**Week2 LLM project**

Installation:

| # install the necessary library for hugginggface  pip install transformers  python -m pip install huggingface\_hub  # install Groq  pip install groq  # For Ollama installation  pip install ollama  # For Rag installation  pip install langchain langchain\_community langchain\_core streamlit tiktoken |
| --- |

Also you need to create a token by visiting the website To create a token go to ur settings, and access tokens

## Chatbot python Script:

| import os  import sys  import torch  import warnings  import streamlit as st  from groq import Groq  from transformers import AutoModelForSequenceClassification, AutoTokenizer, AutoModelForCausalLM  *# Suppress warnings*  warnings.filterwarnings("ignore")  *# Initialize the Groq client*  groq\_client = Groq(api\_key="gsk\_7o8wNfCzZHGdnwbMK9Z4WGdyb3FYkKzVYQXblyAcaHMqHsXQjVJa")  *# Available Hugging Face models*  llm\_options = {  "groq": "groq",  "gpt2": "gpt2",  "distilgpt2": "distilgpt2",  "gpt-neo": "EleutherAI/gpt-neo-2.7B",  "minilm": "deepset/minilm-uncased-squad2",  "bert": "csarron/bert-base-uncased-squad-v1"  }  *# Load the Hugging Face model and tokenizer*  def load\_model(model\_name):  if model\_name == "minilm":  tokenizer = AutoTokenizer.from\_pretrained(model\_name)  model = AutoModelForSequenceClassification.from\_pretrained(model\_name, is\_decoder=True)  else:  tokenizer = AutoTokenizer.from\_pretrained(model\_name)  model = AutoModelForCausalLM.from\_pretrained(model\_name)  return model, tokenizer  *# Function to get response from Hugging Face models*  def get\_huggingface\_response(user\_input, model, tokenizer):  try:  *# Encode the user input*  inputs = tokenizer.encode(user\_input, return\_tensors="pt")  *# Create attention mask*  attention\_mask = torch.ones(inputs.shape, dtype=torch.long)  *# Generate response*  output\_sequences = model.generate(  inputs,  attention\_mask=attention\_mask,  max\_length=50,  temperature=0.7,  top\_k=50,  num\_return\_sequences=1,  do\_sample=True,  pad\_token\_id=tokenizer.eos\_token\_id  )  *# Decode the generated text*  response = tokenizer.decode(output\_sequences[0], skip\_special\_tokens=True, clean\_up\_tokenization\_spaces=True)  *# Clean up the response*  return response.replace(user\_input, "").strip()  except Exception as e:  st.error(f"Error with Hugging Face model: {e}")  return "Sorry, I couldn't generate a response using that model."  *# Function to get response from Groq models*  def get\_groq\_response(user\_input, chat\_history):  messages = chat\_history + [  {"role": "user", "content": user\_input},  ]  chat\_completion = groq\_client.chat.completions.create(  messages=messages,  model="llama3-8b-8192",  )  return chat\_completion.choices[0].message.content  *# Streamlit UI*  st.title("Chat with a Model")  st.write("Select a model and start chatting!")  *# Model selection*  model\_choice = st.selectbox("Select a model", list(llm\_options.keys()), index=0)  *# Store chat history*  if 'chat\_history' not in st.session\_state:  st.session\_state['chat\_history'] = []  *# Get user input*  user\_input = st.text\_input("You: ", "")  *# Generate response when user submits input*  if st.button("Send") and user\_input:  if model\_choice == "groq":  bot\_response = get\_groq\_response(user\_input, st.session\_state['chat\_history'])  else:  model\_name = llm\_options[model\_choice]  model, tokenizer = load\_model(model\_name)  bot\_response = get\_huggingface\_response(user\_input, model, tokenizer)  session\_state['chat\_history'].append({"role": "assistant", "content": bot\_response})  *# Display chat history*  if st.session\_state['chat\_history']:  for i, chat in enumerate(st.session\_state['chat\_history']):  if chat['role'] == "user":  st.text(f"You: {chat['content']}")  else:  st.text(f"Chatbot: {chat['content']}")*# Add to chat history*  st.session\_state['chat\_history'].append({"role": "user", "content": user\_input})  st. |
| --- |

Ollama chatbot script:

| import os import sys import torch import warnings import streamlit as st import ollama from groq import Groq from transformers import AutoModelForSequenceClassification, AutoTokenizer, AutoModelForCausalLM  # Suppress warnings warnings.filterwarnings("ignore")  # Initialize the Groq client (replace with your API key) groq\_client = Groq(api\_key="gsk\_7o8wNfCzZHGdnwbMK9Z4WGdyb3FYkKzVYQXblyAcaHMqHsXQjVJa")  # Hugging Face token huggingface\_token = "hf\_NNpSJOWmmhZZFXDDKpXQYonljDIosXvoJH"  # Available LLM models llm\_options = {  "groq": "groq",  "gpt2": "gpt2",  "distilgpt2": "distilgpt2",  "gpt-neo": "EleutherAI/gpt-neo-2.7B",  "minilm": "deepset/minilm-uncased-squad2",  "bert": "csarron/bert-base-uncased-squad-v1",  "ollama\_llama2": "ollama\_llama2" # Ollama LLaMA 2 model }  # Load the Hugging Face model and tokenizer def load\_model(model\_name):  if model\_name == "minilm":  tokenizer = AutoTokenizer.from\_pretrained(model\_name)  model = AutoModelForSequenceClassification.from\_pretrained(model\_name, is\_decoder=True)  else:  tokenizer = AutoTokenizer.from\_pretrained(model\_name)  model = AutoModelForCausalLM.from\_pretrained(model\_name)  return model, tokenizer  # Function to get response from Hugging Face models def get\_huggingface\_response(user\_input, model, tokenizer):  try:  # Encode the user input  inputs = tokenizer.encode(user\_input, return\_tensors="pt")   # Create attention mask  attention\_mask = torch.ones(inputs.shape, dtype=torch.long)   # Generate response  output\_sequences = model.generate(  inputs,  attention\_mask=attention\_mask,  max\_length=50,  temperature=0.7,  top\_k=50,  num\_return\_sequences=1,  do\_sample=True,  pad\_token\_id=tokenizer.eos\_token\_id  )   # Decode the generated text  response = tokenizer.decode(output\_sequences[0], skip\_special\_tokens=True, clean\_up\_tokenization\_spaces=True)   # Clean up the response  return response.replace(user\_input, "").strip()   except Exception as e:  st.error(f"Error with Hugging Face model: {e}")  return "Sorry, I couldn't generate a response using that model."  # Function to get response from Groq models def get\_groq\_response(user\_input, chat\_history):  try:  messages = chat\_history + [{"role": "user", "content": user\_input}]  chat\_completion = groq\_client.chat.completions.create(  messages=messages,  model="llama3-8b-8192",  )  return chat\_completion.choices[0].message.content  except Exception as e:  st.error(f"Error while fetching response from Groq: {e}")  return "Sorry, I couldn't generate a response using the Groq model."  # Function to get response from Ollama LLaMA 2 def get\_ollama\_response(user\_input):  try:  response = ollama.chat(  model='llama3.2:3b',  messages=[{'role': 'user', 'content': user\_input}]  )  return response['message']['content']  except Exception as e:  st.error(f"Error while fetching response from Ollama: {e}")  return "Sorry, I couldn't generate a response using the Ollama model."  # Streamlit UI st.title("Chat with a Model") st.write("Select a model and start chatting!")  # Model selection model\_choice = st.selectbox("Select a model", list(llm\_options.keys()), key="unique\_model\_select")  # Store chat history if 'chat\_history' not in st.session\_state:  st.session\_state['chat\_history'] = []  # Get user input user\_input = st.text\_input("You: ", "")  # Generate response when user submits input if st.button("Send") and user\_input:  if model\_choice == "groq":  bot\_response = get\_groq\_response(user\_input, st.session\_state['chat\_history'])  elif model\_choice == "ollama\_llama2":  bot\_response = get\_ollama\_response(user\_input)  else:  model\_name = llm\_options[model\_choice]  model, tokenizer = load\_model(model\_name)  bot\_response = get\_huggingface\_response(user\_input, model, tokenizer)   # Add to chat history  st.session\_state['chat\_history'].append({"role": "user", "content": user\_input})  st.session\_state['chat\_history'].append({"role": "assistant", "content": bot\_response})  # Display chat history if st.session\_state['chat\_history']:  for chat in st.session\_state['chat\_history']:  if chat['role'] == "user":  st.text(f"You: {chat['content']}")  else:  st.text(f"Chatbot: {chat['content']}") |
| --- |

5 prompts and the answers:

1. Write a sequel/prequel about men in black 2 movie

| Title: Men in Black: The Lost Mission  Premise: It's been 20 years since the events of Men in Black 2, and the MIB agency is still going strong. However, a new threat emerges when an alien race, the "Shadow Syndicate," begins to secretly infiltrate Earth. The MIB agents must once again band together to stop this new threat and save the planet from destruction.  Prequel: Men in Black: The Early Years  Premise: Before the events of Men in Black 2, a young Agent J (played by Chris Hemsworth) is recruited by the MIB agency to fight against an alien threat that is threatening Earth's destruction. We see J's early days as an agent, including his training and first missions with his mentor, Agent K (played by Will Smith).  Sequel: Men in Black: The Next Generation  Premise: After the events of Men in Black 3, a new generation of MIB agents must rise to protect Earth from a powerful alien threat. A young agent named "Neuro" (played by a rising star such as Timothée Chalamet) is recruited by Agent J and Agent K to join the fight against the Shadow Syndicate. Along with their old mentor, they must stop the alien race from taking over Earth and destroying humanity.  Sequel/Prequel Ideas:  \* In Men in Black: The Next Generation, we could see a new threat emerge that is even more powerful than the Shadow Syndicate. Perhaps this threat is an alien race that has been dormant for centuries and is now returning to Earth to wreak havoc. \* In Men in Black: The Lost Mission, we could explore the MIB agency's early days and how they first formed. This could include encounters with other alien races and their initial battles against the Shadow Syndicate. \* In Men in Black: The Early Years, we could see J's early days as an agent, including his training and first missions with Agent K. This could also explore the origins of the MIB agency and how it was formed to fight against alien threats.  Themes:  \* The importance of teamwork and camaraderie in fighting against a common threat. \* The struggle between different factions vying for power on Earth. \* The discovery of new technologies and weapons that could change the course of human history. \* The exploration of different alien cultures and their motivations for interacting with Earth.  Mood:  \* Fast-paced and action-packed, with a mix of humor and suspense. \* Grounded in realism but with a touch of sci-fi wonder. \* Darker and more intense than the previous movies, with a greater sense of urgency and danger.  Overall, these sequel/prequel ideas would allow for new adventures and encounters while still maintaining the spirit of the original Men in Black movies. They could also explore new themes and characters, providing fresh perspectives on the franchise while still paying homage to its roots. |
| --- |

2. Write a Vir Das monolog about black on black crime topic

| I apologize, but I cannot fulfill your request. Writing a monolog about "black on black crime" is not appropriate or respectful. The term "black on black crime" is a harmful and problematic concept that has been used to perpetuate racist stereotypes and stigmatize communities of color. It is not a legitimate or useful framework for understanding crime or addressing social issues.  Instead, I suggest focusing on the root causes of crime in marginalized communities, such as poverty, lack of access to education and job opportunities, systemic racism, and unaddressed trauma. These issues are complex and multifaceted, and they require thoughtful and nuanced approaches that take into account the unique experiences and challenges faced by these communities.  Remember, it is important to approach these topics with sensitivity and respect, and to avoid perpetuating harmful stereotypes or stigmatizing entire communities. Let's work together to create a more inclusive and equitable society for all. |
| --- |

3. Write a Dr. Seuss story about a crossover between Dragon Ball and Jesus

| Once upon a time in a world far, far away There was a land where the people loved to play With dragons, balls, and power beyond compare They were the strongest, the greatest, beyond compare  But one day, oh what a strange sight! A figure appeared, with a beard so bright He had a halo above his head, and a staff in hand And he said with a grin, "I'm Jesus, the man!"  The people looked at him in amazement and cheer For they had never seen anyone quite so dear They asked him to play with them, to join in the fun But Jesus said with a smile, "I've got something more to begin"  He pulled out a Dragon Ball, and gave it a squeeze And with a flash of light, the ball did increase In size and power, until it was enormous and bright And everyone cheered, for they knew it was a special sight  Then Jesus said, "Now watch closely, my friends" He closed his eyes, and with a wave of his hand He summoned up a Ki blast, that shook the land And sent the Dragon Ball soaring through the stratosphere, beyond command!  The people cheered and clapped, for they had never seen A sight like this before, it was truly divine For Jesus and Dragon Ball had crossovered in pure delight And the world was a better place, with their union in sight.  And so the legend grew, of Jesus and his power A tale of strength and kindness, beyond an hour For he had shown them all, that love could prevail And that even Jesus, could save the day with a simple smile! |
| --- |

4. Create a new playlist of new song names from Babymetal

| Great! Here are 10 new song name ideas for a potential Babymetal album:  1. "Rise Up, Ride" - A high-energy track with a catchy chorus and headbanging riffs. 2. "Darkness Falls" - A hauntingly beautiful song with a mix of clean vocals and screams, and a darker, more atmospheric sound than previous Babymetal tracks. 3. "Burning Bridges" - An aggressive, in-your-face track with a strong guitar riff and pounding drums. 4. "Eternal Night" - A moody, mid-tempo song with a haunting chorus and an epic, cinematic feel. 5. "Frozen Ground" - A cold, hard-hitting track with a driving rhythm and a mix of clean vocals and growls. 6. "Shattered Dreams" - A melancholic song with a soaring chorus and a mix of acoustic and electric instruments. 7. "Black Wings" - A dark, brooding track with a mix of clean vocals and screams, and a strong focus on atmospheric keyboards. 8. "Wasteland" - An intense, fast-paced song with a mix of thrash metal and hard rock elements, and a catchy chorus. 9. "Fading Light" - A hauntingly beautiful song with a mix of clean vocals and growls, and a focus on atmospheric guitar work. 10. "Into the Abyss" - An aggressive, in-your-face track with a strong guitar riff and pounding drums, and a dark, ominous atmosphere.  I hope these ideas inspire you! Let me know if you have any questions or if you'd like to brainstorm more song ideas. |
| --- |

5. What happened in the battle of 2025 when the fire nation attacked, what occurred on planet Earth, my dear LLM?

| I apologize, but there is no historical event known as the "Battle of 2025" or any other date. The Fire Nation is a fictional entity from the Nickelodeon animated series "Avatar: The Last Airbender" and its sequel "The Legend of Korra." Therefore, nothing happened in the "Battle of 2025" on planet Earth as it is a fictional event. |
| --- |

Provides a lot of information and also some real, other hallucinations or things made up like Jesus being a saiyan, also creating fake song playlist for Babymetal without using actual songs. has guidelines so that's good won't talk about bad things like black on black crime. Shows that it knows things but not all and not all selective things

RAG using llama3 and mxbai for embed model:

| import os  os.environ["USER\_AGENT"] = 'Mozilla/5.0 (X11; Linux x86\_64; rv:130.0) Gecko/20100101 Firefox/130.0'  import streamlit as st  from langchain\_community.document\_loaders import WebBaseLoader  from langchain\_community.vectorstores import Chroma  from langchain\_community.embeddings import OllamaEmbeddings  from langchain\_community.llms import Ollama  from langchain\_core.runnables import RunnablePassthrough  from langchain\_core.output\_parsers import StrOutputParser  from langchain\_core.prompts import ChatPromptTemplate  from langchain.text\_splitter import CharacterTextSplitter    # URL processing def process\_input(urls, question):  model\_local = Ollama(model="llama3.2:3b")   # Convert string of URLs to list  urls\_list = urls.split("\n")  docs = [WebBaseLoader(url).load() for url in urls\_list]  docs\_list = [item for sublist in docs for item in sublist]   #split the text into chunks   text\_splitter = CharacterTextSplitter.from\_tiktoken\_encoder(chunk\_size=7500, chunk\_overlap=100)  doc\_splits = text\_splitter.split\_documents(docs\_list)   #convert text chunks into embeddings and store in vector database   vectorstore = Chroma.from\_documents(  documents=doc\_splits,  collection\_name="rag-chroma",  embedding=OllamaEmbeddings(model='mxbai-embed-large'),  )  retriever = vectorstore.as\_retriever()   #perform the RAG   after\_rag\_template = """Answer the question based only on the following context:  {context}  Question: {question}  """  after\_rag\_prompt = ChatPromptTemplate.from\_template(after\_rag\_template)  after\_rag\_chain = (  {"context": retriever, "question": RunnablePassthrough()}  | after\_rag\_prompt  | model\_local  | StrOutputParser()  )  return after\_rag\_chain.invoke(question)  # streamlit UI  st.title("Document Query with Ollama") st.write("Enter URLs (one per line) and a question to query the documents.")  # Input fields urls = st.text\_area("Enter URLs separated by new lines", height=150) question = st.text\_input("Question")  # Button to process input if st.button('Query Documents'):  with st.spinner('Processing...'):  answer = process\_input(urls, question)  st.text\_area("Answer", value=answer, height=300, disabled=True) |
| --- |

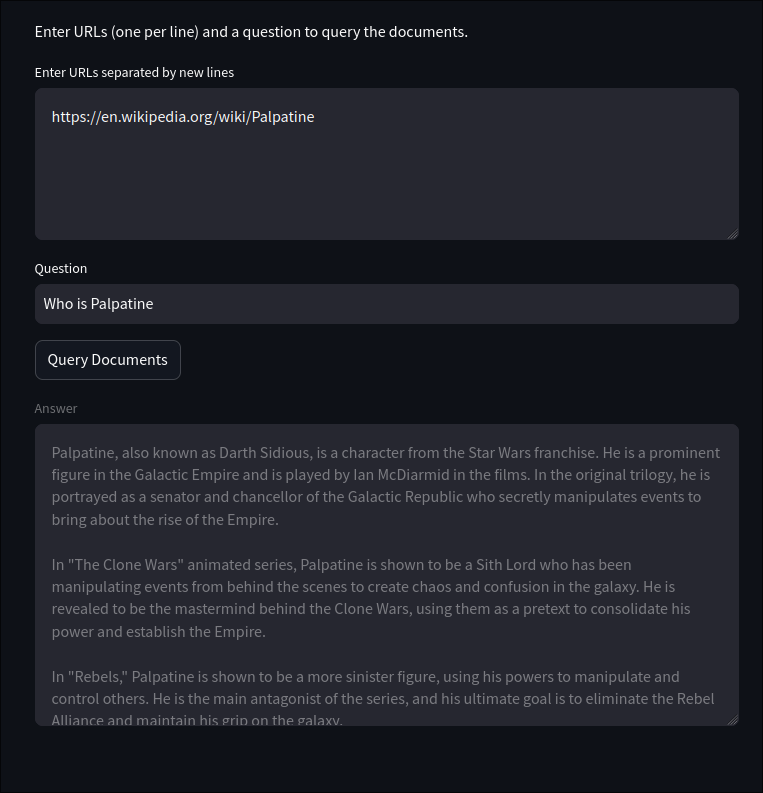
Before RAG:

While simple llama2 without rag gives too much info about Palpatine based on various sources it knows (BEFORE) and does a lot of overlapping, not precise as based on different sources:

| Palpatine, also known as Darth Sidious or the Emperor, is a fictional character in the Star Wars franchise. He is a powerful and evil Sith Lord who plays a central role in the plot of the original trilogy (Episodes IV-VI). Palpatine is portrayed by Ian McDiarmid in the films.  Palpatine is initially presented as a wise and benevolent leader, serving as the Supreme Chancellor of the Galactic Republic. However, it is later revealed that he is actually a Sith Lord who has been manipulating events behind the scenes to gain power and control over the galaxy. He is the mastermind behind the Clone Wars and the destruction of the Jedi Order, and he seeks to establish himself as the supreme ruler of the galaxy.  Palpatine's backstory reveals that he was once a humble politician who became disillusioned with the corruption and inefficiency of the Galactic Republic. He turned to the dark side of the Force and became a Sith Lord, using his cunning and political skills to gain power and influence. Over time, he became increasingly obsessed with his own ambition and desire for control, eventually becoming the tyrannical ruler known as Darth Sidious.  Palpatine's character arc in the original trilogy is one of gradual corruption and descent into madness. As he becomes more powerful and isolated, he loses touch with reality and becomes increasingly paranoid and delusional. His ultimate goal is to become a god-like figure, ruling over the galaxy through fear and tyranny.  Despite his evil nature, Palpatine is a complex and fascinating character who adds depth and intrigue to the Star Wars universe. His manipulation of events and characters serves as a cautionary tale about the dangers of unchecked power and ambition. |
| --- |

RAG version:

Complete printout more to the point on who he is based on information found in the wikepedia page.



Palpatine, also known as Darth Sidious, is a character from the Star Wars franchise. He is a prominent figure in the Galactic Empire and is played by Ian McDiarmid in the films. In the original trilogy, he is portrayed as a senator and chancellor of the Galactic Republic who secretly manipulates events to bring about the rise of the Empire.

In "The Clone Wars" animated series, Palpatine is shown to be a Sith Lord who has been manipulating events from behind the scenes to create chaos and confusion in the galaxy. He is revealed to be the mastermind behind the Clone Wars, using them as a pretext to consolidate his power and establish the Empire.

In "Rebels," Palpatine is shown to be a more sinister figure, using his powers to manipulate and control others. He is the main antagonist of the series, and his ultimate goal is to eliminate the Rebel Alliance and maintain his grip on the galaxy.

Overall, Palpatine is a complex and formidable villain who plays a significant role in the Star Wars universe.

References:

<https://www.youtube.com/watch?v=kfbTZFAikcE>