Gebze Technical University
Computer Engineering Department

CSE 484 – Natural Language Processing

Turkish Question Answering System for Gebze Technical University Official Student Rules

HW-2 Report

Muhammet Akkurt

1901042644

Introduction

This project aims to design and implement a QA system tailored specifically for the rules and regulations of Gebze Technical University (GTU). University students often find it challenging to navigate dense regulatory documents to find answers to their queries. By addressing this issue, the proposed QA system seeks to streamline information retrieval, enabling students to access relevant information quickly and efficiently.

The system is built around transformer-based models, particularly fine-tuned for the Turkish language, to accurately process and extract answers from GTU's official student rules and regulations. It incorporates semantic search for retrieving the most relevant sections of the document and a question-answering model to extract precise answers. A user-friendly web interface ensures that the system is accessible and easy to use for GTU students.

Objectives

The primary objectives of this project include:

1. Parsing and preprocessing GTU's official student rules document to create a structured QA dataset.
2. Fine-tuning a transformer-based Turkish-language model for precise answer extraction.
3. Integrating semantic search to improve the system's accuracy and relevance.

4. Developing an intuitive web-based interface to enhance user experience.

This report outlines the methodologies, challenges, implementation details, and evaluation metrics of the QA system, aiming to provide a comprehensive understanding of its development and performance.

Design and Implementation

Data Preparation

To train the question-answering model effectively, it was necessary to extract, clean, and preprocess the text data from these PDFs systematically.

Reading Pages: Each page of the PDFs was processed iteratively to extract text while ensuring multi-page documents were handled seamlessly.

Filtering Non-content Text:

- Header and footer sections containing information such as "Doküman No," "Form No," and page numbers were excluded.
- Filtering was applied to separate header information from metadata.

First Page Handling: Headers, titles, and irrelevant metadata were identified and excluded. Uppercase section headers were captured to retain context hierarchy.

Subsequent Pages: Similar rules were applied to ensure only the main content was extracted. Skipped repeated or irrelevant markers, such as page numbers or document identifiers.

The cleaned text from all documents was consolidated into a single text file (merged_text.txt) for further processing. Each document's content was stored as a separate block to facilitate downstream annotation.

Challenges:

- The tables on the last page of the documents were cleaned manually.
- Some texts required manual cleaning of data that did not fit the overall structure.

QA Dataset Preparation

Usage of the Language Model:

To automatically generate question-answer pairs from the connections, various large language models (LLMs) were utilized. The fundamental steps of the study were carried out as follows:

1. **Model Selection**:
    - **Local Model Usage**: Initially, the "meta-llama/Llama-3.1-8B-Instruct" model was employed to generate questions and answers. This model was optimized for efficient and fast operation on GPU hardware.
    - **Usage of Larger Models**: Due to hardware limitations, larger and more powerful models were accessed via Github Marketplace. In this context, open-source models such as "Mistral Large 24.11" and "Meta-Llama-3.1-70B-Instruct" were used.
2. **Prompt Design**:
    - A special prompt was designed to guide the model in producing accurate and meaningful question-answer pairs from the connections. This prompt was carefully crafted to direct the interaction between the language model and the user, as well as to format the generated results.

- The prompt included the following rules:
  - Creation of precise questions and answers in Turkish.
  - Answers should contain direct quotes from the text.
  - Questions should be directly related to the content of the connection and meaningful.
  - The outputs should be presented in JSON format.

3. **Model Fine-Tuning and Adjustments**:
   - Hyperparameters such as temperature (to optimize output randomness), top-p (to control output probability), and repetition penalties were configured.
   - Outputs were further refined to ensure that the responses were both meaningful and detailed, aiming for high-quality results.

Creation of the Dataset

The generated question and answer pairs were manually supplemented with their original contexts. This process was carried out to ensure that each question and answer was consistent with its context. Additionally, the outputs from the language model were reviewed and aligned with the text using the "checker.py" script to ensure accuracy and consistency.

The dataset was structured as follows:

- **Context**: Meaningful sections extracted from regulatory texts.
- **Question**: Meaningful questions generated by the language model from the context.
- **Answer**: Answers directly quoted from the context.

The created QA dataset was generated from the 12 regulations listed below. It consists of a total of 808 elements.

```
pdf_files = [
    "YÖ-0005_Dikey_Geçiş_Lisans_Uygulama_Yönergesi R1.pdf",
    "YÖ-0032_Prof._Dr._Nejat_Goyunc_Kutuphanesi_Hizmetlerinden_Yararlanma_Yonergesi R1.pdf",
    "YÖ-0004 Çift Anadal Programı Yönergesi R2.pdf",
    "YN-0001 Ön Lisans ve Lisans Eğitim-Öğretim Yönetmeliği R7.pdf",
    "YN-0009_Yonetmelik_Formu_Lisans_Egitim_ve_Ogretim_Yonetmeligi-2.pdf",
    "YÖ-0002 Önlisans-Lisans İngilizce Hazırlık Eğitim-Öğretim ve Sınav Yönergesi R7.pdf",
    "YÖ-0012 Yandal Programı Yönergesi R3.pdf",
    "YÖ-0013 Önlisans-Lisans Programları Yatay Geçiş Yönergesi R3.pdf",
    "YÖ-0024 Öğrenci Toplulukları Kuruluş ve İşleyiş Yönergesi R4.pdf",
    "YÖ-0011 Uluslararası Öğrencilerin Lisans Programlarına Başvuru, Kabul ve Kayıt Yönergesi R7.pdf",
    "YN-0015 Yaz Öğretimi Yönetmeliği R1.pdf",
    "YO-0071_Temel_Bilimler_Fakultesi_Lisans_Eitimi_Staj_Yonergesi_R0.pdf",
```

Challenges:

- Language models running locally may fail to generate meaningful and high-quality question-answer pairs.
- The request limits of large language models operating on the cloud caused time delays.
- To enable the language models to generate more detailed question-answer pairs, the contexts extracted from the texts needed to be provided in smaller chunks. As a result, more frequent requests were made.

- Due to the potential for language models to lose context and token limitations, the contexts to be added to the dataset had to be manually appended, which was highly time-consuming.

Training Process

Answer Position Identification:

- For each example, the start and end positions of the answer within the context (start_char and end_char) were identified.
- If the answer could not be found within the context, the example was excluded from the dataset to ensure the data's consistency and validity.

Data Formatting

The dataset was converted into the required structure using the Dataset class from Hugging Face:

- **context**: The context text.
- **question**: The asked question.
- **answers**: The answer text and its start position in the context.

As a result of this process, a QA dataset containing only valid data was obtained.

Dataset Splitting

The dataset was divided into different subsets to be used during the training and evaluation phases:

- **Training Dataset (80%)**: Used for training the model.
- **Validation Dataset (10%)**: Used to evaluate the model's performance during training.
- **Test Dataset (10%)**: Reserved for assessing the model's final performance.

Tokenization Process

The steps used to prepare the model's inputs are as follows:

- **Tokenizer Selection**: The **dbmdz/bert-base-turkish-cased** tokenizer, optimized for Turkish, was utilized.
- **Context and Question Processing**: Each question-context pair was tokenized, with a maximum limit of 512 tokens.
- **Determination of Start and End Positions**: The start and end positions of the answer within the context were calculated and labeled for the model's training.

Fine-tuning the Transformer Model

Training Configuration:

- Learning Rate: Set to 3e-5 to ensure stable convergence during training.
- Batch Size: Set to 4 for both training and evaluation to balance memory usage and computation speed.

- Number of Epochs: The model was fine-tuned for 3 epochs to achieve an optimal balance between underfitting and overfitting.
- Evaluation Strategy: Model performance was evaluated at regular intervals during training to track its progress and optimize hyperparameters.

For the question-answering system, a transformer model was trained using the AutoModelForQuestionAnswering class. The model was fine-tuned to provide accurate answers to questions based on contexts in the Turkish language.

Challenges:

- In addition to the current model, other models such as FacebookAI/xlm-roberta-large and google/mt5-small were also trained. However, the best results were achieved with dbmdz/bert-base-turkish-cased, leading to the decision to use this model.

Evaluation

Evaluation Process:

The performance of the model was evaluated using the test dataset. The test dataset consisted of data that the model had not seen before and was designed to objectively measure the model's generalization capability. During the evaluation, two primary metrics were used to assess the accuracy and contextual alignment of the model's answers:

- **Exact Match (EM):** Measures the percentage of predictions that exactly match the ground truth.
- **F1 Score:** Evaluates the overlap between the predicted and true answers, considering both precision and recall.

Normalization Process:

During the evaluation, the answers were normalized by removing case sensitivity, punctuation, and extra spaces. This process ensured a fairer assessment of the model's predictions.

Results:

The metrics obtained from the evaluation conducted on the test dataset are as follows:

Model	Exact Match (EM)	F1 Score
dbmdz/bert-base-turkish-cased	**60.98**	**83.61**
FacebookAI/xlm-roberta-large	**50.00**	**66.47**

Overall Evaluation:

The model trained with **dbmdz/bert-base-turkish-cased** demonstrated higher success in generating meaningful and accurate answers in Turkish contexts. The high F1 Score highlights the model's flexibility and ability to understand the context, while deficiencies in the Exact Match (EM) score may be attributed to factors such as data diversity and variations in expressions.

Interactive Interface of the Model

Purpose of the Interface:

Both a backend and frontend were developed to provide the model's question-answering service on a user-friendly platform. This interface was designed to allow Gebze Technical University (GTU) students to easily access information from regulation texts. Users can input their questions in Turkish, and the system will provide context-based answers generated by the model.

Backend Structure

The backend was developed using the FastAPI framework to process questions, identify relevant contexts, and generate answers. The main components of the backend are as follows:

Semantic Search

Semantic search was implemented using sentence embeddings to determine the most relevant contexts for a user's query. The process includes the following steps:

- **Preparation of Embedding Vectors:**
    - All contexts and questions were pre-converted into embedding vectors using the sentence-transformers library.
    - Model Used: The **emrecan/bert-base-turkish-cased-mean-nli-stsb-tr** model was selected to

accurately capture the meaning of Turkish contexts.

- **Similarity Calculation:**
  - The user's question is converted into a vector using the embedding model.
  - The cosine similarity method is used to calculate the similarity score between the question vector and all context vectors.
- **Combined Similarity Score Calculation:**
  - The get_combined_similarity method combines question and context similarity scores using a weighted merging formula. The weight parameter (α) allows for fine-tuning the importance of question and context similarities.
  - α was set to 0.6 after testing multiple values to optimize the performance.
- **Selection of the Best Contexts:**
  - Contexts are ranked based on their combined similarity scores.
  - Contexts with a combined similarity score exceeding a threshold (e.g., 0.2) are considered valid.
  - The top two most similar contexts are combined and sent to the question-answering process.

    - If no suitable context is found, the user receives the message: *"Bu soru için uygun bir cevap bulunamadı."*

Frontend Design

The frontend provides a web-based interface where users can easily submit their questions and quickly view the answers.

User Interface (UI): Includes a text input field for users to type their questions and an answer display area to show the model's responses.

- **Question Submission:**
    - Users can type a question and click the "Ask Question" button to send the query to the backend.
- **Loading Animation:**
    - A loading indicator is displayed while the question is being processed, ensuring a smooth user experience.
- **Answer Display:**
    - Once the backend generates an answer, it is dynamically presented to the user in the answer display area.

Error Handling

- **Client-Side Validation:**

    - Prevents users from submitting empty questions by validating input fields.
- **Meaningful Error Messages:**
    - Any error messages returned by the backend are displayed in a user-friendly manner, ensuring clear communication with the user.

To ensure faster responses, all regulation contexts were pre-converted into embedding vectors and stored in memory. This process eliminates the need to process the contexts repeatedly for each query, allowing for quicker replies.

Challenges:

- To ensure faster responses, all regulation contexts were pre-converted into embedding vectors and stored in memory. This process eliminates the need to process the contexts repeatedly for each query, allowing for quicker replies.
- In addition to the current embedding model, other models, such as **paraphrase-multilingual-mpnet-base-v2**, were also tested. However, the **emrecan/bert-base-turkish-cased-mean-nli-stsb-tr** model was chosen due to its better performance.
- During fine-tuning for context selection, various numbers of contexts and similarity scores were tested. The model's ability to provide accurate answers was evaluated through these tests, and the parameters were adjusted accordingly.

Web Based QA System

To deploy the model as an interactive demo on the web, the backend and frontend were slightly modified for **Google Colab**. The following steps were taken:

- **Model and Dataset Upload**:
  - The trained model and dataset were uploaded to **Google Drive**.
  - The backend notebook was designed to execute step by step.
- **Ngrok Integration**:
  - A **Ngrok account** is required to expose the Colab server to the web.
  - Since the free version of Ngrok does not provide permanent or static URLs, both the backend and frontend were updated to dynamically handle the changing URL.
- **Session Management**:
  - Due to the limitations of Colab's free version, it is not possible to run the model for extended periods.
  - The backend must be re-executed in each session, as the Colab instance resets after a certain period.

REFERENCES

- https://www.gtu.edu.tr/icerik/1479/592/lisans-yonetmelik-ve-yonergeler.aspx
- https://huggingface.co/meta-llama/Llama-3.1-8B-Instruct
- https://github.com/marketplace/models/azureml-mistral/Mistral-Large-2411
- https://github.com/marketplace/models/azureml-meta/Meta-Llama-3-1-70B-Instruct
- https://huggingface.co/dbmdz/bert-base-turkish-cased
- https://huggingface.co/FacebookAI/xlm-roberta-large
- https://huggingface.co/google/mt5-small
- https://huggingface.co/emrecan/bert-base-turkish-cased-mean-nli-stsb-tr

- https://huggingface.co/sentence-transformers/paraphrase-multilingual-mpnet-base-v2
- https://ngrok.com/
- https://colab.research.google.com/