AshBot: The Explainer

Introduction

AshBot is an AI-powered chatbot that can answer questions about Ashesi University. But how does it work? Let's break down the process into simple steps and explore the magic behind AshBot's intelligence.

1. Gathering Knowledge: The PDF Collection

Imagine you have a stack of books about Ashesi University. These books contain all sorts of information - from course details to campus life. In our digital world, these "books" are PDF documents. AshBot starts its journey by "reading" these PDFs.

```python
folder_path = './ashbot_files/'
loader = PyPDFDirectoryLoader(folder_path)
documents = loader.load()
```

This code is like telling AshBot, "Hey, all your books are in this folder. Go read them!"

2. Breaking Down the Information: Chunking

Now, imagine if you had to memorize entire books at once. Pretty tough, right? AshBot faces a similar challenge. So, instead of dealing with whole documents, it breaks them down into smaller, manageable pieces called "chunks."

```python
text_splitter = RecursiveCharacterTextSplitter(chunk_size=500, chunk_overlap=100)
all_splits = text_splitter.split_documents(documents)
```

This is like taking those books and dividing them into small, easy-to-read paragraphs. The `chunk_size=500` means each "paragraph" is about 500 characters long, and

`chunk_overlap=100` means each paragraph shares a bit of information with the next one, ensuring no information is lost between chunks.

3. Understanding the Essence: Embeddings

Here's where things get interesting. AshBot doesn't just remember the words; it tries to understand the meaning behind them. It does this by turning each chunk of text into a list of numbers called an "embedding."

```python
embedding_model_name = "sentence-transformers/all-mpnet-base-v2"
embeddings = HuggingFaceEmbeddings(model_name=embedding_model_name,
model_kwargs=model_kwargs)
```

Think of embeddings like this: if you had to describe a cat, you might use numbers to represent its features (4 for legs, 2 for ears, 1 for tail, etc.). AshBot does something similar with text, capturing the essence of each chunk in numbers.

4. Organizing Knowledge: Vector Store

Now that AshBot has all these chunks and their number representations (embeddings), it needs to organize them. This is where the "vector store" comes in.

```python
vectordb = Chroma.from_documents(documents=all_splits, embedding=embeddings,
persist_directory="chroma_db")
```

Imagine a giant library where books are arranged not by author or title, but by their content. Similar books are placed close together. This is what the vector store does with our text chunks.

5. The Brain of AshBot: The Language Model

While the vector store is like AshBot's memory, the language model is its brain. We use a model called Zephyr 7B Alpha, which is incredibly good at understanding and generating human-like text.

```python
model_name = "anakin87/zephyr-7b-alpha-sharded"
model = load_quantized_model(model_name)
tokenizer = initialize_tokenizer(model_name)
```

Loading this model is like giving AshBot a powerful brain that understands language nuances.

6. Preparing for Conversation: The Pipeline

With its brain ready, AshBot needs a way to process information and generate responses. This is what the pipeline does.

```python
pipe = pipeline(
    "text-generation",
    model=model,
    tokenizer=tokenizer,
    # ... other parameters ...
)
```

Think of this pipeline as AshBot's thought process. It takes in information and figures out how to respond.

7. Putting It All Together: The Conversation Chain

Now we have all the pieces: the organized knowledge (vector store) and the thinking process (language model and pipeline). The conversation chain brings these together.

```python
qa_chain = ConversationalRetrievalChain.from_llm(
    llm=llm,
    retriever=retriever,
    memory=memory,
    # ... other parameters ...
)
```

This chain does several things:
1. It takes your question.
2. It looks through the organized knowledge to find relevant information.
3. It uses the language model to understand the question and formulate a response.
4. It keeps track of the conversation, so it can provide context-aware answers.

8. The User Interface: Gradio

Finally, we need a way for you to interact with AshBot. This is where Gradio comes in.

```python
with gr.Blocks() as demo:
    chatbot = gr.Chatbot(label='Chat with Ashesi Bot', ...)
    msg = gr.Textbox(label="Type your message here...")
    # ... other UI elements ...
```

Gradio creates a simple chat interface where you can type questions and see AshBot's responses.

Conclusion: The Complete Picture

So, when you ask AshBot a question, here's what happens:

1. Your question goes through the conversation chain.

2. The chain searches the vector store for relevant information.
3. It retrieves the most similar chunks of text to your question.
4. The language model uses these chunks and its understanding of language to formulate a response.
5. The response is displayed to you through the Gradio interface.

And voilà! You've just had a conversation with a bot that can access and understand a vast amount of information about Ashesi University, all starting from a collection of PDF documents.

This entire process happens in seconds, giving you the experience of chatting with an knowledgeable assistant about Ashesi University!