Question Answering System for Israel-Hamas War Articles

Wassaf Ali

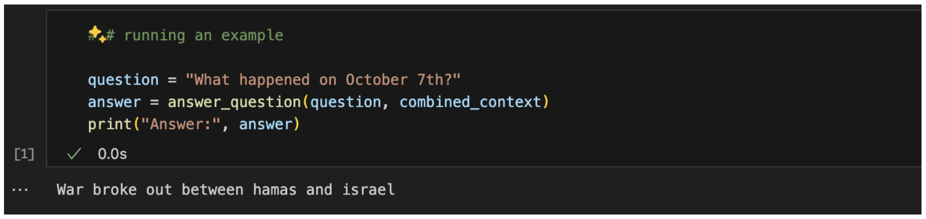
June 9, 2024

1 Introduction

This document provides a detailed explanation of the Python code used to create a rudimentary question-answering (QA) system for news articles related to the Israel-Hamas war. The system leverages a pre-trained transformer model to answer questions based on a provided dataset of articles.

2 Code Explanation

```
import json
import re
from transformers import pipeline
import wikipedia

# Load and filter data
with open('path_to_your_json_file.json', 'r') as file:
    articles = json.load(file)

def clean_text(text):
    # Remove special characters and digits
    text = re.sub(r'[^\w\s\.,]', '', text)
    return text

relevant_articles = [article for article in articles if 'Israel' in article['tex

for article in relevant_articles:
    article['cleaned_text'] = clean_text(article['text'])

# Fetch additional information
def fetch_wikipedia_content(query):
    try:
        summary = wikipedia.summary(query, sentences=5)
        return summary[:100] # Extract only the top 100 characters
    except wikipedia.exceptions.DisambiguationError as e:
        return None

additional_info = fetch_wikipedia_content('Israel-Hamas-war')

# Combine context
combined_context = " ".join([article['cleaned_text'] for article in relevant_art
(additional_info if additional_info else "")

# Load QA model (using a more advanced model)
qa_pipeline = pipeline('question-answering', model='bert-large-uncased-whole-wor

# QA function
def answer_question(question, context):
    result = qa_pipeline(question=question, context=context)
    return result['answer']

# Example question
question = "What happened on October 7th?"
answer = answer_question(question, combined_context)
print("Answer:", answer)
```



```
# running an example

question = "What happened on October 7th?"
answer = answer_question(question, combined_context)
print("Answer:", answer)
[1]  ✓ 0.0s
...  War broke out between hamas and israel
```

Figure 1: Output Example

2.1 Detailed Explanation

2.1.1 Imports

The script imports necessary libraries:

- `json` for handling JSON data.
- `re` for regular expressions.
- `transformers` for leveraging pre-trained language models.
- `wikipedia` for fetching additional information from Wikipedia.

2.1.2 Loading and Filtering Data

```
with open('path_to_your_json_file.json', 'r') as file:
    articles = json.load(file)
```

This part loads the JSON file containing news articles.

2.1.3 Cleaning Text

```
def clean_text(text):
    text = re.sub(r'[^\w\s\.,]', '', text)
    return text
```

The `clean_text` function removes special characters and digits from the article text.

2.1.4 Filtering Relevant Articles

```
relevant_articles = [article for article in articles if 'Israel' in article['tex
```

This line filters the articles to include only those containing both "Israel" and "Hamas".

2.1.5 Cleaning the Filtered Articles

```
for article in relevant_articles:
    article['cleaned_text'] = clean_text(article['text'])
```

The loop iterates through the relevant articles and applies the `clean_text` function.

2.1.6 Fetching Additional Information

```
def fetch_wikipedia_content(query):
    try:
        summary = wikipedia.summary(query, sentences=5)
        return summary[:100] # Extract only the top 100 characters
    except wikipedia.exceptions.DisambiguationError as e:
        return None

additional_info = fetch_wikipedia_content('Israel-Hamas-war')
```

This function fetches a summary of the Israel-Hamas war from Wikipedia and extracts only the top 100 characters.

2.1.7 Combining Context

```
combined_context = " ".join([article['cleaned_text'] for article in relevant_art
```

This line combines the cleaned text of relevant articles and additional Wikipedia information into a single context string.

2.1.8 Loading QA Model

```
qa_pipeline = pipeline('question-answering', model='bert-large-uncased-whole-wor
```

The code uses the Hugging Face `pipeline` to load a pre-trained QA model.

2.1.9 Answering Questions

```
def answer_question(question, context):
    result = qa_pipeline(question=question, context=context)
    return result['answer']
```

The `answer_question` function uses the QA model to find answers based on the provided question and context.

2.1.10 Example Usage

```
question = "What happened on October 7th?"
answer = answer_question(question, combined_context)
print("Answer:", answer)
```

The script provides an example question and prints the answer derived from the combined context.

3 Alternative Approaches

Several alternatives could be used to improve or modify the system:

- **Different Models**: Use other pre-trained models like `RoBERTa`, `ALBERT`, or `T5`.
- **Enhanced Cleaning**: Implement more sophisticated text cleaning techniques to handle noise better.
- **Additional Data Sources**: Augment the context with information from other reliable sources like DBpedia.
- **Fine-tuning**: Fine-tune the model on a domain-specific dataset to improve accuracy.

4 Conclusion

This document provided an in-depth explanation of the Python code to create a QA system for articles about the Israel-Hamas war. The system leverages the BERT model for question answering, cleans and filters the data, and augments context with Wikipedia information. Various alternative approaches were also discussed to enhance the system.