**Project III | Business Case: Building a Multimodal AI Chatbot for YouTube Video QA**

**Report**

Loz Rigby

**Contents**

1. Introduction
2. Downloading and text extraction
3. Tokenising, Text splitting, Embedding and Indexing for Pinecone
4. Initialising the VectorStore
5. Querying
6. Evaluation on queries
7. Deployment
8. Limitations and improvements
9. Conclusion
10. **Introduction**

**Project Overview**

The goal of this final project is to develop an AI bot that combines the power of text and audio processing to answer questions about YouTube videos. The bot will utilize natural language processing (NLP) techniques and speech recognition to analyse both textual and audio input, extract relevant information from YouTube videos, and provide accurate answers to user queries.

The objective of this project is for Stanford students who are in the Machine Learning Class to use the bot to ask questions on all their lectures, to avoid having to search through long lectures and notes.

**Key Objectives**

1. Develop a text-based question answering (QA) model using pre-trained language models. You may find it useful to fine-tune your model.
2. Integrate speech recognition capabilities to convert audio into text transcripts.
3. Build a conversational interface for users to interact with the bot via text.
4. Retrieve, analyse, and store into a vector database using Pinecone YouTube video content to generate answers to user questions.
5. Test and evaluate the bot's performance in accurately answering questions about YouTube videos.

**Tools and Technologies**

* Shutil
* Tqdm
* Uuid
* Whisper
* Openai
* Ffmpeg
* Streamlit
* Python
* Numpy
* Langchain
* Langchain-Openai
* Langchain-Community
* Langgraph
* Youtube\_Dl
* Pydub
* Openai
* Tiktoken
* Pinecone
* Langsmith.Evaluation
* Google.Colab
* Os
* Glob
* shutil
* tqdm
* uuid
* whisper
* openai
* Ffmpeg

1. **Downloading and text extraction**

**Downloading Videos from YouTube**

Firstly, videos were selected from Stanford YouTube Channel Machine Learning Playlist. It consisted of 19 videos, accumulating nearly 26 hours’ worth of lectures. The videos were downloaded into a directory stored locally to later be processed.

**Audio chunking**

The audio files are then processed. The main function, split audio, takes an audio file and splits it into smaller chunks, each 10 minutes long. For each chunk, a new MP3 file is created and saved, with the filenames indicating their respective parts. The function then returns a list of these chunk filenames. Audio files are chunked to make them more manageable for transcription services or APIs, which often have limits on the length or size of the audio files they can process at one time. Chunking also helps in efficiently handling, uploading, and processing audio data in smaller segments.

**Code Snippet of chunking:**

A screen shot of a computer code

Description automatically generated

**Transcribing via Whisper**

The chunked audio files from the directory are transcribed into text using the Whisper base model. The code first checks if the audio files are larger than 25 MB. If they are, it splits them into smaller chunks to make them more manageable for transcription. For each chunk or the entire audio file (if under 25 MB), the audio is loaded and transcribed using Whisper's transcribe function. The resulting transcript is saved as a text file in the output directory.

Chunking is necessary to handle file size limits of transcription models and APIs, manage memory usage efficiently, and facilitate error handling by processing smaller, manageable segments. It also allows for parallel processing, speeding up the transcription process, and provides incremental results for easier review and usage. This approach ensures that large audio files can be effectively and efficiently transcribed which is essential when all audio files are over an hour long like in this case.

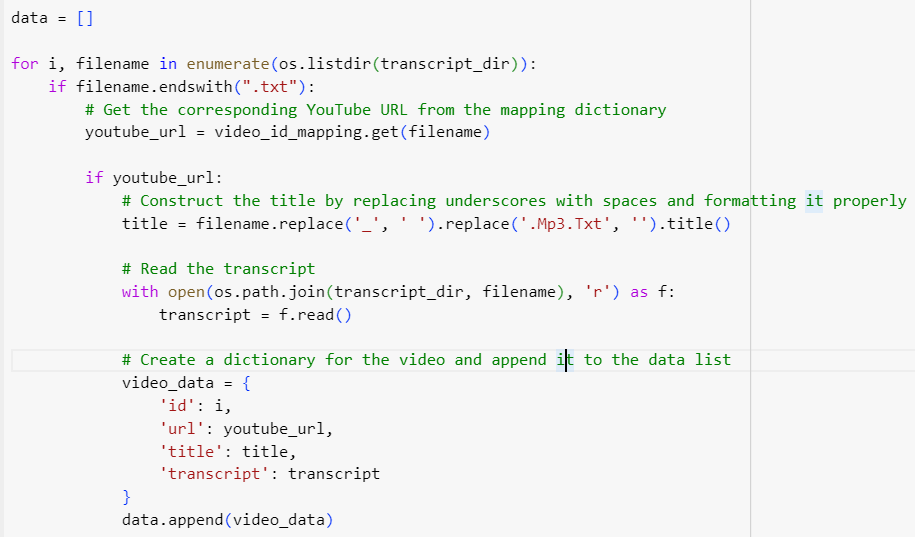
Whisper is an effective model for transcription due to its state-of-the-art performance in speech recognition, developed by OpenAI. It can handle a wide variety of audio qualities and accents. Additionally, Whisper's robustness in transcribing speech even in noisy environments such as in the lecture theatre and its ability to handle long-form audio inputs make it ideal for this task of transcribing lectures. The model's ability to efficiently process and generate accurate transcriptions helps streamline workflows that require converting audio to text.

These chunked transcripts were rejoined so each lecture had a singular full transcript file with accuracy checked manually by watching the videos and readying along with the transcriptions.

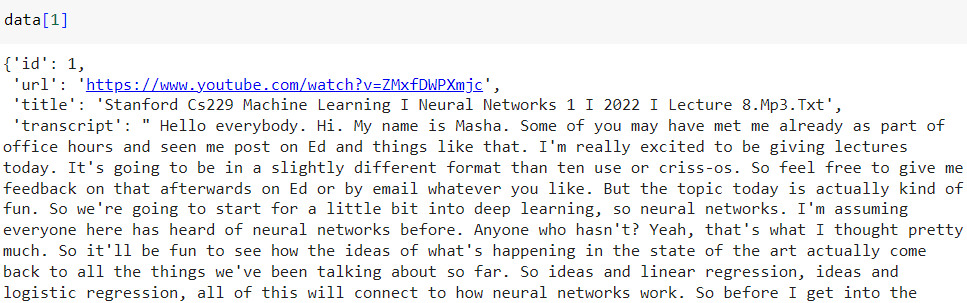
**Preprocessing**

Firstly, to ensure that the URLs are aligned to the correct transcripts within the dictionaries, the titles are mapped to the corresponding URLs. Top of Form

After this, the transcript files are processed and stored in a directory to construct a structured dataset of video information. It iterates through each text file in the directory, checking for files with a .txt extension to ensure only transcript files are processed. For each file, it retrieves the corresponding YouTube URL from the video dictionary, then constructs a readable title by replacing underscores with spaces and capitalizing the words and removes the file extension (‘. Mp3.Txt'). It then reads the content of each transcript file and creates a dictionary containing an ID (index), the YouTube URL, the formatted title, and the transcript content. This dictionary is appended to the data list. This process organizes the transcript data, linking each transcript to its corresponding YouTube video, and prepares the data for further use, such as display or analysis.

Bottom of Form

Creating dictionaries in the list data that look like this when inspected:



1. **Tokenising, Text splitting, Embedding and Indexing for Pinecone**

**Tokenisation and text splitting**

Then the transcript text is tokenized and split into manageable chunks using specific criteria. Setting up a tokenizer based on the 'cl100k\_base' encoding from tiktoken, which is tailored for the GPT-3.5-turbo model. The number of tokens is calculated to ensure the text is properly segmented according to token count rather than character count.

Next, Langchain Recursive Text Splitter is used to split the text. It uses a chunk size of 400 tokens with an overlap of 20 tokens between chunks. The split is performed using a hierarchy of separators: double newlines, single newlines, spaces, and finally, no separator if needed. This approach ensures the text is divided at logical points, like paragraphs or sentences, whenever possible.

Splitting the text into smaller chunks is necessary for efficiently processing and analysing long texts, especially for tasks like generating embeddings, or feeding into language models, which often have token limits. This ensures that the text remains coherent within each chunk while facilitating easier handling and processing.

Code snippet:

A screenshot of a computer code

Description automatically generated

**Embedding**

To generate embeddings for text data OpenAI's text-embedding-ada-002 model was used. This setup configures the environment to generate embeddings, which are numerical representations of text capturing its semantic meaning. These embeddings are crucial for various natural language processing tasks, such as similarity searches, clustering, and classification, enabling sophisticated text analysis and processing.

A screenshot of a computer code

Description automatically generated

**Indexing for Pinecone**

Firstly, a Pinecone vector index is being created and set up for storing and querying embeddings generated by the `text-embedding-ada-002` model from OpenAI. First, a Serverless Spec is instantiated to specify the cloud provider (AWS) and region (`us-east-1`) for the Pinecone service. The code then defines the name of the index which will be `transcription-index`. The index is created with a specified dimensionality of 1536, so it matches the output dimensions of the `text-embedding-ada-002` model and uses the ‘dotproduct’ metric.

It then connects to the index and retrieves its statistics to confirm successful setup.

This process is necessary to ensure that the infrastructure for storing and querying text embeddings is properly configured, enabling efficient retrieval and analysis of large-scale semantic data for applications such as search, recommendation, and clustering.

The data records text embeddings were batch uploaded to the Pinecone index for efficient retrieval and analysis. The batch limit was 100, which sets the maximum number of text chunks processed and uploaded in one batch. The texts and metadata lists are initialized to store text chunks and their associated metadata, respectively.

For each record, it first extracts metadata fields, including the record's ID, source URL, and title. Then the transcripts are into smaller chunks using the text splitter object. Metadata for each chunk is created, including the chunk number and the transcript text, and these are appended to the texts and metadata lists.

When the batch limit is reached, a batch upload is triggered. Unique IDs are generated for each chunk, and embeddings for the text chunks are generated using the embedding model set up previously. These embeddings, along with their metadata, are uploaded to the Pinecone index using the upsert method. After each batch upload, the texts and metadata lists are cleared to prepare for the next batch. If there are any remaining text chunks after the loop finishes (i.e., a partial batch that did not reach the batch limit), a final upload is performed using the same process.

This approach is necessary for several reasons. By processing and uploading text embeddings in batches, the code can handle large datasets efficiently without overwhelming memory or computational resources. Batching ensures that the system can scale to accommodate very large datasets, as it processes manageable portions of the data at a time. Batching makes it easier to isolate and manage errors during the embedding and upload process, improving reliability.

A screen shot of a computer code

Description automatically generated

1. **Initializing the VectorStore**

The VectorStore is then initialised using the Pinecone service to facilitate efficient storage, retrieval, and querying of text embeddings. The text selection variable is set to "transcript", indicating that the metadata field containing the text data to be indexed and queried is labelled "transcript." Langchain Vectorstores is then instantiated with the Pinecone ‘transcriptions-project’ index. This setup creates a vectorstore object that integrates Pinecone's vector database with the embedding function, enabling the system to store text embeddings and perform similarity searches or other vector-based queries efficiently. By linking the embedding function and the text field, the vectorstore object ensures that queries can be processed by converting input text into embeddings, which are then matched against the stored embeddings in the Pinecone index for relevant results. This configuration is essential for this application as it requires fast and scalable text retrieval and analysis.

Pinecone snippet:

A screenshot of a computer

Description automatically generated

1. **Querying**

**Similarity Querying testing**

Similarity search was tested using the pinecone vectorstore to find the most relevant documents for a given query. The query, "Why do you split testing and training data?" is passed to the similarity search method, which uses the pre-configured embedding function to convert the query into an embedding. The method then searches the Pinecone index to find and return the top 3 most relevant documents based on their similarity to the query embedding. This allows for efficient retrieval of documents that are semantically related to the query.

A screenshot of a computer screen

Description automatically generated

**Generative Question-Answering**

This code snippet sets up a system for querying video transcripts using a language model from OpenAI, combined with a vector store for efficient information retrieval. The process begins with initializing a language model (LLM) using the ChatOpenAI class, with GPT-3.5-turbo as the model, and setting the temperature to 0.0 to ensure deterministic responses. This language model will be used to generate and process natural language queries.

Next, memory is added using Conversation Buffer Memory to manage the conversation's memory, allowing the system to maintain context and track the dialogue history. Also to improve coherence and contextuality of the conversation.

A retriever is then created, integrating the LLM with this based on the previously configured ‘transcriptions-index’ Pinecone Vectorstore. The vectorstore is converted into a retriever, which facilitates searching and retrieving relevant text segments from the stored lecture embeddings. The chain type parameter is set to "stuff," indicating a specific method of chaining operations. The `memory` parameter integrates the conversation memory, ensuring the retrieval process is context-aware and able to refer back to previous interactions.

The next step involves defining a tool for the React agent. The tool, named ‘video transcript retriever’, is configured to use the query run function, which runs the retriever to search and return excerpts from the video transcripts. The tool's description clarifies its purpose and functionality, making it easier to integrate into the agent's operations.

Finally, the React agent is initialized, which combines the LLM, tools and the memory. The agent type is specified as "conversational-react-description," indicating that it will handle conversational queries and provide descriptive responses based on the retrieved information. This step integrates all components, setting up a system capable of engaging in natural language queries about lecture, retrieving relevant information efficiently, and maintaining conversational context.

In addition, no external knowledge agent was added as the aim was to retrieve information solely from what was discussed in the lectures.

Overall, each step in this process is designed to create a robust and efficient query system that leverages advanced language models and vector-based retrieval to provide precise and contextually relevant answers, enhancing the user's ability to interact with and extract information from Stanford machine learning lectures.

A screenshot of a computer program

Description automatically generated

A close-up of a text

Description automatically generated

This is an example of the query: “Why do you split testing and training data?” being called. Being processed by the retriever, converting the query to an embedding, then using the Pinecone Vectorstore to find the most relevant text for this query. Then passed through the LLM to generate a coherent and contextually accurate response.

With the memory ensuring any relevant context from previous interactions is considered.

The final output is a structured response that includes the generated answer to the query.

1. **Evaluations on Queries**

Queries were then evaluated using Langchain evaluators on correctness, conciseness and coherence. These were compared about self-written reference answers.







A close-up of a computer code

Description automatically generated

**Model Answer**: The names of the lecturers mentioned in the context are Tong Yima and Chris.

**Reference Answer:** Tong Yima and Chris

Example: (Langchain evaluator tool changed according to each evaluation)

A screenshot of a computer program

Description automatically generated

**Correctness:** Reasoning: 1. Correctness: The submission mentions Tong Yima and Chris as the names of the lecturers. The reference also lists Tong Yima and Chris as the names of the lecturers. Therefore, the submission is correct, accurate, and factual., 'value': 'Y', 'score': 1.

**Conciseness**: Reasoning: 1. Is the submission concise and to the point? The submission mentions the names of the lecturers, Tong Yima and Chris, which directly answers the question without any unnecessary information. The submission is concise and to the point. 'value': 'Y', 'score': 1

**Coherence:** Reasoning: 1. Coherence: The submission states that the names of the lecturers mentioned in the context are Tong Yima and Chris. The reference provided also includes Tong Yima and Chris. The submission is clear, well-structured, and organized in presenting the information. The submission is coherent and directly addresses the question asked. Therefore, based on the coherence criterion, the submission meets the criteria. 'value': 'Y', 'score': 1

**Query 4:** ‘What is linear regression?’

**Model Answer:** Linear regression is a statistical method used to model the relationship between a dependent variable and one or more independent variables by fitting a straight line to the data points. It is commonly used for predicting continuous outcomes and is a fundamental technique in statistics and machine learning.

**Reference answer:** In machine learning, linear regression is used to predict a continuous outcome variable based on one or more predictor variables by fitting a linear relationship. It finds the line of best fit that minimizes the sum of squared differences between observed and predicted values.

**Correctness**: Reasoning: '1. Correctness: The submission correctly defines linear regression as a statistical method used to model the relationship between a dependent variable and one or more independent variables by fitting a straight line to the data points. The submission accurately states that linear regression is commonly used for predicting continuous outcomes and is a fundamental technique in statistics and machine learning. The submission accurately describes the purpose and process of linear regression in machine learning. Based on the comparison with the reference provided, the submission is correct, accurate, and factual. Therefore, the submission meets the Criteria. 'value': 'Y', 'score': 1

**Conciseness:** Reasoning': 'Step by step reasoning. The submission provides a clear and concise definition of linear regression. The submission does not include any unnecessary information and directly addresses the question asked. The submission is focused on explaining the concept of linear regression without going into unnecessary details. Conclusion: The submission meets the criteria for conciseness. 'value': 'Y', 'score': 1

**Coherence:** Reasoning'**:** 1. Coherence: The submission provides a clear definition of linear regression as a statistical method used to model the relationship between variables. It explains the purpose of linear regression in predicting continuous outcomes and its significance in statistics and machine learning. The submission is well-structured, starting with a definition and then providing additional information about the application and importance of linear regression. Based on the step-by-step reasoning, the submission meets the criteria for coherence. 'value': 'Y', 'score': 1

**Query 5**: ‘What is the type of generative learning algorithms discussed?’

**Model answer:** The type of generative learning algorithms discussed are Gaussian discriminative analysis.

**Reference answer:** Gaussian Discriminant Analysis (GDA)

**Correctness:** Reasoning: '1. Correctness: The submission states that the type of generative learning algorithms discussed are Gaussian discriminative analysis. However, the correct type should be Gaussian discriminant analysis, not Gaussian discriminative analysis. This is an incorrect statement. Therefore, the submission does not meet the criteria of correctness. 'value': 'N', 'score': 0

**Conciseness:** Reasoning: '1. Conciseness: The submission directly answers the question without any unnecessary information. The submission is clear and to the point, mentioning Gaussian discriminative analysis as the type of generative learning algorithms discussed. The submission is concise and does not include any extra details that are not relevant to the question. Therefore, based on the criteria of conciseness, the submission meets the criteria. Y', 'value': 'Y', 'score': 1

**Coherence:** Reasoning: '1. Coherence: The submission states that the type of generative learning algorithms discussed are Gaussian discriminative analysis. However, the correct term should be Gaussian Discriminant Analysis (GDA) as per the reference provided. The submission is clear and concise in its response, directly answering the question asked. The submission is not well-structured as it contains a minor error in the term used. Therefore, based on the coherence criterion, the submission does not fully meet the criteria. 'value': 'N', 'score': 0

**Query 6:** What is the difference between unsupervised learning and supervised learning?

**Model Answer:** The main difference between unsupervised learning and supervised learning is the presence of labels in the data. In supervised learning, the data comes with labels that the model learns to predict, such as class labels or target values. On the other hand, in unsupervised learning, the data does not have any labels, and the model needs to find patterns or structures in the data without explicit guidance from labelled examples.

**Reference Answer:** Unsupervised learning finds patterns or structures in unlabelled data, while supervised learning uses labelled data to train models for predicting outcomes.

**Correctness**: Reasoning: '1. Is the submission correct, accurate, and factual? The submission correctly states that the main difference between unsupervised learning and supervised learning is the presence of labels in the data. It accurately describes supervised learning as having labelled data that the model learns to predict, and unsupervised learning as lacking labels. The submission is factual in stating that in supervised learning, the model learns to predict labels, while in unsupervised learning, the model finds patterns or structures in the data without labels. Therefore, the submission meets the criteria. Y', 'value': 'Y', 'score': 1

**Conciseness:** Reasoning: 'Step by step reasoning: The submission provides a clear explanation of the main difference between unsupervised learning and supervised learning. The submission is concise and directly addresses the difference between the two types of learning. The submission does not include any unnecessary information and focuses on the key point of the presence of labels in the data. Therefore, the submission meets the criteria for conciseness. Y', 'value': 'Y', 'score': 1

**Coherence**: Reasoning: '1. Coherence: The submission clearly explains the main difference between unsupervised learning and supervised learning. It is well-structured, starting with a clear introduction and then providing detailed explanations for each type of learning. The submission is organized in a logical manner, first discussing supervised learning and then moving on to unsupervised learning. Based on the above reasoning, the submission meets the coherence criterion. Y', 'value': 'Y', 'score': 1

**Query 7:** Differences between reinforcement and unsupervised/supervised learning?

**Model Answer:** Reinforcement learning involves making sequential decisions to maximize rewards in an environment, where the agent learns through trial and error. It focuses on learning optimal actions based on feedback received from the environment. On the other hand, supervised learning involves learning from labelled data, where the algorithm is trained on input-output pairs to make predictions. Unsupervised learning, like clustering algorithms, deals with finding hidden patterns or structures in unlabelled data without explicit feedback or guidance. The key difference lies in the nature of the learning process and the type of information available during training.

**Reference Answer:** Reinforcement learning trains agents to make decisions by maximizing rewards, while supervised learning uses labelled data for predictions and unsupervised learning identifies patterns in unlabelled data.

**Correctness**: Reasoning: '1. The submission correctly explains that reinforcement learning involves making sequential decisions to maximize rewards in an environment through trial and error, while supervised learning involves learning from labelled data and unsupervised learning deals with finding hidden patterns in unlabelled data. The submission accurately describes the key differences between reinforcement learning, supervised learning, and unsupervised learning. The submission provides factual information about the nature of the learning process and the type of information available during training for each type of learning.', 'value': 'Y', 'score': 1.

**Conciseness:** Reasoning: 'Step by step reasoning: The submission provides a concise explanation of the differences between reinforcement learning, supervised learning, and unsupervised learning. The submission covers the key points of each type of learning in a clear and straightforward manner. The submission does not include unnecessary details or elaborations, sticking to the main differences between the learning types. Conclusion: The submission meets the criteria for conciseness. Y', 'value': 'Y', 'score': 1

**Coherence:** Reasoning: '1. Coherence: The submission provides a clear and concise explanation of the differences between reinforcement learning, supervised learning, and unsupervised learning. It is well-structured, starting with a definition of reinforcement learning and then moving on to supervised and unsupervised learning. The submission is organized in a logical manner, comparing and contrasting the three types of learning methods.\n- The information is presented in a coherent way, with a clear focus on the key differences between the learning processes. Therefore, based on the above reasoning, the submission meets the criteria for coherence. Y', 'value': 'Y', 'score': 1

**Query 8:** What is an example of using AI systems in healthcare for heart conditions?

**Model Answer:** One example of using AI systems in healthcare for heart conditions is the development of a computer vision system that analyses ultrasound videos of the human heart to assess different cardiac conditions in patients. This system was developed at Stanford and has been deployed in healthcare settings to help healthcare professionals in diagnosing and monitoring heart conditions.

**Reference Answer:** Computer visions system for assessing heart conditions.  The idea is that there are ultrasound videos looking at the human heart.  A system was developed to read these videos and assess different cardiac conditions of a patient.

**Correctness**: Reasoning: '1. Correctness: The submission accurately describes the use of AI systems in healthcare for heart conditions, specifically mentioning the development of a computer vision system that analyses ultrasound videos of the human heart to assess different cardiac conditions in patients. The submission also correctly states that this system was developed at Stanford and has been deployed in healthcare settings. The information provided in the submission aligns with the reference provided. Therefore, the submission meets the criteria.', 'value': 'Y', 'score': 1.

**Conciseness**: Reasoning: '1. The submission provides a clear and concise example of using AI systems in healthcare for heart conditions. It directly mentions the development of a computer vision system that analyses ultrasound videos of the human heart to assess different cardiac conditions in patients. The information is presented in a straightforward manner without unnecessary details. Therefore, the submission meets the criteria for conciseness. Y', 'value': 'Y', 'score': 1

**Coherence**: Reasoning: '1. Coherence: The submission is coherent as it clearly explains the example of using AI systems in healthcare for heart conditions. The submission is well-structured, starting with the example and providing details about the development and deployment of the system. The submission is organized, with a clear flow of information from the introduction of the example to its application in healthcare settings. Therefore, the submission meets the criteria.', 'value': 'Y', 'score': 1.

**Query 9**: What is high level summary of backpropagation?

**Model Answer:** The high-level summary of backpropagation is that it is a technique used to compute the gradient of the loss function for a neural network. This gradient calculation is crucial for updating the parameters of the network during training, typically using algorithms like stochastic gradient descent. The process involves computing the gradient of the loss function with respect to the network's parameters by propagating the error backwards through the network layers. This allows the network to learn and improve its performance over time.

**Reference Answer:** Backpropagation is an algorithm used in training neural networks, where the error is calculated at the output and propagated backward through the network to update the weights. This process uses the gradient descent optimization technique to minimize the error by adjusting the weights iteratively.

**Correctness:** Reasoning: "The submission accurately describes backpropagation as a technique used to compute the gradient of the loss function for a neural network. The submission correctly states that the gradient calculation is crucial for updating the parameters of the network during training, typically using algorithms like stochastic gradient descent. The submission accurately explains the process of computing the gradient of the loss function with respect to the network's parameters by propagating the error backwards through the network layers. The submission correctly mentions that backpropagation allows the network to learn and improve its performance over time. Therefore, the submission meets the Criteria.", 'value': 'Y', 'score': 1.

**Conciseness:** Reasoning: 'Step by step reasoning: The submission provides a high-level summary of backpropagation in a clear and concise manner. The submission explains the purpose and process of backpropagation effectively without unnecessary details. The submission stays focused on the main points without going off on tangents. Conclusion: The submission meets the criteria.', 'value': 'Y', 'score': 1

**Coherence**: Reasoning: 'Step by step reasoning: The submission is coherent as it explains the high-level summary of backpropagation in a logical and structured manner. The submission is well-structured as it follows a clear flow of explaining what backpropagation is, why it is important, and how it works. The submission is organized as it presents the information in a systematic way, starting from the definition of backpropagation to its application in updating neural network parameters. Therefore, the submission meets the Criteria. Y', 'value': 'Y', 'score': 1

1. **Deployment**

**Objective**

The primary objectives of this application are twofold: firstly, to enable users to upload a YouTube video URL, download the audio, transcribe it, and store the transcript; secondly, to provide an interface for users to ask questions related to Stanford CS229 Machine Learning content and retrieve relevant information from the Pinecone Vectorstore ‘transcripts-index’ created previously using natural language processing techniques.

**Process**

Due to issues with API key storage, the first step is to enter the OPENAI and Pinecone keys directly into the app to avoid issues.

To set up the Streamlit environment a Python Script (app.py) is created from our model notebook that was created earlier.

To process a YouTube video uploaded the model repeats what was done to the Stanford lecture YouTube files when they were downloaded in the model. Harnessing the power of the model built earlier for smaller individual videos. But rather than placing them in the ‘transcripts-index’ like what was done with the Stanford videos, an empty pinecone index is initialised, and the embeddings are placed in there. This is so the model can query only on the YouTube video uploaded. This index is deleted after the querying is complete.

To query the Stanford videos, the ‘transcriptions-index’ is called immediately with no data processing necessary as the embeddings are already stored in the Vectorstore. So, when the user inputs a query, the application retrieves the relevant information from Pinecone using the embedded vectors and displays the answer.

**YouTube video example**

****

1. **Limitations and improvements**

**Limitations**

**Handling Long Queries and Context**: The current implementation might struggle with handling very long queries or maintaining context over extended conversations. This can impact the relevance and coherence of responses, particularly for complex or multi-part questions.

**Scalability**: While Pinecone provides efficient vector search capabilities, handling many simultaneous queries or a significantly larger dataset might pose performance challenges. This could result in slower response times or the need for more computational resources.

**Real-Time Processing Constraints**: The current implementation might face challenges with real-time processing and responding to queries on newly uploaded YouTube videos, especially if the videos are lengthy and require extensive transcription and embedding.

**Time constraints**: Only having a week for project completion meant that a lot of potential ideas to improve the model were unable to be completed.

**Personal doubts**: Personal doubts on my own ability to create an efficient bot meant that progress was slow at the beginning, increasing the impact of other limitations such as time constraints.

**Improvements**

To enhance transcription accuracy, fine-tuning the Whisper model specifically for lecture audio would address the unique vocabulary and structure of academic content, significantly reducing errors. Implementing advanced noise reduction techniques before transcription can further improve audio quality, ensuring more precise transcriptions.

Improving context handling for long queries is crucial. Integrating a sophisticated memory system will help the AI bot retain context across extended conversations, enhancing relevance and coherence. Additionally, hierarchical query processing can break down complex queries into smaller parts, allowing the bot to handle each segment effectively, resulting in more accurate responses.

Continuous learning and bias mitigation are essential for maintaining the AI's performance. By incorporating continuous learning mechanisms, the model can adapt to new information over time, improving accuracy. Techniques to identify and mitigate biases will ensure that the responses remain fair and unbiased, increasing the system's credibility.

Enhancing the user interface is vital for better engagement. Developing an interactive UI that supports voice commands, multi-modal inputs, and visual aids will make the system more user-friendly. Integrating a feedback mechanism will allow users to rate the responses, providing data for continuous improvement and ensuring the AI evolves based on user needs and experiences.

1. **Conclusion**

This project demonstrates the potential of integrating advanced natural language processing and speech recognition technologies to enhance educational experiences. The objective was to create an AI bot capable of answering questions about Stanford's machine learning lectures on YouTube. By developing this chatbot, the system provides a valuable tool for students to quickly and efficiently retrieve relevant information from their lectures. The project successfully utilized state-of-the-art models like Whisper for transcription and GPT-3.5-turbo for natural language understanding, combined with the scalable vector search capabilities of Pinecone.

The system offers significant benefits, such as reducing the time spent searching through lecture content and providing precise answers to complex queries. It was positively evaluated, demonstrating its effectiveness in addressing the needs of students. However, addressing issues related to transcription accuracy, handling long queries, scalability, and data privacy will be essential for further improvements. Additionally, extending the system's capabilities to handle a broader range of content and queries will enhance its versatility and usefulness.

Overall, this project underscores the importance of leveraging cutting-edge AI technologies in educational settings and sets the stage for future developments in AI-driven educational tools.