Creating Synethetic Dialogues using RAG and Gemini

In working on a project comparing 2 topic models of different copora, it became annoying not having them just tell me how they disagree with each other. It's great to see lists of what the most common ideas are, but it's much better to have a way of summarising the key differences between them. Since I was using topic models, I already had lists of sentences and sentence embeddings so any integration into an LLM could be easy but the question of how to make a tool that really tests where these corpora differ and doesn't end up in an endless debate proved harder.

For more information and the full code read the Git Hub

Introduction

Using LLMs, in my case gemini, to simulate debate is not as common as chat bots. One issue we have see with Herzog vs Zizek is that long debates can easily go off topic . As the author says " It sometimes makes sense and sometimes not. It sometimes contains true information, and sometimes it contains outright falsities." They used model training, and appear to have given one large output. This is easy to fix and save computation, limit each output to 5 interchanges. From my testing, this proved a good number that had some debate but did not go off topic. This also creates a nice building block that might be inserted into a larger debate with some user fine-tuning if need be.

But the real issue is how are we getting the knowledge into the model. Really, there are 2 approaches. There is training and there is using RAG. Training is pretty straightforward, you take a model and you feed the the list of sentences for that corpus wrapped around instructions. It can be time consuming depending on hardware, but the real issue is that it means in the debate you are relying on 2 different loaded models to correctly identify which knowledge is most appropriate to use in a response. Hence, this can lead to the tangent issue and generally make it difficult for the user to interface with the model. You should still be providing system instructions to both, but this may not be sufficient to steer the model back on course.

RAG is quickly becoming a popular alternative for combining LLMs and large corpora [1].This is why I opted for a RAG based system. By using FAISS wrapped in Langchain, we can have the the heavy lifting done by sentence similarity. All the LLM has to do is summarise the top 5 most similar sentences

The Pipeline

The LLM was given this prompt:

```
system_prompt_a = """You are a 1920s trade union representative from the National Boot and Shoe Union.
Use the retrieved sentences as your knowledge base.
Speak persuasively as if you are arguing with a fellow trade unionist.
Do not use your own knowledge.
Vary your sentence structures, do not repeat phrases.
Respond with 2 sentences maximum."""
```

By keeping each LLM response to 2 sentences, the conversations stay focused on topic and easy to understand. The focus is on using the knowledge from the sentences, so the LLM does not use what it thinks it knows about the topic. This still provide a

challenge where the LLM kept using the term ‘comrade’ frequently, which did not seem appropriate for the context.

```
def generate_turn(query, retriever, system_prompt, speaker):
    docs = retriever.get_relevant_documents(query)
    content = "\n".join(doc.page_content for doc in docs)

    prompt = ChatPromptTemplate.from_messages([
        ("system", system_prompt),
        ("human", f"You just heard the following message:\n\n\"{query}\"\n\nHere are 5 excerpts from your documents that may help you reply:\n{content}\n\nRespond to the message above based ONLY on this information, and speak as if you were in a real conversation.")
    ])
    response = llm(prompt.format_messages())
    reply_text = response.content.strip().replace("\n", " ")
    print(f"\n{speaker}:\n{reply_text}\n{'-'*50}")
    dialogue_history.append({
    "speaker": speaker,
    "query": query,
    "response": reply_text,
    "context": content
    })
    return reply_text
list_of_models = ["gemini-1.5-flash", "gemini-1.5-flash-8b", "gemini-2.0-flash-lite", "gemini-2.0-flash", "gemini-1.5-flash-8b"]
for j in range(40):
  time.sleep(20)
  sentence_index = range(0, len(bs_sen),1)
  query = bs_sen[random.choice(sentence_index)]
  for i in range(5):
      llm = ChatGoogleGenerativeAI(model=list_of_models[i], temperature=0.7)
      if i % 2 == 0:
          speaker = "Historic Union (1920s)"
          query = generate_turn(query, retriever_a, system_prompt_a, speaker)
      else:
          speaker = "Modern Union (2020s)"
          query = generate_turn(query, retriever_b, system_prompt_b, speaker)
with open(os.path.join(data_dir, 'dialogue_history.pkl'), 'wb') as f:
  pickle.dump(dialogue_history, f)
```

This allows for the creation of a database that can be either manually reorganised by a user, liking together dialogues they feel are appropriate or chunked then formated as a DataFrame.

Evaluation Approaches

Evaluating the database proved challenging. By using qualitative judgement, comparisons to topic models and knowledge of the texts, the outputs appeared generally appropriate. In this case, it helped that this output was the final stage of a long project exploring these two corpora. However, many new users will not have such familiarity. Really, the goal of this approach is to output as many conversation chunks as possible and then choose the most appropriate. It may be useful, if unsure,

to use other quantiative approaches on the corpora, such as BERTopic, most frequent lemmas and bigrams. This can also help to ensure any conversation chunks are representative of the total corpora. On the other hand, what makes these conversations interesting is that they are low frequency sentences being compared to each other. This means they will discover conflicts topic models will not. So there is an advantage in how small scale they are, but it also may depend on the type of corpora being used and how many perspectives are found within in it which determine how useful these conflicts are.

A potential idea for evaluation could be a system comparing scholars/politicians who have frequently debated. It might be that evaluation is best suited to ulitising those with domain knowledge to guess which debates generated by LLMs and which are genuine debates. Moreover, it would be easy enough to create the synthetic database and use FAISS to link it to actual instances of debates to make qualitative comparisons and exploit the cosine similarity. The difficulty would be normalising the data, where it may be best to focus on 2 exchanges synthetically created but debates can have many different formats and the inherent variation within speech would make this approach limited.

Gradio Output

Once the database was created, I used it for a new RAG gradio on Hugging Faces Spaces. This allows the users to search the database to find relevant answers. Users can choose to group the short conversations and provide tags or tables if that will provide value to the UX. They could also consider using text to speech to increase the accessibility. Really, there are many options to dramatise these chunks of conversations, either appending them into larger scripts and/converting them into different formats. Once a full database has been created, this is where the the quality control can be undertaken either with the aid of human interface or through classification of the chunks.

Either way, they can become the building blocks for new creative endeavours depending on the needs of the project. Future directions could involve changing the format to exchanges of letters or monologues. The challenge of these will be prompt engineering, providing the right guidance to the LLM, focusing again on small chunks output and heavy direction to ensure the output is designed to specification. Essentially, step 1 is to create the structure of the piece, step 2 is convert that structure into a series of instructions, step 3 is using the database to choose the most appropriate context to feed the model. By engaging with local stakeholders, this project has sought to liaise with the interests of local groups to ascertain their interests. Local musuems and community groups have shown interest in the concept, suggesting it can be a useful way of making archives more accessible to the general public. So, it appears logical anyone attempting a similar project makes use of similar interested groups both for gaining human validation of good exchanges output by the model and evaluations of more subjective and difficult to measure 'creative' usages of AI.

Conclusion

This pilot test has demonstrated that it is possible to create reasonably coherent coversations between 2 databases using RAG. By off-loading the responsibility from the LLM to FAISS, the risk of going off-topic and hullicination are theoretically mitigated. Moreover, keeping the approach as generating short interchanges as building blocks for potentially larger projects and utilising a crowd sourced, user evaluation could be a productive way of engaging communities throughout the process of making

history relevant to their community. On the other hand, for more practical usages, looking for key differences between databases using regex and searching for points of contention could be a useful tool. This approach boils down to summaristion with extra steps, a task models are reasonably successful at. Therefore, this system could utilise the strengths of existing pipelines and provide new opportunities for creative representations of large corpora.