Zorawar Jaiswal

Prof. Virginia Mushkatblat

COMP 582

Oct 30, 2024

# **Automatic Quiz Generation using RAG and LLMs**

**Abstract**

The advent of LLMs has led to the automation of various processes that directly or indirectly impact the human effort. Automatic Quiz Generation is one of those applications that can help reduce human effort and improve the quality of education. A lot of research has been done to improve the quality of quiz generation and to help cover as many sources of information as possible. This paper presents a novel architecture which utilizes the Retrieval Augmented Generation (RAG) pipeline and incorporates an LLM to generate quizzes either from one-word to one-liner inputs or from a plethora of user uploaded documents. The paper also covers the latest trends in the industry such as the shift toward Vector Database from traditional systems and the involvement of LlamaIndex to ingest data.

**Introduction**

The world is moving at a pace faster than ever with the advent of Artificial Intelligence and text generation. A plethora of applications have been developed since the inception of Chat GPT where most of the applications integrate some form of Large Language Model (LLM) and a pipeline to ingest the user data.

A few examples of the LLMs available across the tech industry today are Chat GPT (OpenAI o1), Gemini (Google), Claude (Antropic) and Llama (Meta). At the center of modern day LLMs lies the transformer architecture in deep learning (Wang, 2022) that is responsible for the widespread creation of such models.

A major area that has scope of improvement with the application of these LLMs is automatic quiz generation. Not even 15 years ago thinking about a system that could generate quizzes on almost any topic or user input would have seemed next to impossible. Today’s technology targets automatic quiz generation as a supplementary tool to enhancing education and learning.

The main application of these tools is to help fasten the process of academic learning by allowing users to either create quizzes on topics of their interest such as a one-line definition as seen in products such as the AI Quiz Creator (aiquizcreator.com) or implementing a user ingestion pipeline to accept PDF files, text box inputs (250-350 characters minimum) or image files to generate quizzes like Revisely (revisely.com), Jotform (jotform.com), Fillout (fillout.com), or QuizGecko (quizgecko.com).

**Related Works**

*Industry*

Some notable trends include QuizGecko that can create flashcards for the AI generated quiz based on the input material to ensure better learning with personalized reports and scores and incorporates Generative AI to create images for flashcards (QuizGecko, 2024).

SafetyCulture goes a step further and allows users to create courses from the scratch using Generative AI. All that the user has to do is define the scope of the course and SafetyCulture would create course content based on preset templates and automatically generate quizzes based on the content (SafetyCulture, 2024).

AI Quiz Generator, on the other hand, stands out as one of the few solutions that without any prompt engineering yields answers on one sentence themes. It utilizes ChatGPT under the hood and is mainly created to accelerate classroom learning by allowing teachers to create quizzes in a short time. The system underneath has classic prompt engineering that amalgams and passes the user query with their proprietary prompt to create quizzes (QuizAlize, 2024).

Fillout another quiz generator requires users to input at least 350 characters of text (5000 maximum) and they generate quizzes utilizing Chat GPT. Once again, they have their own prompt engineering to create quizzes (Fillout, 2024).

Jotform goes a step further and allows majorly files to be uploaded. They limit it to 5 MB and can process on PDF files, powerpoint presentations, word documents and text files and allow the users to choose between MCQs, Single choice questions and Yes or No questions. Jotform is also powered by Chat GPT at its core. Jotform yields better results for single line prompts to generate quizzes in comparison to QuizAlize as QuizAlize was only able to generate a single question for the prompt ‘Rivers that start with the letter “N”’ (Jotform, 2024).

Revisely happens to be a mostly paid service from the UK. It goes a step beyond the other products and allows users to process text files, PDF documents, Powerpoint presentations, images, and even videos. They also allow multi-language support meaning the quiz could be created in languages other than English and as a premium feature, they are also able to create quizzes from hand-written documents. They provide no information about how they achieve AI capabilities but are also able to create flashcards in addition to quizzes (Revisely, 2024).

Finally, precise prompt engineering also yields exceptional results when using ChatGPT (4.0 mini) (OpenAI, 2022) by OpenAI.

*Academia*

(Lin et al., 2015) in their research presented Sherlock, that utilized TF-IDF vectors and implemented semantic similarity using K Means Clustering and segregated questions into Hard, Medium and Easy levels. This is followed by a template-based QA generator where [mask] question labels are replaced with the target theme. They used the BBC Wildlife Dataset for their QA generation.

(Singh & Sharma, 2018), worked on the SQuAD dataset. They elaborate on using an Encoder-Decoder mechanism where the questions are first encoded into hidden state and self-attention is applied, and later on decoded to optimize the probability of the correct answer. The process is highly context-unaware but was faster than its predecessors in terms of training time.

(Vachev et al., 2022) in their work presented Leaf, an MCQ generator framework that was built upon the transformer architecture. They utilized the SQuAD1.1 dataset (100K questions and answers) and the [MASK] token to avoid overfitting. They were able to achieve 41% accuracy and the F1 score came out to 53.

(Mao et al., 2022), in their work describe a Question Recommendation System for quizzes. They argue that the current industry trends only account for past data. In their system, they propose that the user should provide feedback either numeric or textual (and perform sentiment analysis) after each question and dynamic plus static recommendations should be combined to provide users the questions they would want to prioritize rather than AI just generating questions without any vetting.

(Thüs et al., 2024) in their work implemented a Retrieval Augmented Generation (RAG) system called OwlMentor that could ingest the atlas of data and later on, generate a quiz once the user provides their query. For their purpose, they vectorized the data using OpenAI’s embedding model (OpenAI, 2022) and then stored this data into Annoy Vector Database. Once the user queried with their interest (topic to quiz upon) the system would search the Annoy Database for similar keywords and generate questions on the same using Chat GPT’s 3.5v. The purpose of their system was to identify gaps in learning and to provide complete explanations to the user.

# **Preliminaries**

Computers do not understand text. At the core level, it is all binary operations. One level above that are decimal and floating point numbers. To make the computer understand our input in natural language, we need to convert the text into numbers.

The conversion of text into numbers can be achieved using encoding (a famous example is one-hot encoding), however, this works only for fixes length of inputs. Say we have just five words in total, encoding would work out to be a great practice here.

However, if we have something bigger like 300 GB of text, encoding would not be very feasible here. For such scenarios, vectorization is the way forward. Vectorization of text converts the text into vectors (floating numbers or arrays). A few common techniques for vectorization of text are (i) Count Vectorization, (ii) N Gram Vectorization, (iii) TF-IDF Vectorization. The latest industry trends elaborate on using transformer-based vectorizers and Chat GPT’s own vectorizer for ease.

Vectorization is always done after the data is cleaned. Data preprocessing steps may include but are not limited to (i) removal of punctuation, (ii) correction of spellings, (iii) removal of stop words (commonly occurring words), (iv) tokenization (splitting of sentences or group of words into individual words), (v) lemmatization or stemming (converting words to the root form).

Once vectors are created, they need to be stored somewhere. The common practice is to store them in pickle files as a binary dump as that allows faster access of the data. At this stage, all the corpus data is in binary form from the initial textual representations.

This data can then be passed in to fine-tune some LLM and train it to reason our queries based on the vectors we created. This is followed by prompt engineering where we create a prompt that when combined with the user query, would lead to optimum and desired results from the LLM.

# **Problem Statement**

Automatic Quiz generation is the need of the hour. Not only can it accelerate learning for students by providing quizzes on the smallest possible texts, it is also a boon for educators. Educators have plenty of important tasks throughout the day and may end up missing out on covering some aspect of the text when designing a quiz or an exam.

The automatic quiz generation can easily target this situation and provide a fast outcome where a quiz is automatically generated based on a prompt, or text input with the type of questions needed. This quiz could then be evaluated by the educator for difficulty, vastness and length and could be assigned to the students.

Currently the solutions available for this problem are mostly closed-sourced. An open-source solution would be a great addition and could help educate people better for a cheaper price.

# **Methodology**

The main process for automatic quiz creation can be divided into two main parts (i) Ingestion of data, (ii) Query Processing. The main components of the system are an LLM, an ingestion pipeline for data, a vectorization module and storage for those vectors. All of this can be tied up to a User Interface using Streamlit (Streamlit, 2024).

The query part can further be divided into two components. One would be to ascertain if the user wants a quiz on any specific topic, this would be a one-word to one-liner topic such as ‘rivers that start with “N”’. Quizzes on such topics would be build using the base knowledge of the LLM without any other user provided data.

The second type of user input can be structured PDF files, text files, presentations, etc. These would be fed into the data ingestion pipeline and then converted to vectors, finally the LLM would retrieve this data and use it to form questions. This part would be forming a Retrieval Augmented Generation (RAG).

For storing the vectorized embeddings, using a vector database would be the best bet. Milvus (Open Source) (Milvus, 2024) and Pinecone (free and paid tiers) (Pinecone, 2024), are the best options to store the embeddings other than storing them in a text file. The advantage of using a vector database is that all the querying within embeddings is similarity based rather than term-matching (as observed in elasticsearch) and thus, would result in a faster retrieval for the LLM.

The data ingestion pipeline can be created using LlamaIndex, which is open source in nature and has various tools to directly read multiple types of input files such as PDF files, images, word documents, etc. LlamaIndex can also directly feed into the LLM and all the prompt engineering would be done here (LlamaIndex, 2024).

The main part would be to integrate an LLM. The best options for LLM are Chat GPT (paid access only) and Ollama (Open Source by Meta). Ollama is a great alternative to Chat GPT but would require high resource computing to run for even a small model (Meta, 2024).

The architecture of the system would be a service invoke system because LlamaIndex is designed to operate in a call and return fashion with the LLMs. Figure 1. Below shows the architecture of the system.

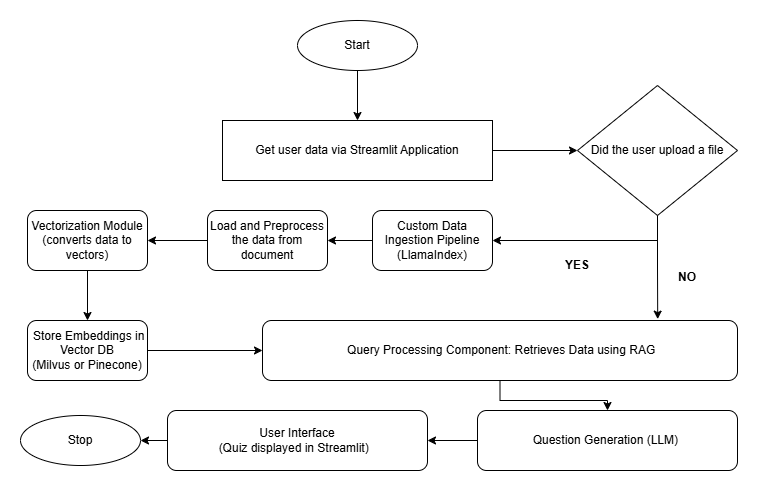


Fig 1. Workflow of the Application

# **Evaluation & Discussion**

The model presented in this paper utilizes Vector Database to store the embeddings. None of the previous works mentioned used Vector Database to store the embeddings. All of them relied on using the pickle or binary dump other than (Thüs et al., 2024). But they implemented their system only for text inputs and not across different file storage formats.

The famous algorithms such as TF-IDF, Named entity Recognition (NER) are techniques of the past as the Transformer Embeddings and Self-Attention when combined with Similarity based searching have replaced them for faster performance and are more scalable in nature.

(Thüs et al., 2024) also do not mention anything about how scalable their system was, but the architecture presented in this paper would ensure scalability and language agnosticism since everything is service-based. Additionally, this paper explores the usage of open-source LLMs instead of relying on closed tools such as Chat GPT.

The major limitation of this paper is to not be able to incorporate quiz generation on handwritten notes as was incorporated in Revisely.

# **Conclusion**

The paper presented a novel architecture to create a service-based RAG application that can take input from the user and automatically generate a quiz on the same. The paper has a plethora of applications since the quiz creation is not limited to a single sentence of user input as users can simply upload files and generate quizzes.

Common applications for this system would be to (i) Generate quizzes for students, (ii) Generate FAQ sections from guidelines, (iii) to create courses, (iv) or be a cool trivia game amongst children, etc. The world of possibilities is never limited and one can always innovate and find more applications.

For future work, a good amount of focus can be to incorporate quiz creation on handwritten notes with a good accuracy. One can also focus on working with unstructured PDF data such as invoices, forms, etc. Or even work on an assessment system that identifies the gaps in learning and provides the user with easier explanations to clear the concepts.

# **Acknowledgements**

I would like to acknowledge Prof. Virginia Mushkatblat for being a source of inspiration and helping out by pointing me to the right direction to research about the latest trends in the industry.

# **References**

Fillout. (2024). *Fillout | AI Quiz Generator*. AI Quiz Maker. https://www.fillout.com/ai-quiz-maker

Jotform. (2024). *Jotform Ai Quiz Generator: Free Ai Quiz Generator*. Jotform | Form Builder. https://www.jotform.com/ai/quiz-generator/

Lin, C., Liu, D., Pang, W., & Wang, Z. (2015). Sherlock: A semi-automatic framework for quiz generation using a hybrid semantic similarity measure. *Cognitive Computation*, *7*(6), 667–679. https://doi.org/10.1007/s12559-015-9347-7

LlamaIndex. (2024). *LlamaIndex, Data Framework for LLM applications*. https://www.llamaindex.ai/

Mao, K., Dong, Q., Wang, Y., & Honga, D. (2022). An exploratory approach to intelligent quiz question recommendation. *Procedia Computer Science*, *207*, 4065–4074. https://doi.org/10.1016/j.procs.2022.09.469

Meta. (21AD). *IBM granite 3.0 models · Ollama Blog*. Ollama. https://ollama.com/blog/ibm-granite

Milvus. (2024). *What is Milvus?*. Milvus, a highly performant distributed vector database for AI apps. https://milvus.io/intro

OpenAI. (2022). *Introducing chatgpt*. https://openai.com/index/chatgpt

Pinecone. (2024). *The vector database to build knowledgeable AI*. https://www.pinecone.io/

QuizAlize. (2024). *Create a quiz for Google forms in seconds*. AI Quiz Creator. https://www.aiquizcreator.com/

QuizGecko. (2024). *Create a quiz online for free using AI*. QuizGecko | Features. https://quizgecko.com/features

Revisely. (2024). *Revisely | Ai Quiz Generator*. https://www.revisely.com/quiz-generator

SafetyCulture. (2024). *Create with ai: SC training: A mobile LMS*. Create with AI | SC Training: A mobile LMS. https://training.safetyculture.com/ai-create/

Singh, J., & Sharma, Y. (2018). Encoder-decoder architectures for generating questions. *Procedia Computer Science*, *132*, 1041–1048. https://doi.org/10.1016/j.procs.2018.05.019

Streamlit. (2024). *Streamlit • A faster way to build and share data apps*. Streamlit.io. https://streamlit.io/

Thüs, D., Malone, S., & Brünken, R. (2024). Exploring generative AI in Higher Education: A RAG system to enhance student engagement with scientific literature. *Frontiers in Psychology*, *15*. https://doi.org/10.3389/fpsyg.2024.1474892

Vachev, K., Hardalov, M., Karadzhov, G., Georgiev, G., Koychev, I., & Nakov, P. (2022). Leaf: Multiple-choice question generation. *Lecture Notes in Computer Science*, *13186*, 321–328. https://doi.org/10.1007/978-3-030-99739-7\_41

Wang, B. (2022). *From Transformer to LLM: Architecture, training and usage*. From Transformer to LLM: Architecture, Training and Usage. https://scholar.harvard.edu/binxuw/classes/machine-learning-scratch/materials/transformers