

An LLM-based Chatbot Answering Questions About Deep Learning

Final Report

Sara Pasquino
Valentin Pinon
Jan Philipp Girgott

Massachusetts Institute of Technology
Sloan School of Management
Hands-on Deep Learning (15.093)

March 11th, 2024



Table of Contents

1	Problem Description	3
2	Data.....	4
2.1	Data Collection	4
2.2	Data Preprocessing	4
3	Methodology	5
3.1	Link to Colab Notebook	5
3.2	Description of Methodology	5
4	Discussion of Results	6
4.1	Exemplary Chat History.....	6
4.2	Technical Results.....	8
4.3	Lessons Learned and Future Work.....	9



1 Problem Description

In our Hands-on Deep Learning (HODL) Project, we will develop an LLM-based chatbot which can answer students' questions on the HODL class content to create an automated Q&A environment. Currently, most MIT classes rely on email contact, Canvas forums, or Piazza to answer student questions regarding the course material. This is suboptimal for MIT, students, and instructors/teaching assistants (TAs) for several reasons:

Students do not get real-time answers to their questions: Traditional methods of communication can lead to delays in responses, leaving students waiting for answers. This can hinder their learning process, especially if they are stuck on a concept that prevents them from progressing.

Answering questions is time-consuming for TAs and instructors: The time spent by TAs and instructors on answering repetitive or similar questions could be better utilized in activities that enhance the educational experience. Time could be used to improve teaching material, conduct research, or grade assignments more efficiently. By automating the Q&A process, TAs and instructors can redirect their efforts towards these more impactful activities.

Maintaining a platform like Piazza is costly for MIT whereas many LLMs are open-source: The financial resources allocated to subscription-based platforms for student-instructor interaction could be optimized. Many Large Language Models (LLMs) are available as open-source projects, which means they can be customized and used without the recurring costs associated with proprietary platforms. Adopting an LLM-based solution could therefore result in significant cost savings for the institution while providing a similar or even superior service.

By addressing these issues, the HODL Project aims to enhance the educational experience for both students and faculty, making learning more efficient and resourceful.



2 Data

2.1 Data Collection

For our project, we are using the following text data for our LLM-based chatbot.

CATEGORY	NAME	TEXT TYPE	SOURCE
2024 LECTURES	Lectures 1 - 11 Section A	Video Transcript	Panopto
2024 PREVIEWINGS	Previewing MNIST	Video Transcript	Dropbox
	Previewing MaxPool		
	Previewing FineTuning		
2024 RECITATIONS	Recitations 1 - 3	Video Transcript	Zoom
COURSE BOOK	Deep Learning with Python	.pdf-file	Canvas
2024 SYLLABUS	Course Syllabus	.pdf-file	Canvas

This data will constitute part of the corpus for our model, thus allowing the Chatbot to “finetune” its knowledge to the specifics of the course.

2.2 Data Preprocessing

Before inputting it into the model, we preprocessed the corpus data. This process consisted of:

1. **Normalizing data:** we removed special characters, headers, footers, as well as non-period punctuation, such as columns, semi-columns, and commas. This allows the following model to receive as input a text that is cleaner, more interpretable, and easier to process.

This process applies to both the textbook data and the transcripts

2. **Clean data:** we further cleaned our data by removing unnecessary white spaces, new lines, and tabs. We also removed HTML tags, in-text citations, and hyperlinks. Again, this is done to make sure that the following model will focus on the most relevant information (i.e., the content per se), without being led astray by unnecessary text details.

This process, on the other hand, applies more to textbook data, as transcripts did not contain links or citations



3 Methodology

3.1 Link to Colab Notebook

<https://colab.research.google.com/drive/1ydfntyYo-BdlCNAS-9VWjdKfbu9-4gZP?usp=sharing>

3.2 Description of Methodology

The methodology we utilized to create the LLM-based Q&A chatbot is the LangChain Retrieval-Augmented Generation (RAG) model, which we implemented as follows:

First, we split our corpus (comprised of textbook, class transcripts and TA sessions transcripts) into **semantic chunks**. This is a necessary step for RAG, as what the algorithm will do - when prompted a new question - will be to search among the semantic chunks which are the most relevant, and extract information from those. We decided to split the text semantically rather than by length or full stops to preserve somewhat of a meaning in the way the chunks are created, and make sure that a whole piece of content is not randomly split into two chunks.

We then initialized a **client** to interact with a **Weaviate instance** - i.e., a vector search engine that allows storing, querying, and retrieving data based on vector embeddings – and call a function to create vector stores from the passed documents.

Next, we initialize the **retriever** based on the vector stores created before, which can then be used to perform semantic search queries within the vector space, enabling the finding of documents that are semantically similar to a query even if they do not share exact keywords.

Moreover, we wanted students to be able to have a full conversation with the Chatbot, not just ask one question at the time, thus, we utilized the LangChain's capabilities to construct **dynamic prompts** on a predefined template and the chat history, ensuring the previously generated context is considered in generating responses. We therefore implemented a contextualizer, which is called only when the prompt is not the first of the conversation.

This **contextualizer** takes the query as input but uses a placeholder to add the chat history to contextualize the query. The contextualizer also has its own system prompt: instead of telling it to be a question/answer assistant, here it instructs to rephrase the query in a standalone question that is contextualized given the chat history but can be understood without it. This allows us not to pass the full chat history into the final query in addition to the chunks retrieved by the retriever, which could overwhelm the system.

The **generative part** of this algorithm was implemented using a GPT API. For this part, we relied on GPT 3.5 turbo, as it allowed us to have enough tokens to answer to feed it our chunks while costing less than GPT 4 turbo.



Lastly, we built a **web-based User Interface**, to make sure that this algorithm was fully and easily usable by students at any time. The interface was created using the Gradio.ChatInterface library, which allows to easily integrate our algorithm in the interface. Here, we implemented a function that extracts the context of a query, which is then used by the Chat Interface generating function to extract the content of what it is to be generated.


However, at the time we started implementing the **Gradio interface**, we had not yet created dynamic prompts. Our function could therefore only reply to one question at a time and did not take chat history as input. This caused a problem as ChatInterface gives two inputs to the function: chat history and the query. This is why we first had to solve dynamic prompts before implementing the user interface.

Furthermore, another problem we encountered was the **format of the inputs**. Indeed, ChatInterface inputs the chat history in a format that could not be interpreted and then leveraged by LangChain. This caused the user interface to correctly answer the first question (as the chat history given into input is just an empty string and therefore, we do not call the contextualizer at that point) but would output an error at the second question. We then had to change the format of the chat history to adapt it to our function and finally make the whole Chat Interface work.

4 Discussion of Results

4.1 Exemplary Chat History

Below, we show are shot exemplary chat with our chatbot. We are covering different topics, ask follow-up questions and request to compare concepts from previous answers. In addition to this, we ask it to return pure text answers, mathematical answers, and code answers. Generally, the performance below is representative of general performance.

 Chatbot

What are Neural Networks ?

Neural networks are designed to transform input data to make different classes separable, similar to uncrumpling a paper ball. They excel at finding representations for complex data manifolds by decomposing geometric transformations into elementary ones. Each layer in a deep network disentangles data, making the process more manageable. Neural networks use gradient-based optimization to adjust weights and biases, learning from training data to minimize prediction error.

Give me 3 applications

Neural networks are commonly used in image classification tasks, such as identifying objects in images or detecting faces. They are also applied in natural language processing tasks, like sentiment analysis or language translation. Additionally, neural networks are used in speech recognition systems, enabling devices to understand and respond to spoken commands.

What are LLMs ?



LLMs, or Large Language Models, are advanced models used in natural language processing tasks. They are capable of generating human-like text and have been instrumental in various applications such as text classification, sentiment analysis, and language translation. LLMs leverage vast amounts of data and complex neural network architectures to understand and generate text with high accuracy and fluency.

What is the difference between the two ?

The difference between neural networks and LLMs lies in their specific applications and architectures. Neural networks are a broader category of models used for various tasks like image classification and speech recognition, while LLMs are specialized models designed specifically for natural language processing tasks. Neural networks typically consist of layers of interconnected nodes, while LLMs often involve transformer architectures with attention mechanisms for processing sequential data like text.

What is the professor's name ?

The professor's name is Professor Ramakrishnan.

What is his background ?

Professor Ramakrishnan grew up in India and pursued his master's and Ph.D. in the United States. He has been involved in various startups, including one called Quotient, now known as Salesforce Einstein for ecommerce. Additionally, he has served as the head of analytics at Oracle and the head of data science at Salesforce. His interests lie in applying AI to business problems, particularly in areas like education, healthcare, and climate change.

Can you give me the formula of the back propagation in a convolutional layer ?

I don't know the specific formula for back propagation in a convolutional layer.

how to geometrically interpret tensor operations ?

Tensor operations can be interpreted as simple geometric transformations of input data. For example, addition of tensors can be visualized as translation, rotation, scaling, or skewing of objects in a geometric space. Each tensor operation corresponds to a specific geometric transformation, allowing for a visual understanding of how data is manipulated and transformed within a neural network.

Give me the code to code a dense layer in keras with 128 neurons and rely activation

To create a dense layer in Keras with 128 neurons and ReLU activation, you can use the following code snippet:

```
from tensorflow import keras
from tensorflow.keras import layers

model = keras.Sequential([
    layers.Dense(128, activation='relu')
])
```

This code defines a sequential model with a dense layer of 128 neurons and ReLU activation function.



4.2 Technical Results

It is hard to evaluate a generative model's effectiveness; this is because, since it does not generate one specific output format regardless of the input, it is hard to produce a universal accuracy metric. Among the state-of-the-art metrics there are:

- **BLEU (Bilingual Evaluation Understudy):** Compares the generated text to one or more reference texts, measuring the similarities based on the precision of n-grams. It is mostly used for evaluating machine translation quality.
- **ROUGE (Recall-Oriented Understudy for Gisting Evaluation):** Like BLEU but focuses on recall rather than precision. It is used to evaluate the quality of summarization by measuring the overlap of n-grams between the generated summary and reference summaries.
- **METEOR (Metric for Evaluation of Translation with Explicit Ordering):** A metric that considers exact word matches, synonyms, and stemming for evaluation, providing a more nuanced comparison than BLEU. It is designed for machine translation but is also applicable to general text generation tasks.

All these metrics, in order to be exploited, would have required the generation of mock correct answers to different prompts, which would have been infeasible in the time frame of the project. Thus, in the interest of time, we evaluated our model's performance more qualitatively.

We tested our model on different prompts and detected whether it showed hallucinations or factual mistakes.

Indeed, we tested the Chatbot on several "critical prompts" among which:

- Prompts that it is not supposed to know the answer to, to test that it was replying "I don't know the answer" when it did not:
 - e.g., "What's Jan's favorite color?"
- Prompts that are not only related to the technical content, but more tailored to the corpus we fed to the algorithm. This was done to make sure that indeed our corpus was being exploited correctly and the Chatbot was tailored on the Hands-on Deep Learning class.
 - e.g., "What's the professor's full name?"
 - e.g., "What's TA Gabriel's last name?"
 - e.g., "Around which chapter of the textbook is Backpropagation mentioned?"

What we observed is that the model performed perfectly on the first two, while it showed some imprecision on the third kind of prompts. This is because we are not implementing a classification algorithm, therefore – understandably – the model is not instructed to classify where topics come from. Nonetheless, it was trained on the Chapters' titles, which were embedded close to part of their own content: this is why



the Chatbot's answers to these kinds of questions are not completely wrong (off by one chapter or one subchapter), but overall, not always precisely correct.

- Prompts that he could know about using its previous training on a lot of data but is not related to the content of the class:
 - e.g., "What is a tartiflette?"

What we observed is that in that case it answers the question and does so accurately, even though there is of course no mention of tartiflette in the material we fed it. We tried to do prompt engineering to ask it to only answer queries using information given in the retrieved chunks. However, this led to a decrease in performance as it sometimes answers it does not know when it used to answer correctly. Therefore, we decided to remove that prompt engineering.

4.3 Lessons Learned and Future Work

The Chatbot generated by our model can answer questions about the class content as well as class-related information (e.g., the professor's name). However, it cannot **input nor output images**. An upgrade to this work would be to improve this model by letting take images as inputs, for instance a problem that a student might have, understand it, and then answer it. Outputting images could also have a positive impact as it would allow the bot to explain more graphically concepts.

Another upgrade that we can think of would be to train it to **code**, to be able to help the students with their assignments with hints in a less obvious manner than would the current AI-powered coding assistants. Moreover, we could implement a parallel agent which, instead of giving the full solution directly, could lead the student in the right direction, which would help them learn better and engage more in finding solutions.

One issue we ran into when testing the model concerns the contextualizer for chat history. This agent is instructed to take the history of the conversation, along with the current question, and reformulate a question which merges the history with the actual question. By instructing the agent in this way, some issues arise, for example:

- When you ask a question that is not a follow-up to the previous conversation, but concerns a different topic, the bot will sometimes output wrong answers, hallucinating on what the actual question is. Conversely, if you ask the same question in a brand-new chat, with no history, the bot will always answer correctly.

The above example shows that, by instructing the contextualizing agent to answer a new question by using all the context above – rephrasing it in a made-up question – what might happen is that it aggregates topics that have nothing to do with the new question, thus increasing the chance of hallucination.



One possible solution, that open paths for further testing, is:

- Instead of instructing the contextualizer to build a new question using history + current prompt, we could instruct it to use history of questions-answers as some example prompt engineering, for example:

Our current contextualizer is prompted as follows:

Given a chat history and the latest user question which might reference context in the chat history, formulate a standalone question which can be understood without the chat history. Do NOT answer the question, just reformulate it if needed and otherwise return it as is.

While we could try prompting it like:

Given question-answer tuples of the chat history and the latest user question which might reference context in the chat history, formulate a standalone question. Which uses the tuples as mock engineered prompts. The standalone question must be understood without the chat history. Do NOT answer the question, just reformulate it if needed and otherwise return it as is.

This requires a significant amount of testing and trial and error, which in the interest of time we just hint to as possible future work.