Question Answering System with BM25

Pranav Sarda

June 9, 2024

Contents

1 Introduction **1**

2 Choosing BM25 and Importing Libraries **2**
- 2.1 Reasoning 1
- 2.2 Dataset 1
- 2.3 Libraries 2

3 Data Preprocessing **2**
- 3.1 Loading the Dataset 2
- 3.2 `clean_article` Function 2

4 Preparing for BM25 Search **2**
- 4.1 Tokenization 2

5 BM25 and Function Definitions **2**
- 5.1 BM25 Initialization 2
- 5.2 `retrieve_articles` Function 2
- 5.3 `answer_question` Function 2
- 5.4 `answer_israel_hamas_question` Function (Optional) 2

6 User Interaction Loop **3**

7 Conclusion **3**

A Code Listings **3**

1 Introduction

The goal of this project is to develop a Question Answering (QA) system capable of retrieving relevant information from a dataset of news articles. The system leverages the BM25 algorithm, a popular method for document retrieval in information retrieval tasks.

2 Choosing BM25 and Importing Libraries

2.1 Reasoning

BM25 is chosen due to its efficiency and effectiveness in document retrieval tasks, making it suitable for retrieving relevant news articles based on a given query.

2.2 Dataset

The dataset consists of news articles stored in JSON format.

2.3 Libraries

The following libraries are utilized in this project:

- `json`: For parsing JSON data.
- `re`: For regular expressions and text manipulation.
- `tqdm`: For displaying progress bars during processing.
- `rank_bm25`: For implementing the BM25 algorithm.
- `transformers`: Potentially for using models like T5 (details not fully provided here).

3 Data Preprocessing

3.1 Loading the Dataset

The dataset is read from a JSON file containing news articles.

3.2 `clean_article` Function

This function performs the following preprocessing steps:

- Removal of special characters and punctuation.
- Conversion of text to lowercase for case-insensitive matching.
- Filtering out articles containing specific keywords such as "Israel" or "Hamas" due to domain-specific focus or limitations in handling these topics.

4 Preparing for BM25 Search

4.1 Tokenization

The articles are tokenized into sequences of words to facilitate analysis by the BM25 algorithm.

5 BM25 and Function Definitions

5.1 BM25 Initialization

A BM25 object is created using the tokenized articles, enabling efficient retrieval of relevant articles based on a query.

5.2 `retrieve_articles` Function

This function takes a question as input and uses the BM25 object to find the most relevant articles from the dataset.

5.3 `answer_question` Function

This function processes the retrieved articles, likely involving further analysis, to generate an answer to the user's question.

5.4 `answer_israel_hamas_question` Function (Optional)

This specialized function handles questions related to sensitive topics like "Israel" or "Hamas". It may involve predefined answers or additional filtering of the retrieved articles.

6 User Interaction Loop

The script enters a continuous loop where:

- The user is prompted to enter a question.
- The appropriate function (e.g., `retrieve_articles` and `answer_question`) is called to process the question and provide an answer.

7 Conclusion

This report outlines the development and implementation of a Question Answering system using the BM25 algorithm. By preprocessing the data and utilizing BM25, the system efficiently retrieves and processes relevant articles to answer user queries.

A Code Listings

Listing 1: Importing Libraries

```
import json
import re
from tqdm import tqdm
from rank_bm25 import BM25Okapi
from transformers import T5ForConditionalGeneration, T5Tokenizer

# Load the JSON file with specified encoding
file_path = 'news.article.json'

#/TR: Load the JSON file containing news articles
with open(file_path, 'r', encoding='utf-8') as file:
    #/TR: Load the JSON content into the 'articles' variable
    articles = json.load(file)

# Preprocess articles
def clean_article(article):
    #/TR: Remove special characters and noise from the article
    article = re.sub(r'\s+', ' ', article)  # Remove extra whitespace
    article = re.sub(r'[^\w\s]', '', article)  # Remove punctuation
    article = article.lower()  # Convert to lowercase
    return article

#/TR: Initialize an empty list to store cleaned articles
cleaned_articles = []
for article in tqdm(articles):
    #/TR: Access the article text using the 'articleBody' key
    article_text = article.get('articleBody', '')
    #/TR: Apply basic filtering for relevant articles
    if 'israel' in article_text.lower() or 'hamas' in article_text.lower():
        #/TR: Clean the article and add it to the list
        cleaned_articles.append(clean_article(article_text))

# Tokenize the articles for BM25
#/TR: Split each cleaned article into individual words
tokenized_articles = [article.split() for article in cleaned_articles]

# Initialize BM25
#/TR: Create a BM25Okapi instance for ranking articles
bm25 = BM25Okapi(tokenized_articles)
```

```
def retrieve_articles(query, k=5):
    #/TR: Tokenize the query and convert to lowercase
    tokenized_query = query.lower().split()
    #/TR: Retrieve the top 'k' relevant articles for the query
    results = bm25.get_top_n(tokenized_query, cleaned_articles, n=k)
    return results

# Load T5 model and tokenizer
model_name = 't5-small'  # You can choose a larger model for better performance
#/TR: Load the pre-trained T5 model for conditional generation
model = T5ForConditionalGeneration.from_pretrained(model_name)
#/TR: Load the pre-trained T5 tokenizer
tokenizer = T5Tokenizer.from_pretrained(model_name)

def answer_question(question, articles):
    #/TR: Combine the articles into a single context string
    context = " ".join(articles)
    #/TR: Create the input text for the T5 model
    input_text = f"question: {question} context: {context}"
    #/TR: Tokenize the input text and prepare it for the T5 model
    input_ids = tokenizer.encode(input_text, return_tensors='pt', truncation=True)
    #/TR: Generate the answer using the T5 model
    outputs = model.generate(input_ids)
    #/TR: Decode the answer and remove special tokens
    answer = tokenizer.decode(outputs[0], skip_special_tokens=True)
    return answer

def answer_israel_hamas_question(question):
    #/TR: Retrieve relevant articles for the question
    retrieved_articles = retrieve_articles(question)
    #/TR: Answer the question using the retrieved articles
    answer = answer_question(question, retrieved_articles)
    return answer

# Repeatedly ask for questions and provide answers
while True:
    #/TR: Prompt the user to enter a question or type 'exit' to stop
    user_question = input("Please enter your question (or type 'exit' to stop): ")
    if user_question.lower() == 'exit':
        print("Exiting the question-answering system. Goodbye!")
        break
    #/TR: Answer the user's question
    answer = answer_israel_hamas_question(user_question)
    print("Answer:", answer)
```