Assignment 2 - Natural Language Processing: LLM-Powered Ahkam Chatbot

Deep Learning – EE569

Talal Malek Badi – 2200208609
Badr Shehim – 2200208256
Ataher Saleh – 2200208085
Supervisor: Dr. Nuri Benbarka

February 16, 2025

Contents

List of Figures

1 Introduction

In this assignment, an LLM-powered chatbot for (Ahkam Sharia) الأحكام الشرعية will be built. The chatbot will answer questions related to العبادات (Ibadat). It utilizes a large language model (LLM) in conjunction with a vector database to efficiently retrieve and generate responses to user queries. The chatbot follows a Retrieval-Augmented Generation (RAG) approach by first retrieving relevant information from the database and then using the LLM to structure the response.

2 Task 1: Collect Relevant Data

2.1 Task Description

The first task was to collect relevant data about العبادات (Ibadat) from Aldorar's website and then save it to a JSON file.

2.2 Procedure

After visiting Aldorar website and analyzing the structure of the HTML and the required data, BeautifulSoup was used for web scraping.

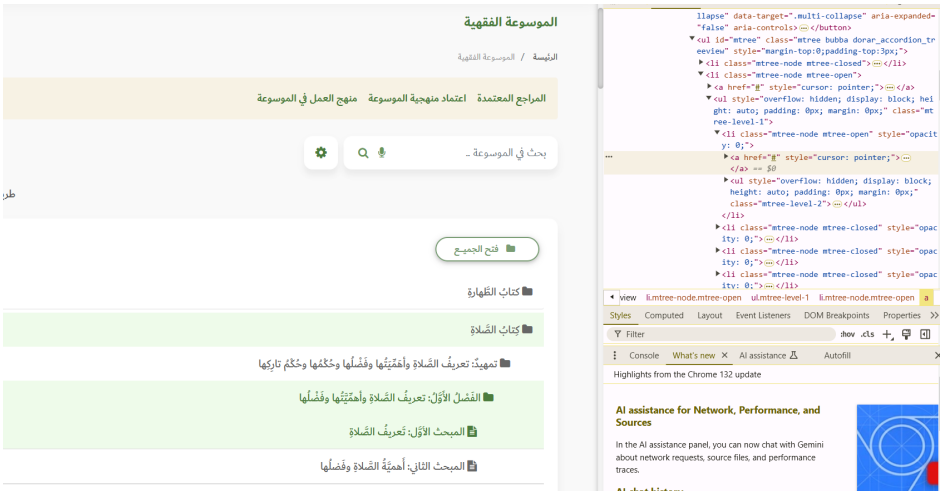

Figure 1: Analyzing the stucture of the website before web scraping

The collected data was processed and structured as a hierarchical tree, and diacritical marks (علامات التشكيل) were removed for improved retrieval (as discussed further in the database generation task).

2.3 Results

The data was successfully collected and saved to a JSON file. The main topics selected include:

- كتاب الطهارة
- كتاب الصلاة
- كتاب الصوم

- كتاب الحج
- كتاب الزكاة

3 Task 2: Create an Evaluation Dataset

3.1 Task Description

The goal of this task is to create a dataset of questions and answers to evaluate the chatbot.

3.2 Procedure

The dataset was divided into three parts: one for generated questions and answers from the collected data, one from Dar Aleftaa's website, and another generated from collected questions from the website. All generation was performed using an LLM (gpt 40-mini), with diacritical marks removed to maintain consistency with the collected data.

3.3 Results

The dataset was successfully created and saved as JSON files. The removal of diacritical marks ensured consistency with the collected dataset.

4 Task 3: Create Proper Prompts

4.1 Task Description

Design prompts to guide the chatbot's behavior when interacting with users.

4.2 Procedure

Prompts were written in a JSON file. A system prompt was designed to control the behavior of the chatbot ensuring that for sensitive topics (Ahkam Sharia), if no relevant data is found, the bot responds with uncertainty. Additionally, an evaluation prompt was designed to guide the grading of chatbot responses.

4.3 Results

The prompts were successfully created and saved to JSON files (further details are discussed in the evaluation step).

5 Task 4: Embed Data into a Vector Database

5.1 Task Description

Convert the collected data into embeddings and store them in a vector database for efficient retrieval.

5.2 Procedure

A vector database was created for a sample of the collected data. After successfully generating embeddings for the sample, tests were conducted by querying with various prompts. It was observed that diacritical marks adversely affected retrieval performance and increased token usage. To address this, diacritical marks were removed from the dataset, questions, and prompts. This approach—similar to lowercasing text for English—resulted in improved retrieval performance and more efficient token usage. Additionally, due to the maximum token limits during embedding, the data was batched by كتب (chapters) and further divided to chunks.

5.3 Results

The dataset was successfully embedded into the vector database using OpenAI's `text-embedding-ada-002-v2` model. Retrieval tests showed satisfactory results. (An example of the improvement is available in the repository: GitHub Repository.)

6 Task 5: Integrate LangChain

6.1 Task Description

Integrate LangChain to build the chatbot application by creating a retrieval-based pipeline that:

1. Queries the vector database for relevant information.
2. Uses an LLM (e.g., OpenAI's GPT) to generate answers based on the retrieved data.

6.2 Procedure

The integration was implemented in `app.py`. Key steps included:

- Initializing the vector store using the embeddings generated earlier.
- Setting up a retrieval chain using LangChain's `RetrievalQA` module.
- Combining the retrieval results with the LLM to generate responses.

6.3 Results

The LangChain integration allowed the chatbot to effectively retrieve contextually relevant data from the vector database and generate coherent answers using the LLM. The full implementation can be found in `chatbot.py` in the repository.

7 Task 6: Build a Web Interface

7.1 Task Description

Create a simple web interface for the chatbot using Gradio that allows users to:

- Interact with the chatbot via a browser.
- View the query history during the session.

7.2 Procedure

The web interface was implemented in `app.py` using Gradio. Key steps included:

- Setting up a Gradio interface that calls the chatbot's query function.
- Maintaining a history of queries and responses to display during the session.

7.3 Results

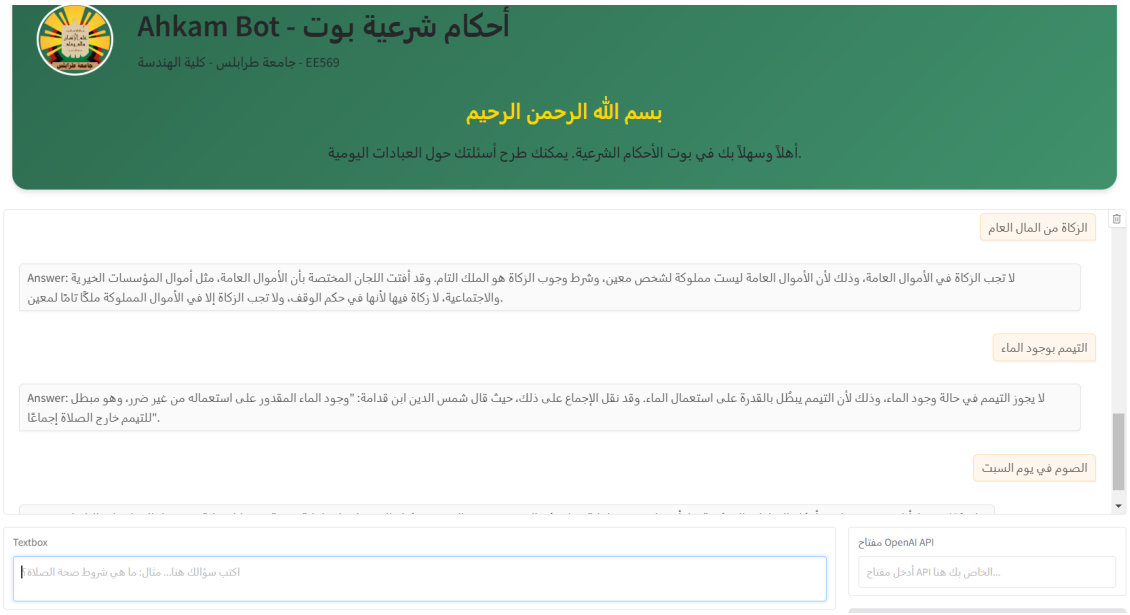

Figure 2: Web interface for the chatbot

The Gradio interface successfully provided a user-friendly way to interact with the chatbot. Users can input queries, receive answers, and view the history of their interactions. The full implementation is available in `app.py` in the repository.

8 Task 7: Evaluate Your Chatbot

8.1 Task Description

The goal of this task is to evaluate the chatbot's performance using the evaluation dataset created in Task 2. The evaluation focuses on:

- **Accuracy:** The percentage of correct answers.
- **Response Quality:** The relevance and completeness of the responses, rated on a scale of 1 to 5.
- **Error Handling:** The chatbot's ability to gracefully handle ambiguous or out-of-scope queries.

Notably, when the chatbot responds with "I don't know the answer," this response is counted as correct, as it reflects proper handling of uncertainty.

8.2 Procedure

The evaluation was performed using a dedicated evaluation script (`evaluation.py`) that:

1. Loads the evaluation dataset from a Weights & Biases artifact.
2. Generates chatbot responses for each query using the Conversational Retrieval Chain.
3. Compares the generated responses against the expected answers using LangChain's QAEvalChain with a custom evaluation prompt.
4. Logs the metrics (accuracy, response quality, and error handling) and writes the results to `evaluation_report.txt`.

A simplified code snippet from `evaluation.py` is shown below:

```
# Load evaluation dataset
eval_dataset = load_eval_dataset(config)

# Generate responses for all questions
eval_dataset = generate_answers(eval_dataset, qa_chain)

# Evaluate the generated responses
eval_dataset = evaluate_answers(eval_dataset, config)
```

8.3 Results and Observations

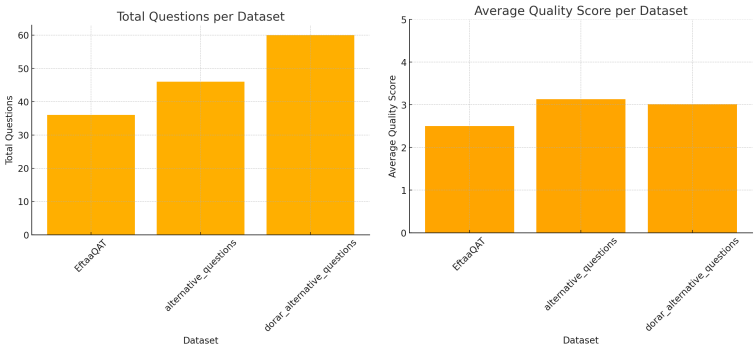

Figure 3: Enter Caption

The evaluation metrics obtained (summarized in `evaluation_results.json`) are as follows:

- **EftaaQAT:** 36 questions, 2 correct answers, 10 incorrect answers, and 23 responses classified as acceptable unknown.
- **EftaaQAT Alternative Questions:** 46 questions, 7 correct answers, 17 incorrect answers, and 22 acceptable unknown responses.
- **Dorar Alternative Questions:** 60 questions, 31 correct answers, 22 incorrect answers, and 7 acceptable unknown responses.

```
"EftaaQAT": {
    "total_questions": 36,
    "correct_answers": 2,
    "incorrect_answers": 10,
    "acceptable_unknown": 23,
    "average_quality_score": 2.5
},
"alternative_questions": {
    "total_questions": 46,
    "correct_answers": 7,
    "incorrect_answers": 17,
    "acceptable_unknown": 22,
    "average_quality_score": 3.130434782608696
},
"dorar_alternative_questions": {
    "total_questions": 60,
    "correct_answers": 31,
    "incorrect_answers": 22,
    "acceptable_unknown": 7,
    "average_quality_score": 3.0166666666666666
}
```


Figure 4: Evalution for both quality and grading

for the pair question in dorar generated questions we noticed that altenative pair may get diffrent grades! as show below.

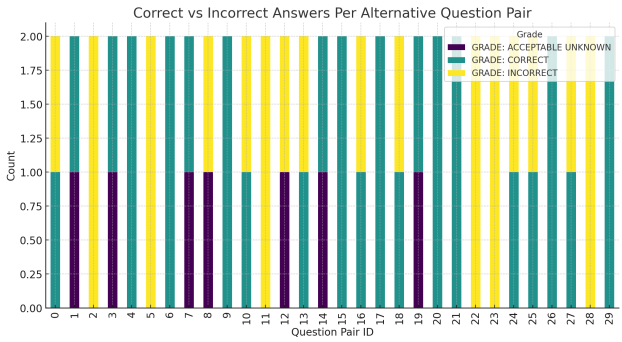

Figure 5: Pairs evaluation

This show that the way of asking the question may change the answer.

8.4 Challenges in Evaluation

Several issues were identified during the evaluation process:

- **Unrealistic or Incorrect Expected Answers:** Some expected answers were based on incorrect premises. For example, consider the following question from the evaluation dataset:

```
{
  "source": "dorar_alternative_questions",
  "question": "ما هي الطريقة المثلى لممارسة مسح الأذنين في الصلاة؟",
  "expected_answer": "يجب على المسلم أن يدخل سبابتيه في صماخيهما ويمسح بإبهاميه ظاهرهما، وهو ما يتفق عليه علماء المذاهب الفقهية الأربعة.",
  "model_answer": "المسح على الأذنين في الصلاة هو جزء من الطهارة، ويكون كالتالي: 1. بعد غسل اليدين، يُستحب أن يُمسح الأذنان بيدين مبتلتين. 2. يُمرر الماسح يديه على الأذنين، بحيث يُمسح ظاهر الأذنين بباطن الأصابع، ويدخل السبابة في داخل الأذن.3. يُفضل أن يكون المسح مرة واحدة. هذه الطريقة تأتي ضمن سنن الوضوء، حيث يُعتبر المسح على الأذنين جزءًا من الطهارة. إذا كان لديك أي استفسار آخر يتعلق بالعبادات، فلا تتردد في طرحه.",
  "grade": "GRADE: INCORRECT"
}
```


Figure 6: Evalustion incorrect questions

In this case, the chatbot correctly identified that ear-wiping (مسح الأذنين) is part of ablution (الوضوء) and not prayer (الصلاة). However, the evaluation system marked it as incorrect because the expected answer was based on a misunderstanding. This highlights a fundamental flaw in the dataset.

- **Correct Handling of Uncertainty:** The chatbot frequently responded with "I don't know the answer" when relevant data was unavailable. This should be counted as a correct response, but the evaluation system sometimes penalized it.
- **Human vs. Automated Evaluation Discrepancy:** A separate human evaluation of chatbot responses showed significantly better results than the automated evaluation. The automated grading system sometimes misclassify partially correct responses as incorrect, leading to lower accuracy scores.
- **Bias in Alternative Question Dataset:** Some generated alternative questions were either too artificial or phrased in a way that did not align with real-world user queries. This resulted in misleading evaluation results.

8.5 Improvement Plan

To improve the chatbot evaluation process, we propose the following changes:

- **Refine the Evaluation Dataset:** Manually review the dataset to remove unrealistic questions and verify the correctness of expected answers.
- **Implement Human-in-the-Loop Review:** Augment automated evaluation with human assessment to better capture nuanced responses.

8.6 Deliverable

A detailed evaluation report, including computed accuracy, response quality ratings, and examples of both correct and incorrect responses, is provided in the json file.

9 Conclusion

In this project, we built an LLM-powered chatbot for Ahkam Sharia. In Task 1, we collected data about العبادات from a trusted website and saved it in a JSON file. In Task 2, we created an evaluation dataset with

questions and answers to test our chatbot. In Task 3, we designed prompts to guide both the chatbot's responses and the evaluation process. In Task 4, we converted the collected data into embeddings and stored them in a vector database, taking care to remove diacritical marks for better performance. In Task 5, we used LangChain to build a retrieval-based pipeline that fetches relevant information and uses an LLM to generate answers. In Task 6, we developed a simple web interface using Gradio so that users can interact with the chatbot easily. Finally, in Task 7, we evaluated the chatbot using both automated and human assessments. The results show that the chatbot works well for most queries, although the automated evaluation sometimes does not match human judgment. Overall, this project has been a good learning experience in building and testing a chatbot for sensitive topics.