Contextual Conversational Agent using Embedding Similarity Information Retrieval

Ameya Panse
apanse@ucsd.edu

Aakarsh Gupta
aag010@ucsd.edu

Siddharth Shukla
sishukla@ucsd.edu

1 Introduction

When a human engages in a conversation with an expert in a particular field to gain knowledge about the concerned domain, any subsequent question that the human poses is framed on the basis of previous conversation history - we ask a question from an accompalished person and depending on their answer, we follow up with another question. This incremental aspect makes human conversations succinct. Nonetheless, answering questions that require reasoning over a long sequence, such as over long and multiple documents, is a challenging task even for the humans. It has been seen that Large Language Models (LLMs) have shown great performances for question answering tasks from different domains.

In this project, we have tackled the problem of Contextual Conversational Question Answering by modelling it as an information retrieval problem at its core and building a conversational agent wrapper. Given one or more document(s) (for example research papers or lecture notes) as a source of information for the question-answer conversation, our conversational agent takes as input these documents (as context), a query (question provided by the user) and provides an answer generated by the LLM for the query, taking inferences from the given context (as documents). Our agent is also capable of providing the references for the answers it generates. This way, we have also tackled the problem of hallucination (Ji et al., 2023), faced by most of the LLMs. When the user decides to exit the conversation, the agent provides a final summary of the whole conversation to the user in the form of flash cards.

For this project, we have chosen the dataset as documents from the lecture notes from - CS231n (Deep Learning for Computer Vision), CS234 (Reinforcement Learning), CS324 (Large Language Models). More details and information about the dataset are provided in Section 4 below. Our conversational agent is also capable of providing cross content answers for the given queries. For example, we can ask pass in documents of two or more than two courses to the agent and get answers to relevant questions from the agent. The agent will find inferences from all the data provided, that is from all the courses whose documents were given. Another practical application of our project would be finding answers to questions on the basis of multiple research papers from a specific domain. This would enable a quicker understanding of newly published papers, where the reader uses our Conversational Agent to gain insight. The user could also use our agent for cross referencing and comparison of techniques discussed in different articles.

2 Proposed vs Accomplished

The below points showcase our accomplished tasks:

- ~~Break up the documents in a organized manner~~ - Done (used MarkDown and PDF splitter)
- ~~Generate relevant Embeddings of the documents and treat them as indices~~ - Done (used HuggingFaceHub api for embedding)
- ~~Generate the embedding of the query + conversation history~~ - Done
- ~~Find relevant Document Embeddings and set them as the context for the answer~~ - Done (used Chroma DB)
- ~~Generate an appropriate prompt for the LLM task~~ - Done - (by appeding the retrieved documents)

- ~~Query a LLM to produce a relevant answer~~ - Done (used openAI GPT 3.5 model)
- ~~Generate references to the documents to prevent hallucinations~~ - Done (used similarity score for retrieval)
- ~~Generate a summary as pointers for reader's future use.~~ - Done (used customized prompt for summarization)

3 Related work

Question Answering (QA) tasks generally involve accessing different sources of information to find an appropriate answer for the input query/question. The initial QA systems, which date back to the mid 19th century were rule-based and developed on small datasets, because of which they did not perform well on a lot of practical applications in the 21st century. With the rising use of Deep Learning techniques are increasing sizes of the datasets, there have been numerous works on open-domain Question Answering tasks and context based text retrieval using language models.

Dense retrieval for open-domain QA tasks has been tackled by Das et al. (2019), who proposed a new framework for open-domain question answering in which the retriever and the reader iteratively interact with each other. Their framework was agnostic to the architecture of the machine reading model, only requiring access to the token-level hidden representations of the reader. Karpukhin et al. (2020) worked on Dense Passage Retrieval for Open-Domain Question Answering to improve on the retrieval component in open-domain question answering. The Dense Passage Retriever uses an encoder which maps all the passages to a lower dimensional vector and these vectors are then used for retrieval. At run-time, the retriever applies a separate encoder to map the input question to the same dimensional vector and then find the top-k passages whose vectors (encodings) are closest to that of the input query.

The task of Conversational Question Answering (CQA) differs from general QA tasks in many ways. In traditional QA tasks, questions are independent of each other and are based on the given passage whereas in CQA, the questions are related to each other which poses entirely different challenges. Some of these challenges are - a). The model needs to encode not just the current question and context paragraphs but also the previous questions and answers that the model would have provided to the user. b). Different questions may be of different types. Some questions may require information about a topic previously discussed, some might require a detailed answer etc. Most of the best performing traditional QA systems fail to address these challenges directly. The availability of recently released dataset Reddy et al. (2019) has allowed the community to make significant advancements in the field of text-based conversational QA. Some efforts (Li et al., 2022a) have recently been made to expand CQA to open-domain setting, where the relevant passages are retrieved rather than sent directly. However, these existing methods only involved one knowledge source and the important visual cues were overlooked. Li et al. (2022b) introduces multimodal conversational question answering (MMCoQA).

Notably, the dataset reveals that students tend to begin dialogues by asking questions about the beginning of the evidence text before progressing towards the end. This finding emphasizes the importance of incorporating the dialog context when developing models for QuAC(Choi et al., 2018) . Overall, the QuAC dataset (Choi et al., 2018) offers a valuable resource for studying and advancing the understanding of contextual question answering within conversational settings.

The above mentioned works provide valuable insights into the challenges and advancements in contextual question answering. Through our project, we have built a contextual conversational agent **(CCA)** that can engage in a conversation with the user and answer the user's queries based on not just the knowledge base that it has access to, but also the context that the user would provide. We have taken some ideas from the currently existing CQA systems like MMCoQA, and also from deep learning based retrieval methods that have performed well on various datasets to build our project.

4 Dataset

For our contextualized conversational agent, we required a collection of informative text that could serve as additional context to provide relevant information during conversations. Our approach aimed to retrieve relevant documents using a vector database (DB) and group them as additional context with the query to enhance the language

model's response. This strategy helps reduce hallucination and allows the model to provide references for the queried information, thereby building confidence in the generated responses.

To accomplish this, we explored datasets containing informative text such as PubMed and arXiv research papers. However, due to the massive scale of these document collections, we opted to narrow down our dataset to a more manageable size. Consequently, we turned our attention to Stanford University, where lecture notes for various courses are readily available. These lecture notes offer a wealth of informative content and are ideal for our purposes.

The statistics of the dataset used for the project is mentioned in the Table 2. By utilizing the lecture notes from Stanford University as our dataset, we empower our conversational agent to provide informative and contextually appropriate answers to user queries. The diversity of the lecture notes allows our agent to draw insights from various domains, including natural language processing, computer vision, and reinforcement learning. Through the application of information retrieval techniques and the integration of relevant content from the lecture notes, our conversational agent aims to deliver accurate and concise responses, culminating in a flashcard-style summary of the conversation by leveraging large language model's capabilities.

This approach not only leverages the rich educational resources provided by Stanford University but also demonstrates the practicality and effectiveness of using real-world lecture notes as a dataset for providing a contextualized answer from a conversational agent. By incorporating the knowledge and insights from these lecture notes, we strive to create an intelligent and reliable agent capable of engaging in meaningful question-answer conversations and providing users with valuable information and summaries.

4.1 Data preprocessing

In the data pre-processing stage of our project, we undertake the task of reading and processing lecture notes and materials in both markdown and PDF formats. To facilitate this process, we utilize the Hugging Face Hub embedder, which allows us to convert the textual content into meaningful numerical representations that can be processed by the lange language model.

The Hugging Face embedding model leverages the pre-trained GPT-2 model to encode documents into dense vector representations. These vectors capture the semantic meaning and contextual information of the text, enabling downstream tasks which in our case is information retrieval.

To vectorize a document using the Hugging Face embedding model, we input the text of the document into the model and obtain the corresponding vector representation as the output. The resulting vector representation encodes the document's content in a high-dimensional vector space, where similar documents are likely to have similar vector representations. Hugging face embedding model, leverages the knowledge learned from vast amounts of pre-training data,so that the model can generate meaningful and informative vector representations for a wide range of texts.

For markdown text, we employ the Markdown Text Splitter provided by Langchain. This splitter enables us to break down the text into smaller, more manageable segments, such as paragraphs or sentences. By segmenting the markdown text, we gain greater granularity and can analyze and process the content at a more fine-grained level. This becomes particularly useful when dealing with long documents or when we need to focus on specific sections within the notes.

On the other hand, for PDF text, we utilize the PDF Text Splitter also provided by Langchain. This splitter enables us to extract the text from the PDF files and split it into relevant segments. PDFs contain information in the form of page, corresponding to each of the lecture slides. It allows us to work with the content in a more organized manner.

Once we have split the markdown and PDF text into smaller segments, we then proceed to embed the text using the Hugging Face Hub embedder. The embedder converts the text segments into numerical representations, often in the form of vectors, that capture the semantic meaning and contextual information of the text. These embeddings serve as input for our models, in the form of additional data, enabling them to understand and process the lecture notes effectively.

It is important to note that the use of the Markdown Text Splitter and PDF Text Splitter, in conjunction with the Hugging Face Hub embedder, is the critical part of the data pre-processing pipeline. The ability to segment and embed the text allows

Subject Code	Number of Documents	Average Characters	Average Tokens	Notes Format
CS 324	15	17543	2209	Markdown
CS 231	28	21372	3201	Markdown
CS 234	15	22722	3365	PDF

Table 1: Dataset Statistics

us to handle large volumes of lecture notes and materials efficiently, enabling our models to generate informed and contextually relevant responses during conversational interactions.

5 Baselines

Our project of building a contextual conversational agent using LLMs falls into an area which hasn't been explored much. We could not find a suitable dataset which contains queries, context and contextualized answers to the queries in a single dataset consisting of high span. Since the outputs of our agent are based on the context provided to the agent, in order to construct the ground truth for the agent, the annotator will have to go through all the documents that would be provided as context to the agent. In our case, it would mean that the annotator would have to complete all the three courses CS231n, CS234 and CS324 in order to formulate ground truths for the queries. This makes it quite difficult to create the ground truths for the queries that the user asks from the agent. Moreover, there is no training of a network involved in our project and we're directly using APIs of trained language models because of which we don't have training or validation loss values either. We have included details about our evaluation routine in the Error Analysis Section.

6 Our Approach

6.1 Conceptual Approach

We model the Conversational Problem from the lens of Information Retrieval. We split all documents in our dataset and embed the parts into a latent space using a LLM. These embeddings are inde with relevant metadata. At every step of the conversation, we embed the user's query using the same LLM and retrieve relevant documents from the preindexed data. The query, relevant documents and past conversation history is then clubbed together as an input to the Chat Model.

6.2 Implementation Details

We have a **working implementation** of our approach. We make use of the following already **available models**. We use the pretrained GPT-2 model as our Embedder. The model is available through the Hugging Face Hub API. As out Chat LLM, we make use of ChatGPT-3.5-turbo which is available through the OpenAI API. Our code is **written from scratch** by ourselves.

6.3 Implementations

1. **Embedding Generator** In the file embed.py, we have implemented the class Embedder. This class takes in raw files in pdf and md format. Then it proceeds to semantically split the files and then embeds them in a latent space. To generate the text to latent space embeddings, we use the GPT-2 model avaiable in the Hugging Face Hub. Ideally, we want to use the latest GPT-3.5 model. However, since the information retrival runs seperately from the conversational agent, a different and *free* model serves our purpose well.

2. **Conversation Agent** In the file conversation.py, we have implemented the Conversation class. Here we have a 'LLMChain'. It uses our embedded documents as a 'retriever' to pull documents that are relevant to the user query. It rephrases the users question by using a prompt template having 'Chat History' and 'Follow Up Input' fields. Finally, the relevant documents and rephrased question are fed into a 'Contextual Question' prompt which specifies to the LLM to find answers from the generated documents.

3. **Reference Citation Agent** In the conversation.py file, we have a print_citations method. Using the retireved documents, we query the metadata to find the course name and specific lecture that was fed in as a context to the Chat LLM.

4. **Conversation Summarizer** Finally, in the

conversation.py file we have a summary prompt. This prompt tells the LLM to generate a short summary and provides an example of a summarised conversation. The prompt template is filled in with the conversation history and then passed on to the LLM for a summary.

6.4 Compute and Runtime

We do not train any of the model, rather we use the already trained model through their APIs. Hence, out project did not require any GPU/TPU compute. The heaviest part of our project was the document embeddings which took around half an hour to run on out local machines. Hence this was done as a pre-processing and the embeddings were stored.

Finally, during runtime, our class interacts with LLMs through exposed APIs and hence need internet connectivity to run.

6.5 Results

We have successfully demonstrated the capabilities of LLMs as a Contextualized Conversations using Similarity Based Information Retrieval. Our agent display the following capabilities:

1. Search Space of over 60 lectures spanning three different courses. This is similar to a gradutate student's workload.

2. Cross References : The ability to answer question relevant to different courses.

3. Contextual Question Answers and Ability to retain and utilize previous chat history as conversational memory.

4. Cites References for each question answered. The enables the user to fact check and prevents hallucinations by the Chat LLM.

5. Generates a Summary at the end of each conversation for the user's convenience in form of flash cards.

6.6 Additional Techniques and Hyperparameters

1. **Hot Wire** Our Conversational Buffer Memory and a corresponding Prompt, it is empty at the initial stage. This caused the model to throw 'I do not know' answers. To fix this, we hotwire the Agent by feeding in relevant question before hand.

2. **Citations** When we split the documents in out data set, we make sure to add in relevant metadata such as Course Name and Lecture Title which is useful during the citation phase.

3. **Hyperparameters** The hyperparameters in our agent (CCA) are the number of retrieved documents as reference, split size of the documents. If the split size becomes too large, the size of the token exceeds the limit which throws an error. If it becomes too small, the context is not enough for the agent to fetch accurate information. The number of retrieved documents for reference we used is 2-4. We face a similar problem with this, in the sense that if it is too large, it exceeds the limit and if it is too small, the agent doesn't have enough context to retrieve the answer from.

7 Error analysis

For evaluating our agent, we conducted some experiments. In each of the experiment, we decided to ask human evaluators to rate each of our agent's contextualized answers out of 5. We also decided to ask the scorers to rate the references provided by the agent out of 5, and to rate the summary provided by the agent at the end of the conversation. It wasn't quite straightforward since we wanted to have our scorers as people who would have completed at least one of the three courses we chose. For each of the experiment, we created a google form and floated the forms to relevant evaluators. For reference, the google forms are also available in our GitHub repository.

7.1 Evaluation on Individual Courses

In order to evaluate our agent, we decided to adopt an approach based on human evaluation for which we first performed 3 experiments, one for each course :-

- CS231n : Lecture notes of CS231n were provided as context, and Computer Vision related questions were asked

- CS234 : Lecture notes of CS234 were provided as context, and questions related to Reinforcement Learning were asked

- CS324 : Lecture notes of CS324 were provided as context, and questions related to Large Language Models were asked

7.2 Evaluation on Multiple Courses

In order to test the effectiveness of our agent when provided with context from multiple sources, we performed another experiment where we passed lectures of all the three courses together and queried the agent with two types of questions :-

- some questions were specific to one of the three courses (similar to ones used in previous three experiments where we passed the context as lecture notes of only course), to which we expect pretty much the same answer (atleast semantically) as we got during the first experiment. We observed that we got very similar answers to what we got previously. We also observed that the agent could fetch out most of the references from the same course to which the query was related to.

- The next type of queries were those, the answers to which belonged to more than one course. For example - we asked the agent about the models where attention is used to which the agent gave an answer comprising of models used in both the Computer Vision (CV) domain and also in the Natural Language Processing (NLP) domain. The agent mentioned image captioning for CV and machine translation for NLP even though we didn't explicitly mention the keywords CV or NLP in the query. Furthermore, the references that the agent cited to for providing the answer were from both the courses - CS231n and CS324.

This shows that our Contextualized Conversational Agent is adept in providing answers on the basis of context and not on the basis of the knowledge base that the LLM has access of. It is also able to fetch the appropriate references for providing the answers which shows that the model is not hallucinating.

7.3 Failed examples

Out of Context The agent does not work well when the user queries a specific question that the information retrieval is not able to retrieve. This happens when either the question is out of context, due to which there is no relevant document available. The agent also doesn't provide very good context when the query is highly generic and not pertaining specifically to the context provided.

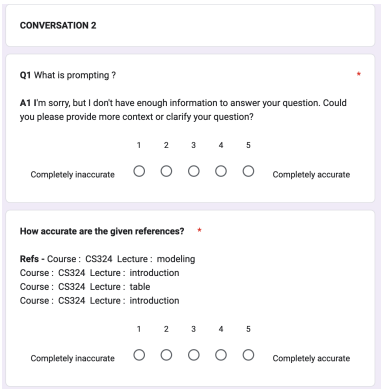

Figure 1: Failed Examples : Asking an Question that is too specific and hence the information retrival does not work well.

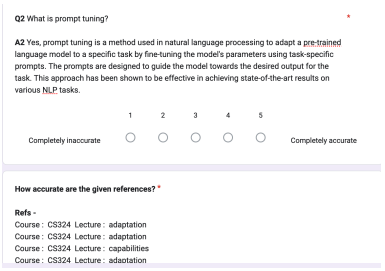

Figure 2: Tweaking Failed Examples

Here 1 the agent says that enough information is not available. However, tweaking the query results in better answers 2

Another case of failure is when the user asks questions that require the model to generalise from a context to a different setting 3

Generalising Context

8 Contributions of group members

- Ameya Panse: Algorithm Deisgn, CodeBase - Embedder, Conversation Agent, Citation Agent and Summary

- Aakarsh Gupta: Citation Agent, Splitting MD documents, Experiment Design, Error Analysis

- Siddharth Shukla: Splitting pdf documents, Evaluation, Error Analysis, Dataset Discovery

9 Conclusion

9.1 Takeaways

- We have provided a working solution to the contextual conversation problem which provides checks and balances for hallucinations.

Dataset	Count	Conversation Score	Citation Score	Summary Score
CS 324	12	4.33	4.5	4.5
CS 231n	11	4.18	4.11	4.5
CS 234	11	4.36	3.87	4.7
Mix	12	3.89	4.33	4.56

Table 2: Dataset Statistics

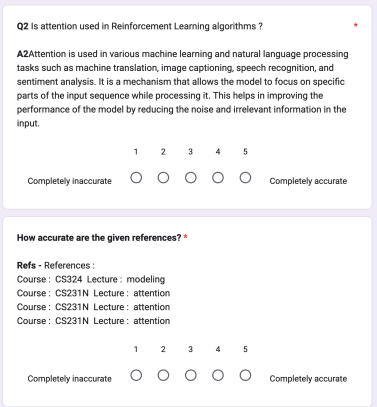

Figure 3: Failed Examples : Asking an Question that is too specific and hence the information retrival does not work well.

- Our solution is a lightweight wrapper and utilizes state of the art LLMs and their APIs.
- We do not need to train, fine-tune or modify any part of the LLM
- This idea has applications in the real world where companies who own private data, such as healthcare applications, do not want to share it with LLMs, but want to utilise their capabilities.

9.2 Difficulties and Surprises

The key to our solution was to strike the balance between the LLM's vast knowledge and the context provided by the Information Retriever. Without the context, the LLM is likely to provide unrelated solutions, where as we wished to utilise the LLM's language model for smooth answers.

The biggest difficulty we faced was choosing the dataset and pre-processing the data into the format that we needed. We had to split the documents in such a way that the splits contained enough information but were not too long. Changing the split size had a huge effect on the contextualised conversation.

The state-of-the-art LLMs are not freely available. Since creating the embeddings was a heaviest task, it would have required a subscription and hence we had to switch to a freely available but less powerful model. This caused a minor degradation in our results.

Finally, since none of us had personally undertaken the courses in our data set, we were not best equipped for evaluating the contextualized answers. Hence we had to go through the references one by one.

9.3 Future Work

We would like to explore the 'Mixed Dataset' setting further. Specifically, we wish to evaluate how well the model generalizes the information from one course to another.

Another area that can be looked into further is the different modalities that can be included in the model. For example, we ignore many images and diagrams provided in the courses that provide further context. One direction is to utilize a pretrained CLIP architecture (Radford et al., 2021) to generate and index visual embeddings.

Finally, we would like to conduct experiments in the various application domains of our project. For example, we can use a research paper as our data set and use our solution for faster and easier understanding through a contextual conversation.

10 AI Tools Disclosure

We have used our project proposal and have written this report on our own. No AI tools were used in this report.

References

Choi, E., He, H., Iyyer, M., Yatskar, M., tau Yih, W., Choi, Y., Liang, P., and Zettlemoyer, L. (2018). Quac : Question answering in context.

Das, R., Dhuliawala, S., Zaheer, M., and McCallum, A. (2019). Multi-step retriever-reader interaction for scalable open-domain question answering.

Ji, Z., Lee, N., Frieske, R., Yu, T., Su, D., Xu, Y., Ishii, E., Bang, Y. J., Madotto, A., and Fung, P. (2023). Survey of hallucination in natural language generation. *ACM Computing Surveys*, 55(12):1–38.

Karpukhin, V., Oğuz, B., Min, S., Lewis, P., Wu, L., Edunov, S., Chen, D., and tau Yih, W. (2020). Dense passage retrieval for open-domain question answering.

Li, Y., Li, W., and Nie, L. (2022a). Dynamic graph reasoning for conversational open-domain question answering. *ACM Trans. Inf. Syst.*, 40(4).

Li, Y., Li, W., and Nie, L. (2022b). MMCoQA: Conversational question answering over text, tables, and images. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4220–4231, Dublin, Ireland. Association for Computational Linguistics.

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G., and Sutskever, I. (2021). Learning transferable visual models from natural language supervision.

Reddy, S., Chen, D., and Manning, C. D. (2019). Coqa: A conversational question answering challenge.