
ChatPapers + Prompt				
BSc	0.185	0.952	0.080	0.369
MSc	0.789	0.961	0.921	0.578
PhD	0.894	0.908	0.972	0.817

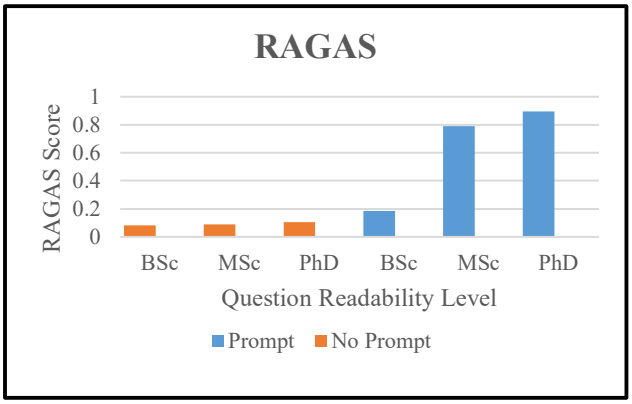

Fig. 8. RAGAS scores for each readability level.

V. DISCUSSION

A. Findings

This study has used both long standing NLP techniques and a cutting edge LLM base framework for the evaluation of a retrieval augmented chatbot. The creation of the chatbot was assisted by those tools as evaluations metrics by which optimal prompts and system parameters could be searched for, acting something like a loss function.

B. Limitations

The literature review in this study was not exhaustive. It is impossible to review all of the relevant literature on such topics as document embeddings, LLM based chatbots, or chatbots in academia as many papers have been published as blogs and not in refereed publications. That said, we hope that the ChatPapers tool provides some assistance in this area in the future.

LLM self-evaluation is an emerging field of study with limited existing knowledge and research to build on top of. The RAGAS framework does a good job of evaluating outputs, but it is yet to undergo a rigorous review in the literature to understand how LLM model biases might affect the scores.

Funding and computational resources have been another constraint in the development of the ChatPapers chatbot. The documents database is 260+GB with a pricing model that scales linearly with total GBs stored. We have at this point only included papers tagged as computer science on arXiv which is about 10% of the papers hosted there. We can

however expect database costs alone to increase 10x just by including all of the papers on arXiv, which is a comparatively small database compared to some others. It should also be noted that the papers on arXiv are pre-print, meaning that they have not necessarily been peer reviewed. Therefore, the data retrieved and presented by the chatbot cannot be relied upon to be 100% correct.

Finally, the embedding model used by the system (all-MiniLM-L6-V2) is powerful for its size. But given more compute resources, we could use a more powerful embedding model that would better capture the semantic meaning of the text items it embeds (both chunk embeddings and queries). A more powerful model could be used like Meta's Llama model, or the OpenAI Ada model. These models output higher dimensional embeddings which may help capture more of the context / semantics in a text chunk.

C. Future Work

The RAGAS framework used here consists of answer relevancy, context relevancy, and faithfulness metrics, though 'context recall' and 'aspect critiques' may also be implemented. These require a ground truth column in the evaluation dataset however - and the creation of such is a labor-intensive exercise. Gathering these ground truths for each of the sample questions could be implemented in the future to bolster the RAGAS score. Also, the RAGAS framework can at this time only be used to evaluate question-contexts-outputs, with no regard to chat history. As it is an open-source project, it may be developed further to encompass metrics that assess the relevance of an answer to a question given the chat history context.

The ChatPapers system has many moving parts and parameters that can be tweaked, tuned, and developed further. It is possible for the system to become orders of magnitude more complicated, with concepts such as child agents, parent child document splitting, and multi-vector retrieval. Child agents can be implemented by instantiating an agent within a tool that is called by a parent agent. Parent-child document splitting is the practice of further dividing the documents and making embeddings of smaller chunks of text that are linked to their parent in the metadata. We have implemented our own alternative to this with the getPriorContext and getSubsequentContext agent tools. And multi-vector retrieval is the practice using either more concise information, such as the paper's title or summary, to make embeddings, or using a simpler method such as taking the average of all vectors related to all text chunks of a document.

D. Application / Significance

It is our hope that the ChatPapers chatbot may be used by scholars to develop their understanding of existing research via a conversational modality. LLMs provide an excellent interface for developing a student's understanding of a topic, but by using in-context learning, we hope to ground the LLMs outputs in real knowledge. We hope that the methods and results discussed herein may make an impact and assist in the development of similar chatbots, either in the domain of computer science research or other knowledge domains using document retrieval augmented generation.

VI. CONCLUSION

In conclusion, we have developed and assessed a ReAct agent based chatbot that uses some 7 tools to interact with a knowledge base of computer science related research papers. The system uses emerging and cutting-edge technologies and frameworks to achieve this and has a user interface that makes it accessible to via a web browser. It is our belief that this type of context augmented system will become increasingly popular in the near future, and we anticipate that these applications may help in the dissemination and the accessibility of scientific knowledge.

VII. REFERENCES

[1] X. Kong, M. Mao, W. Wang, J. Liu and B. Xu, "VOPRec: Vector Representation Learning of Papers with Text Information and Structural Identity for Recommendation," *IEEE Transactions on Emerging Topics in Computing,* vol. 9, no. doi: 10.1109/TETC.2018.2830698, pp. 226-237, 2021.

[2] Y. Wei, Y. He and C. Yang, "JETAM: A Joint Embedding Method for Research Paper Mutil-Label Classification," *2022 International Conference on Image Processing, Computer Vision and Machine Learning (ICICML),* no. doi: 10.1109/ICICML57342.2022.10009721, pp. 390-394, 2022.

[3] A. Kanakia, Z. Shen, D. Eide and K. Wang, "A Scalable Hybrid Research Paper Recommender System for Microsoft Academic," *Association for Computing Machinery,* no. 10.1145/3308558.3313700, p. 2893–2899, 2019.

[4] R. Singh, G. Gaonkar, V. Bandre, N. Sarang and S. Deshpande, "Scientific Paper Recommendation System," *2023 IEEE 8th International Conference for Convergence in Technology (I2CT),* no. doi: 10.1109/I2CT57861.2023.10126196, pp. 1-4, 2023.

[5] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W. Yih, T. Rocktäschel and S. Riedel, "Retrieval-augmented generation for knowledge-intensive nlp tasks," no. 33, pp. 9459-9474.

[6] OpenAI, "GPT-4 Technical Report," 27 3 2023. [Online]. Available: https://arxiv.org/abs/2303.08774. [Accessed 5 9 2023].

[7] M. Dean, "Evaluation Questions," [Online]. Available: https://github.com/dean-boi/com748_eval_questions/.

[8] RAGAS, "ragas (note: see relese 0.0.12 for version used here)," [Online]. Available: https://github.com/explodinggradients/ragas.

[9] pgVector, "pgVector," [Online]. Available: https://github.com/pgvector/pgvector.

[10] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan and Y. Cao, "ReAct: Synergizing Reasoning and Acting in Language Models," 10 3 2023. [Online]. Available: https://arxiv.org/abs/2210.03629. [Accessed 8 9 2023].

[11] L. Wang, W. Xu, Y. Lan, Z. Hu, Y. Lan, R. K.-W. Lee and E.-P. Lim, "Plan-and-Solve Prompting: Improving Zero-Shot Chain-of-Thought Reasoning by Large Language Models," 26 5 2023. [Online]. Available: https://arxiv.org/abs/2305.04091. [Accessed 8 9 2023].

[12] O. Press, M. Zhang, S. Min, L. Schmidt, N. A. Smith and M. Lewis, "MEASURING AND NARROWING," 7 10 2022. [Online]. Available: https://ofir.io/self-ask.pdf. [Accessed 8 9 2023].