# *Project Progress Report: Information Retrieval using Large Language Models through Question Answering for Machine Learning and Artificial Intelligence Research Papers*

*( CS 410:  Text Information Systems , Fall 2023)*

*(Team: Lexical Wizard)*

| Team : Lexical Wizards | | | |
|---|---|---|---|
| # | Name | ID | Email |
| 1 | Kumar, Amrit | amritk2 | amritk2@illinois.edu |
| 2. | Gattu, Sudha Mrudula | sudhamg2 | sudhamg2@illinois.edu |
| 3. | Madhavan, Siddharth | sm120 | sm120@illinois.edu |
| 4. | Awachat, Ishika | awachat2 | awachat2@illinois.edu |

| Update History | | | | |
|---|---|---|---|---|
| # | Version | Date | Author | Update Details |
| 1 | draft | 2023-11-18 | Amrit Kumar | Draft Document |
| 2 | draft | 2023-11-18 | Siddharth Madhavan | Text embedding section |
| 3 | draft | 2023-11-19 | Amrit Kumar | System Architecture and Virtual Store Setup |
| 4 | draft | 2023-11-19 | Sudha Mrudula | Web Scraping Section |
| 5 | draft | 2023-11-19 | Amrit Kumar, Ishika Awachat | Interactive Chat window section |
| 6 | draft | 2023-11-19 | Amrit Kumar | Summary section and Overall progress |
| 7 | v1 | 2023-11-19 | Siddharth Madhavan | Edits and formatting |

## Table of Contents

## Introduction:

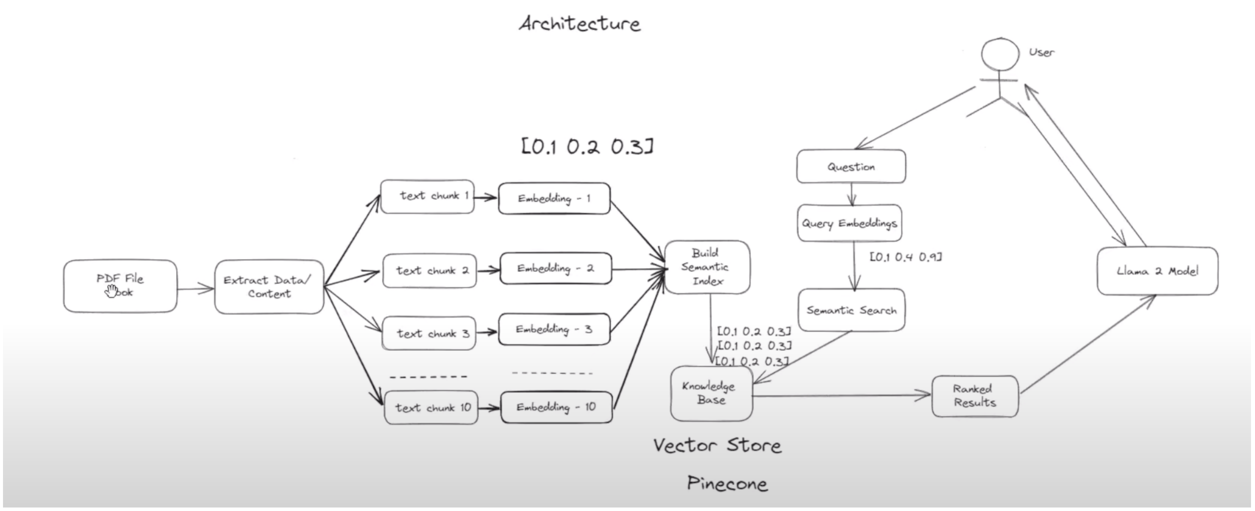
The rapid proliferation of research papers in the fields of Machine Learning and Artificial Intelligence has led to a substantial increase in the volume of textual information. This surge poses a significant challenge for knowledge seekers who wish to access specific information within this vast corpus. To address this issue, we propose a comprehensive project titled "Information Retrieval using Large Language Models through Question Answering." This project aims to leverage the power of modern NLP models to extract, index, and retrieve information from research papers in the domain of Machine Learning and Artificial Intelligence.

The aim of this document is to share the progress on this project implementation. This project has following main modules,

- System Design/ Architecture
- Web scraping:
- Text Chunk generation/ embedding
- Storing embedded data in Index Vector store.
- Information Retrieval and chat conversation using Large Language Model
- Interactive Chat interface

## System Design/Architecture

- This Large Language Model powered application is intended to perform information retrieval from the loaded document/web and provide the personalized answer of user questions through a chat window .
- We are intending to use LLAMA2 (large language model) for the implementation .
- Below is the system design for **LexicalWizard** application



- **Flow**:
  - <u>**Preprocessing**</u>
    - The data (PDF/ Web Data) for a specific domain will be loaded
    - Data from the source will be extracted and splitted.
    - Splitted data (chunks) will be converted into **embeddings** .

        - Embeddings will be saved into a Indexed Vector Database i.e. Pinecone
        - Indexed Vector Store will be acting as a “Knowledge Base” to answer user queries.
    - **Answering User queries**:
        - When user posts his/her question, embeddings will be created for the provided query
        - Similarity search will be performed in the Pinecone vector store using the query embeddings.
        - Similarity Search will return **TOP RANKED** chunks from the index vector store.
        - The r**anked results** will be sent to the **LLAMA2** model along with the user/system prompts to answer the asked queries
        - Answers from the LLAMA2 model will be displayed in the chat window as an answer.
        - For further questions, previous chat conversations will be passed as context to continue the conversation.

## Web Scraping:

1. Which tasks have been completed?
    a. Created a baseline template in which the program accepts a user query and conducts searches on the DuckDuckGo search engine to identify potential and relevant websites that address the query.
    b. Integrated the DDGS (DuckDuckGoSearch) api to retrieve search results with a given query
    c. After retrieving the relevant website for the given query, the program saves the search results into a text file
2. Which tasks are pending?
    a. Rather than fetching websites and urls, implementing the feature to where the program also retrieves PDF documents
    b. Integrating the url lib package to read and parse the urls
    c. Upon parsing the URLs, the program retrieves the most optimal answer for the given query.
3. Any challenges?
    a. Identifying the most suitable website/PDF document from the search results

## Text Chunk generation/ embedding

1. Which tasks have been completed?
    a. We have integrated hugging face api to get text embeddings
    b. We have successfully split a pdf into separate lines and have used chuck_overlap to retain context of that paragraph
    c. We are able to get some questions answered based on a sample hugging face Question and Answer module (proof of concept)
2. Which tasks are pending?
    a. Incorporating the logic to take on multiple pdfs
3. Any challenges?
    a. Improving text embeddings to capture the context for multiple pdfs

## Storing the text embedding in index database

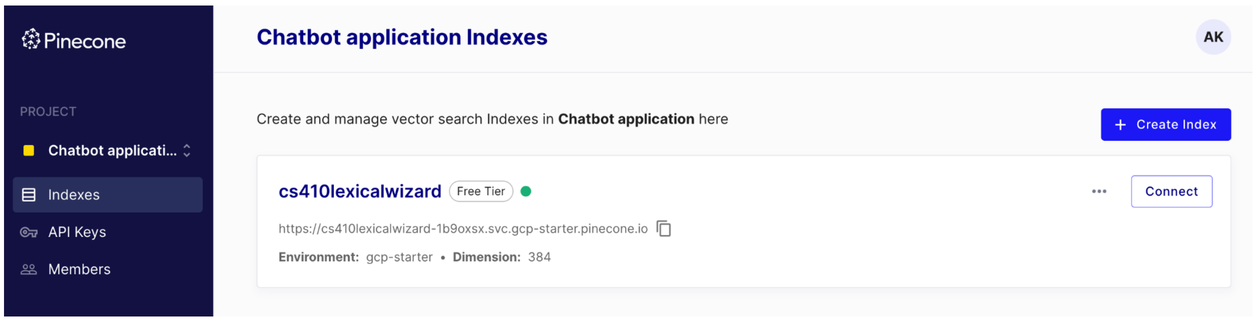
We are using Pinecone Vector Database to store the vector embeddings.

### Why Vector Database

- Applications that involve large language models, generative AI, and semantic search rely on vector embeddings, a type of data that represents semantic information.
- This information allows AI applications to gain understanding and maintain a long-term memory that they can draw upon when executing complex tasks.
- Vector databases like Pinecone offer optimized storage and querying capabilities for embeddings.
- Reference: https://docs.pinecone.io/docs/overview

### Configuring Pinecone Database

- Pinecone vector database will store embeddings for knowledge source chunk embeddings and the embedding of the user questions.
- Setup a index database in Pinecone io: https://app.pinecone.io/



Chatbot application Indexes

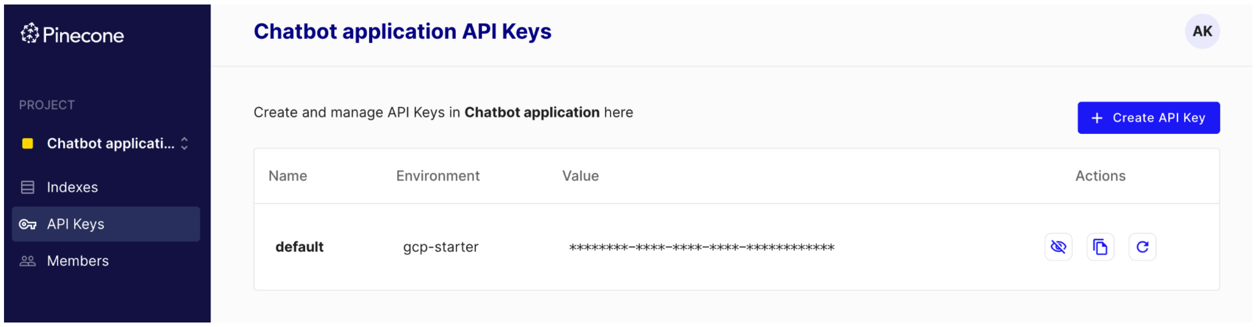
Create and manage vector search Indexes in **Chatbot application** here

cs410lexicalwizard

https://cs410lexicalwizard-1b9oxsx.svc.gcp-starter.pinecone.io

**Environment:** gcp-starter • **Dimension:** 384

- Generate/Copy the `PINECONE_API_KEY` , `PINECONE_API_ENV` values, this will be needed for connecting to the pinecone vector database programmatically.



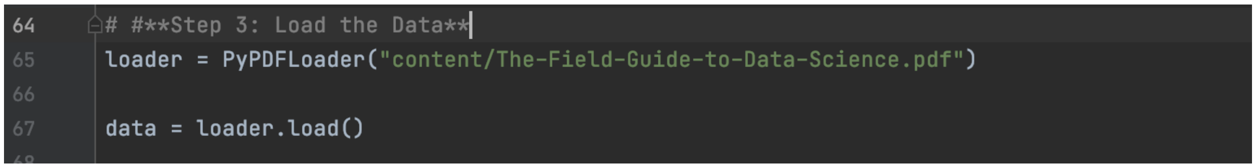
**Chatbot application API Keys**

Create and manage API Keys in **Chatbot application** here

| Name | Environment | Value | Actions |
|---|---|---|---|
| **default** | gcp-starter | ********-****-****-****-************ | |

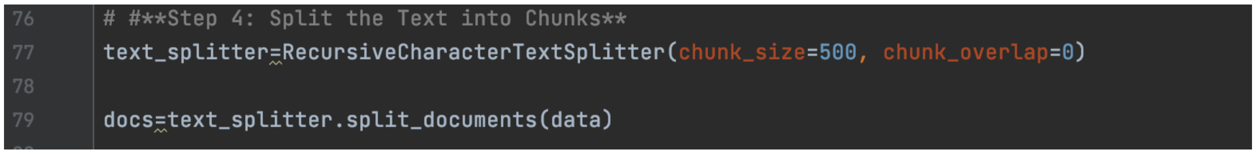
## Using Pinecone Vector database programmatically

- Package to be installed : “pinecone-client”
- Load Data



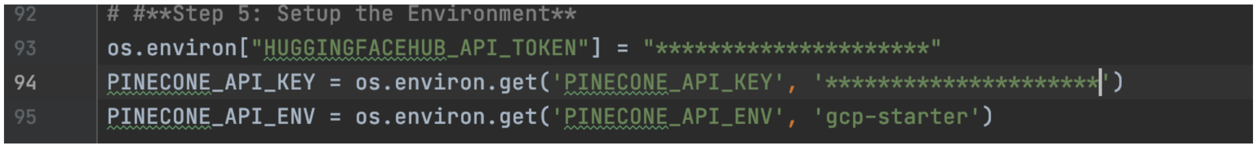
```
# #**Step 3: Load the Data**
loader = PyPDFLoader("content/The-Field-Guide-to-Data-Science.pdf")

data = loader.load()
```

- Split the text into chunks



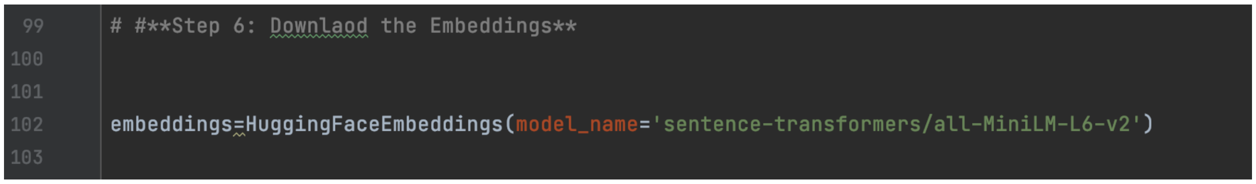
```
# #**Step 4: Split the Text into Chunks**
text_splitter=RecursiveCharacterTextSplitter(chunk_size=500, chunk_overlap=0)

docs=text_splitter.split_documents(data)
```

- Setup the PineCone environment



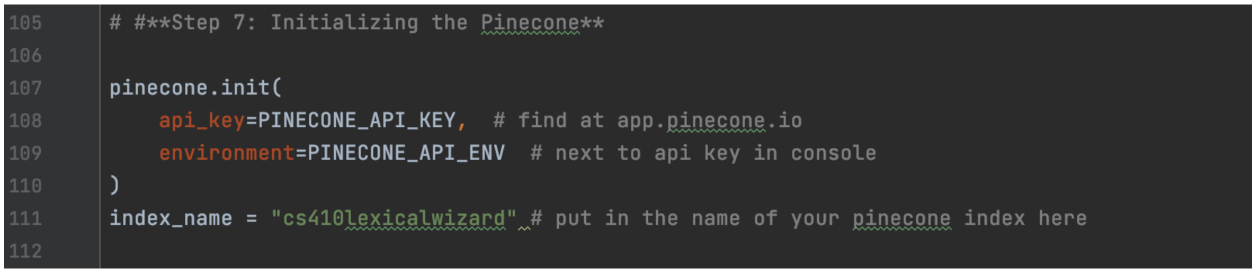
```
# #**Step 5: Setup the Environment**
os.environ["HUGGINGFACEHUB_API_TOKEN"] = "********************"
PINECONE_API_KEY = os.environ.get('PINECONE_API_KEY', '*********************')
PINECONE_API_ENV = os.environ.get('PINECONE_API_ENV', 'gcp-starter')
```

- Download the embeddings



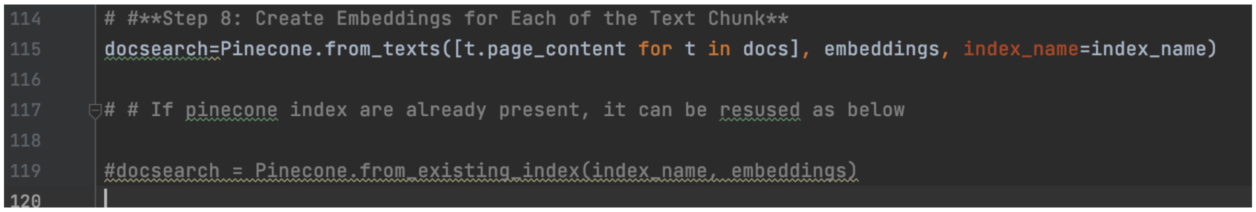
```
# #**Step 6: Downlaod the Embeddings**


embeddings=HuggingFaceEmbeddings(model_name='sentence-transformers/all-MiniLM-L6-v2')

```

- Initialize the Pinecone Vector Database



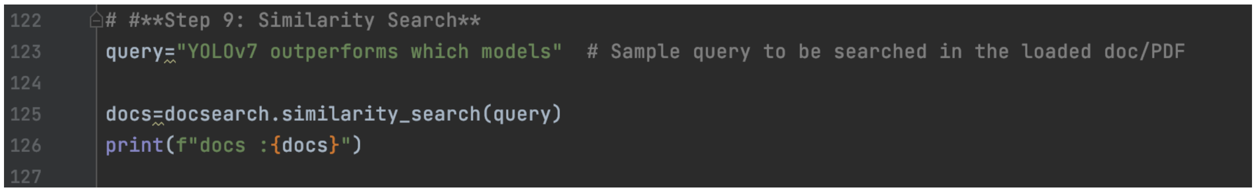
```
# #**Step 7: Initializing the Pinecone**

pinecone.init(
    api_key=PINECONE_API_KEY,  # find at app.pinecone.io
    environment=PINECONE_API_ENV  # next to api key in console
)
index_name = "cs410lexicalwizard" # put in the name of your pinecone index here
```

- Create embeddings for the chunks created above



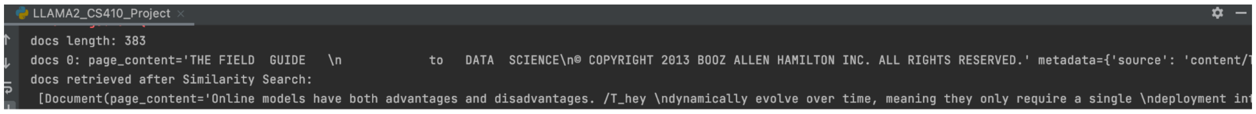
```
# #**Step 8: Create Embeddings for Each of the Text Chunk**
docsearch=Pinecone.from_texts([t.page_content for t in docs], embeddings, index_name=index_name)

# # If pinecone index are already present, it can be resused as below

#docsearch = Pinecone.from_existing_index(index_name, embeddings)

```

- Perform Similarity Search



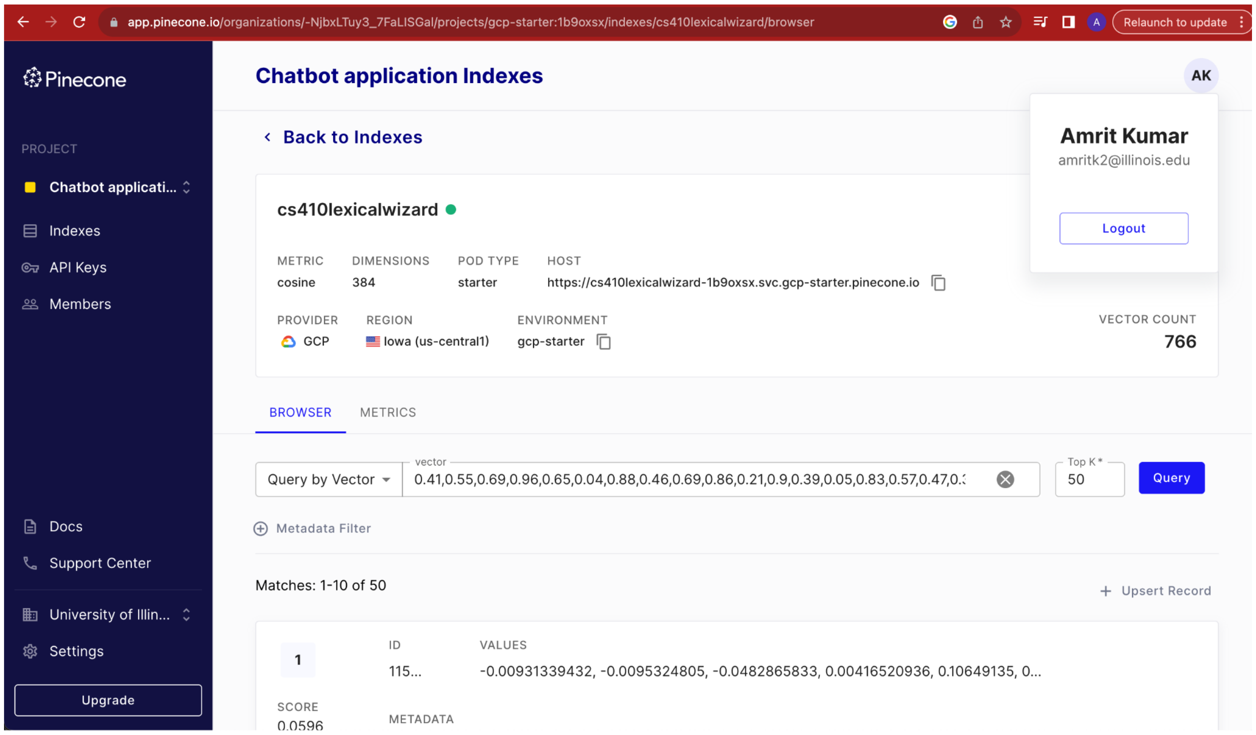
```
# #**Step 9: Similarity Search**
query="YOLOv7 outperforms which models"  # Sample query to be searched in the loaded doc/PDF

docs=docsearch.similarity_search(query)
print(f"docs :{docs}")
```

**Console Output:**



```
docs length: 383
docs 0: page_content='THE FIELD  GUIDE   \n            to   DATA  SCIENCE\n© COPYRIGHT 2013 BOOZ ALLEN HAMILTON INC. ALL RIGHTS RESERVED.' metadata={'source': 'content/T
docs retrieved after Similarity Search:
 [Document(page_content='Online models have both advantages and disadvantages. /T_hey \ndynamically evolve over time, meaning they only require a single \ndeployment int
```

- Verify the index database if embeddings are loaded or not in Pinecone portal



Chatbot application Indexes

Amrit Kumar
amritk2@illinois.edu

cs410lexicalwizard

| METRIC | DIMENSIONS | POD TYPE | HOST |
|---|---|---|---|
| cosine | 384 | starter | https://cs410lexicalwizard-1b9oxsx.svc.gcp-starter.pinecone.io |

| PROVIDER | REGION | ENVIRONMENT | VECTOR COUNT |
|---|---|---|---|
| GCP | Iowa (us-central1) | gcp-starter | 766 |

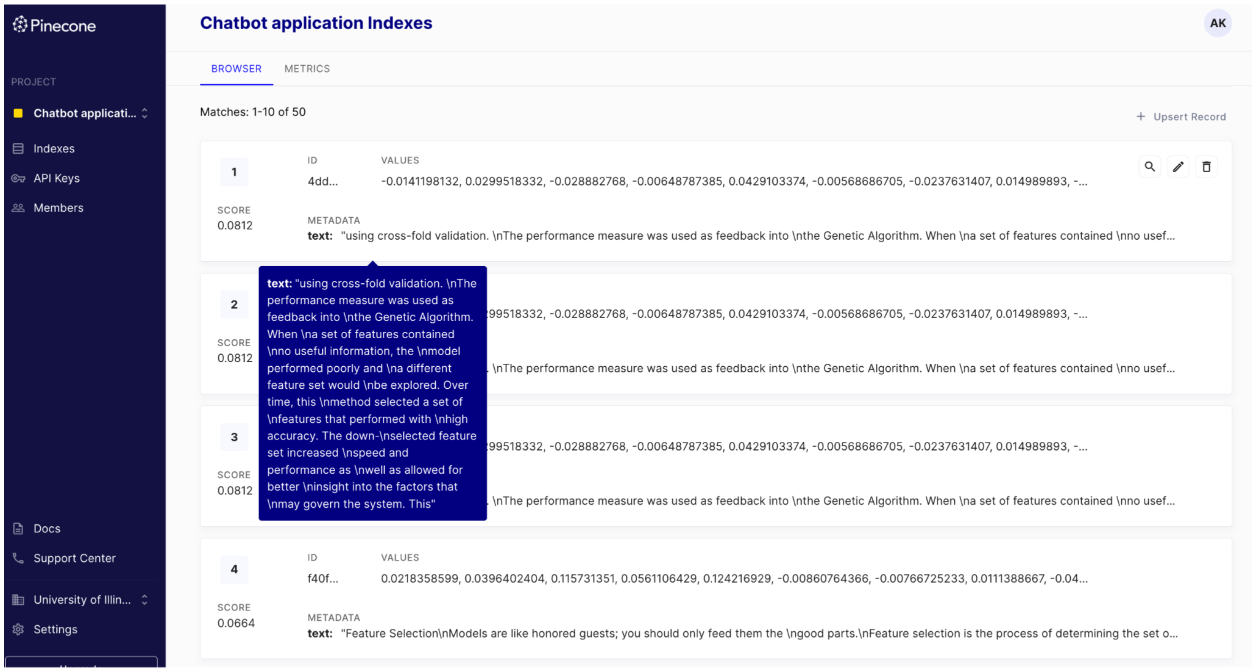
Query by Vector — vector: 0.41,0.55,0.69,0.96,0.65,0.04,0.88,0.46,0.69,0.86,0.21,0.9,0.39,0.05,0.83,0.57,0.47,0.: — Top K*: 50

Matches: 1-10 of 50

| | ID | VALUES |
|---|---|---|
| 1 | 115... | -0.00931339432, -0.0095324805, -0.0482865833, 0.00416520936, 0.10649135, 0... |

SCORE 0.0596 — METADATA

## indexes from the loaded data



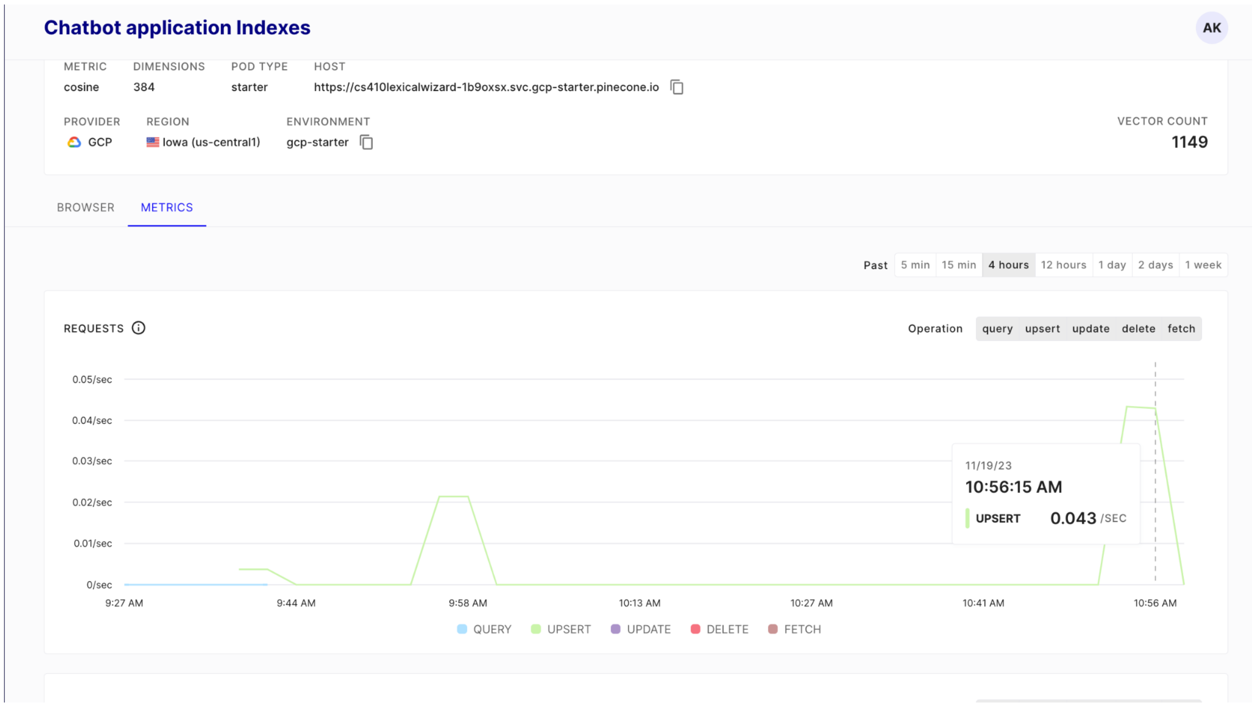
Chatbot application Indexes

Matches: 1-10 of 50

| | ID | VALUES | SCORE | METADATA |
|---|---|---|---|---|
| 1 | 4dd... | -0.0141198132, 0.0299518332, -0.028882768, -0.00648787385, 0.0429103374, -0.00568686705, -0.0237631407, 0.014989893, -... | 0.0812 | text: "using cross-fold validation. \nThe performance measure was used as feedback into \nthe Genetic Algorithm. When \na set of features contained \nno usef... |
| 2 | | 99518332, -0.028882768, -0.00648787385, 0.0429103374, -0.00568686705, -0.0237631407, 0.014989893, -... | 0.0812 | . \nThe performance measure was used as feedback into \nthe Genetic Algorithm. When \na set of features contained \nno usef... |
| 3 | | 99518332, -0.028882768, -0.00648787385, 0.0429103374, -0.00568686705, -0.0237631407, 0.014989893, -... | 0.0812 | . \nThe performance measure was used as feedback into \nthe Genetic Algorithm. When \na set of features contained \nno usef... |
| 4 | f40f... | 0.0218358599, 0.0396402404, 0.115731351, 0.0561106429, 0.124216929, -0.00860764366, -0.00766725233, 0.0111388667, -0.04... | 0.0664 | text: "Feature Selection\nModels are like honored guests; you should only feed them the \ngood parts.\nFeature selection is the process of determining the set o... |

**text:** "using cross-fold validation. \nThe performance measure was used as feedback into \nthe Genetic Algorithm. When \na set of features contained \nno useful information, the \nmodel performed poorly and \na different feature set would \nbe explored. Over time, this \nmethod selected a set of \nfeatures that performed with \nhigh accuracy. The down-\nselected feature set increased \nspeed and performance as \nwell as allowed for better \ninsight into the factors that \nmay govern the system. This"

## Usage Metrics

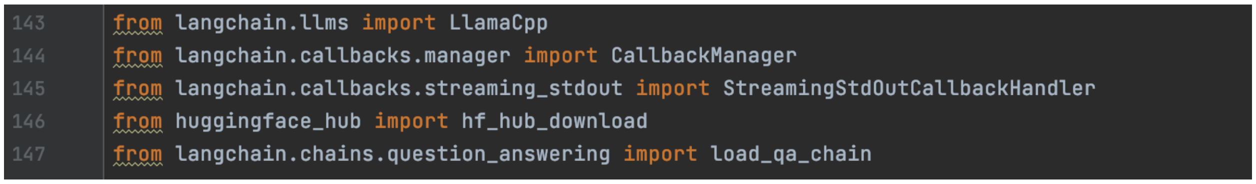

Chatbot application Indexes

| METRIC | DIMENSIONS | POD TYPE | HOST |
|---|---|---|---|
| cosine | 384 | starter | https://cs410lexicalwizard-1b9oxsx.svc.gcp-starter.pinecone.io |

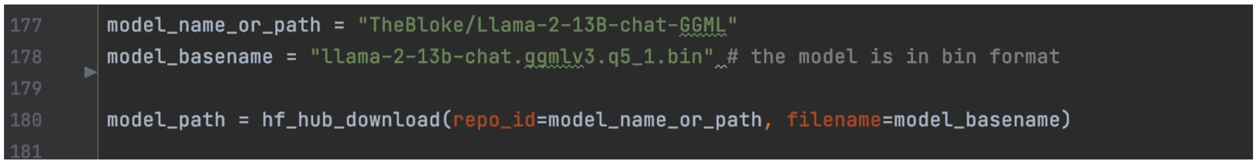
| PROVIDER | REGION | ENVIRONMENT | VECTOR COUNT |
|---|---|---|---|
| GCP | Iowa (us-central1) | gcp-starter | 1149 |

## Information Retrieval and chat conversation using Large Language Model

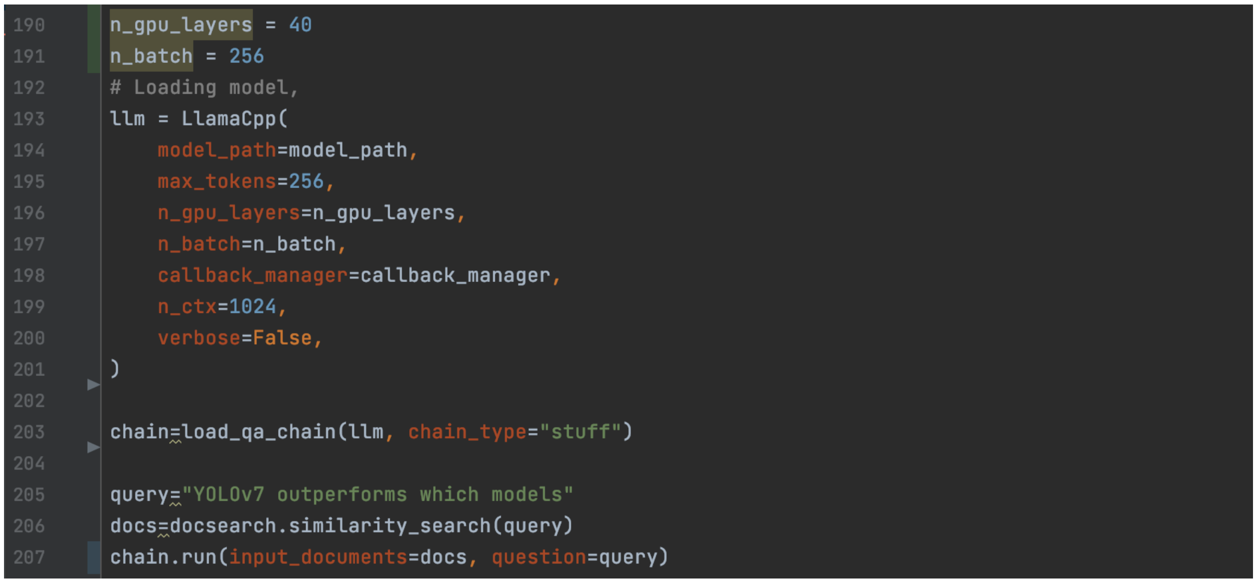
This  module's development is in progress.



```
from langchain.llms import LlamaCpp
from langchain.callbacks.manager import CallbackManager
from langchain.callbacks.streaming_stdout import StreamingStdOutCallbackHandler
from huggingface_hub import hf_hub_download
from langchain.chains.question_answering import load_qa_chain
```

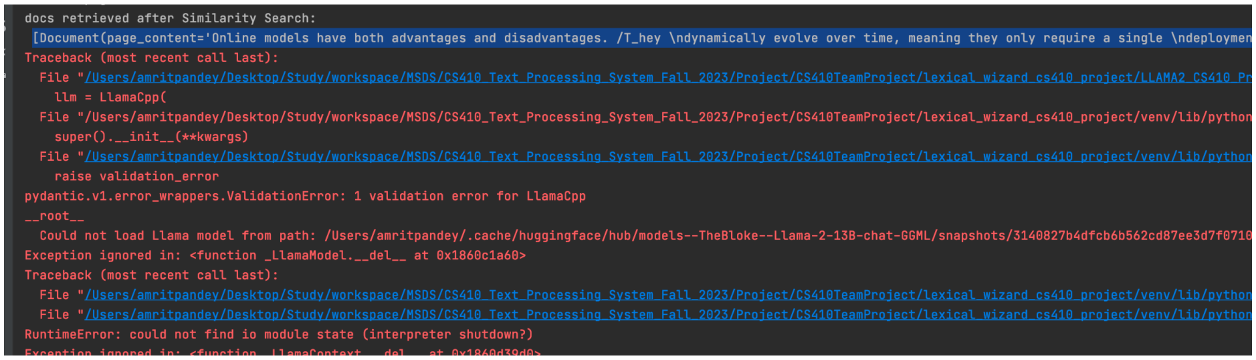


```
model_name_or_path = "TheBloke/Llama-2-13B-chat-GGML"
model_basename = "llama-2-13b-chat.ggmlv3.q5_1.bin" # the model is in bin format

model_path = hf_hub_download(repo_id=model_name_or_path, filename=model_basename)
```



```python
n_gpu_layers = 40
n_batch = 256
# Loading model,
llm = LlamaCpp(
    model_path=model_path,
    max_tokens=256,
    n_gpu_layers=n_gpu_layers,
    n_batch=n_batch,
    callback_manager=callback_manager,
    n_ctx=1024,
    verbose=False,
)

chain=load_qa_chain(llm, chain_type="stuff")

query="YOLOv7 outperforms which models"
docs=docsearch.similarity_search(query)
chain.run(input_documents=docs, question=query)
```

1. Which tasks have been completed?:
    a. Using LLAMA2  implement the Information Retrieval module.
    b. LLAMA2 is one of the most popular open source large language model implementation provided by **META**
    c. Installed LLAMA2 as per the instructions https://ai.meta.com/llama/
    d. Sample module has been implemented , code testing is in progress.
2. Which tasks are pending?
    a. As per the current progress, getting exceptions while loading the LlamaCpp model in local, which we are working on.
    b. We are working on resolving the issue.

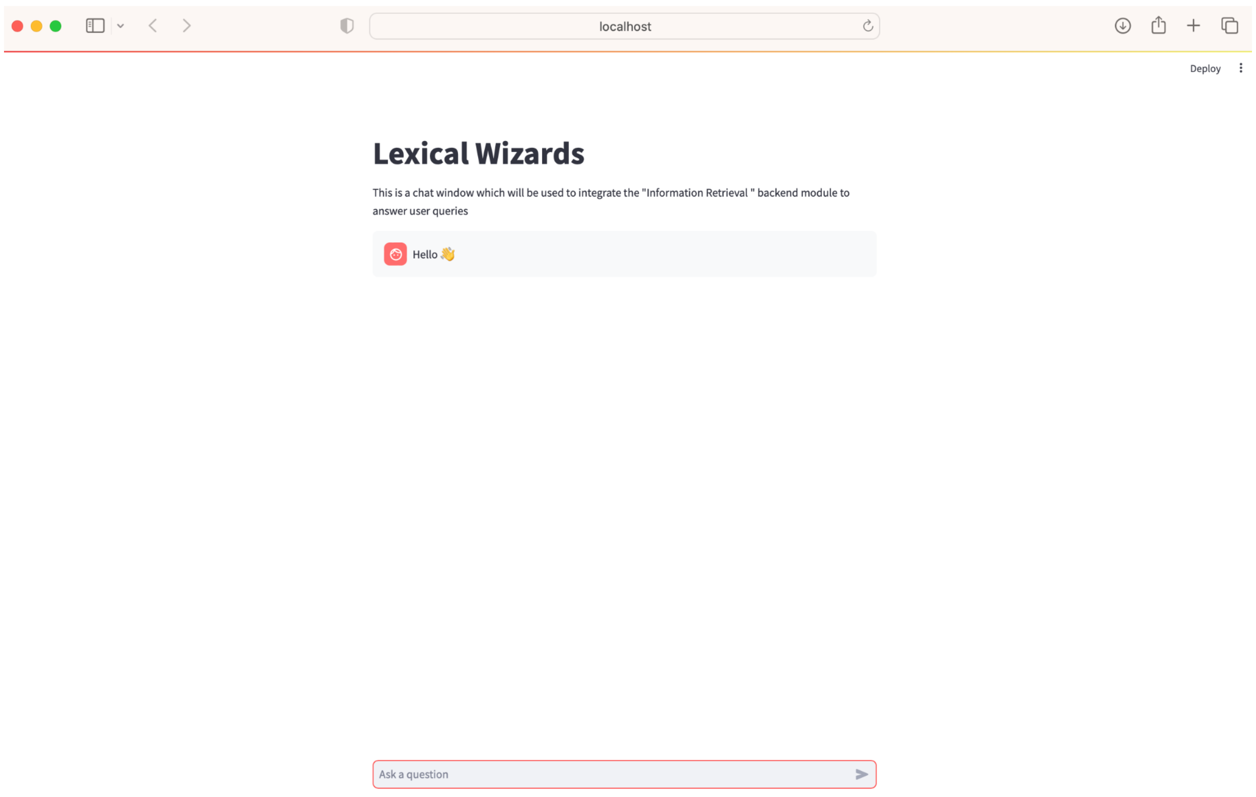


```
docs retrieved after Similarity Search:
 [Document(page_content='Online models have both advantages and disadvantages. /T_hey \ndynamically evolve over time, meaning they only require a single \ndeploymen
Traceback (most recent call last):
  File "/Users/amritpandey/Desktop/Study/workspace/MSDS/CS410_Text_Processing_System_Fall_2023/Project/CS410TeamProject/lexical_wizard_cs410_project/LLAMA2_CS410_Pr
    llm = LlamaCpp(
  File "/Users/amritpandey/Desktop/Study/workspace/MSDS/CS410_Text_Processing_System_Fall_2023/Project/CS410TeamProject/lexical_wizard_cs410_project/venv/lib/python
    super().__init__(**kwargs)
  File "/Users/amritpandey/Desktop/Study/workspace/MSDS/CS410_Text_Processing_System_Fall_2023/Project/CS410TeamProject/lexical_wizard_cs410_project/venv/lib/python
    raise validation_error
pydantic.v1.error_wrappers.ValidationError: 1 validation error for LlamaCpp
__root__
  Could not load Llama model from path: /Users/amritpandey/.cache/huggingface/hub/models--TheBloke--Llama-2-13B-chat-GGML/snapshots/3140827b4dfcb6b562cd87ee3d7f0710
Exception ignored in: <function _LlamaModel.__del__ at 0x1860c1a60>
Traceback (most recent call last):
  File "/Users/amritpandey/Desktop/Study/workspace/MSDS/CS410_Text_Processing_System_Fall_2023/Project/CS410TeamProject/lexical_wizard_cs410_project/venv/lib/python
  File "/Users/amritpandey/Desktop/Study/workspace/MSDS/CS410_Text_Processing_System_Fall_2023/Project/CS410TeamProject/lexical_wizard_cs410_project/venv/lib/python
RuntimeError: could not find io module state (interpreter shutdown?)
Exception ignored in: <function _LlamaContext.__del__ at 0x1860d39d0>
```

3. Any challenges?
    a. No blocker as of now.

## Interactive Chat interface

1. Which tasks have been completed?
    a. We have been able to implement a simple chat window using Streamlit in a python script.
    b. This window will be act as an user interface where an user can post domain specific questions and the application will post relevant answer based on Information retrieval using LLAMA2 (large language model)
    c. Tested with posting simple texts as shown in the screenshot below, the window works as expected.
2. Which tasks are pending?
    a. Integration with the backend Information retrieval system (LLAMA2 powered) API
        i. Specifically sending user input to retrieval system and then displaying the fetched information for user
    b. Conversation context: to remember previous conversations.
3. Any challenges?
    a. No challenges so far, besides potential challenges involving integration.



Lexical Wizards

This is a chat window which will be used to integrate the "Information Retrieval " backend module to answer user queries

Hello 👋

## Summary:

Overall , our progress is good so far. We have been able to finish following items:

- System Design
- Tech stack: Here are few important packages, libraries
    - Large Language Model: LLAMA2
    - DuckDuckGo search: for web scraping
    - Vector Store Database: Pinecone
    - UI Interface: Streamlit
    - Other important libraries/APIs: Hugging Face embeddings, langchain, pyloader etc
- **Proof of Concepts**: We have been able to implement POC to ensure different modules are implementable and narrow down the tech stack system design. Here are the proof of concepts that we have worked on:
    - **Text chunk generation/ embeddings**:
        - Able to implement a template code for loading the PDF , converting into embeddings and implementing a QA module, which answers user queries based on the user query.
        - This was a very simple implementation to prove the viability of the project goal.
    - **Web Scraping**:
        - Able to programmatically scrape information from the web pages using DuckDuckGo search
        - The scrapped information will be embedded in the vector store to enhance the knowledge base and provide rich context for the system to provide answers to user queries in addition to the context created through the PDF load.
    - **Vector Store Database Using Pinecone**:
        - Successfully able to set up the Pinecone vector database to store vector embeddings as indexes for quick retrieval and persistent memory.
        - Successfully implemented the code to perform:
            - saving logic for embedded chunks
            - Ranking logic: Similarity Search
    - **Chat window**:
        - Chatbot user interface for user conversation with the system.
- Pending Items:
    - Proof of concept:
        - Chat conversation LLAMA 2 implementation (IN PROGRESS currently)
    - Integrating different modules together
    - Code cleanup and reviews