My Approach and the code

I am working on creating a Question Answering (QA) system using a JSON file containing 37,000 news articles. The goal is to focus on the Israel-Hamas war and provide accurate answers to user queries.

The first step is data preprocessing, which involves loading the JSON data and cleaning the text to make it suitable for the QA system. I will utilize the BM25 algorithm to filter relevant articles related to the Israel-Hamas war based on the query.

To answer user questions, I will use a pre-trained QA model from Hugging Face's Transformers library. This model takes a question and a context (a combination of relevant articles) and returns an answer.

This QA system leverages the BM25 algorithm to filter relevant articles and a pre-trained BERT model to provide accurate answers based on the most pertinent articles related to the query.

The Code:

```python
import json

import re
import nltk
from rank_bm25 import BM25Okapi
from transformers import pipeline

nltk.download('punkt')

# Load JSON data with specified encoding
with open(r'C:\Users\Arpit0\Downloads\news.article.json', 'r', encoding='utf-
8') as f:
    articles = json.load(f)

# Data Preprocessing
def clean_text(text):
    # Remove special characters, punctuation, and irrelevant symbols
    text = re.sub(r'\s+', ' ', text)
    text = re.sub(r'\n', ' ', text)
    text = re.sub(r'[^\w\s]', '', text)
    return text.lower()

for article in articles:
    article['cleaned_body'] = clean_text(article['articleBody'])

# Filtering relevant articles using BM25
corpus = [article['cleaned_body'] for article in articles]
tokenized_corpus = [nltk.word_tokenize(doc) for doc in corpus]
bm25 = BM25Okapi(tokenized_corpus)
```

```python
def filter_relevant_articles(query, corpus, bm25, top_n=10):
    tokenized_query = nltk.word_tokenize(query.lower())
    doc_scores = bm25.get_scores(tokenized_query)
    top_n_indices = doc_scores.argsort()[-top_n:][::-1]
    return [corpus[i] for i in top_n_indices]

# QA Model Setup
qa_pipeline = pipeline("question-answering", model="bert-large-uncased-whole-
word-masking-finetuned-squad")

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

# Example Question
question = "What happened at the Al-Shifa Hospital?"

# Retrieve relevant articles
relevant_articles = filter_relevant_articles(question, corpus, bm25)

# Combine relevant articles to form context
context = " ".join(relevant_articles)

# Get answer
answer = answer_question(question, context)

print("Question:", question)
print("Answer:", answer)
```