Final Report - Semantic Search Engine for Efficient Wikipedia Querying

Vibhav Rajkumar, Agrim Kataria , Aayush Bandopadhyay , Aditya Adusumalli

Summary

The primary objective of this project is to develop a semantic search engine specifically for Wikipedia's computer science articles, enhancing the precision and accessibility of searches through a deeper understanding of user queries' context and intent. The interface is designed to be intuitive, supporting a broad range of user inputs, from simple to complex queries, while the engine uses a two-model architecture to provide multiple options. Advanced data processing techniques also assist in streamlining the application.

The engine utilizes both machine learning and traditional algorithms. The modern BERT transformer model is commonly used in NLP tasks, and we leverage its capabilities in our engine. Meanwhile, the TF-IDF model is a traditional, non-machine learning model that also provides highly accurate search capabilities.

Introduction

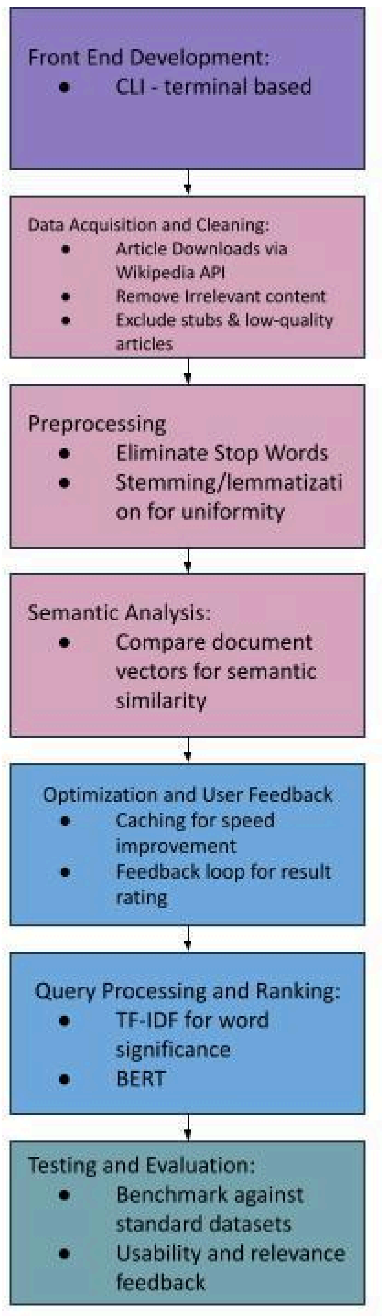
Wikipedia serves as an indispensable resource in the digital age, offering an extensive database of information to users worldwide. However, the platform's vast and diverse content presents significant challenges in effective information retrieval, particularly for computer scientists seeking high-level overviews of complex topics. This project aims to develop a semantic search engine, employing techniques learned in CS 410, to transform how users access information on Wikipedia. By enhancing the efficiency and accuracy of searches specifically within the computer science domain, the project focuses on improving the speed and relevance of search results through sophisticated data processing and natural language understanding techniques, simplifying the task of navigating Wikipedia's extensive corpus.

Data Preprocessing

Wikipedia is a widely used site that includes millions of large, wordy articles and documents ranging on a variety of different topics. In order to enhance data quality and relevancy, we have integrated several data preprocessing techniques on Wikipedia's large corpus, focusing specifically on computer science articles. Firsty, we utilized Wikipedia's API in our code to include only the most commonly accessed computer science articles and consequently ensured a targeted and relevant data set. The Wikipedia API also assists with data cleaning as it automatically excludes sparse articles. Additionally, the API removes irrelevant items such as external links and images as it includes data in raw text format. We also conducted other forms of text preparation for efficient tokenization by removing punctuation, special character, excess whitespace, and standardizing capitalization. We removed the complexity of our text data and enabled efficient querying through a stop word removal function to exclude commonly used and low information words. This allows our queries to primarily focus on relevant terms. Also, through the stemming and lemmatization techniques to condense words to their root forms and further simplifying the text data. Overall, these data preprocessing techniques are crucial for our VSM to enable more accurate indexing/retrieval by simplifying the text input.

Architecture

The architecture presents a structured pipeline for a data processing system, starting with a front-end which uses a command line interface on any terminal on one's computer, suitable for user interactions. The subsequent stages focus on offline and online operations, beginning with data acquisition and cleaning using the Wikipedia API to ensure the quality and relevance of content. Preprocessing tasks such as removing stop words and applying stemming or lemmatization are handled offline to prepare data for analysis. Semantic analysis is conducted online, comparing document vectors to assess semantic similarity. This is followed by an optimization phase that incorporates caching for improved speed and a user feedback loop to refine results. The system employs advanced text processing techniques like TF-IDF and BERT during the query processing and ranking phase, and it concludes with thorough testing and evaluation to ensure usability and relevance based on standard datasets and user feedback.


Front End Development:
CLI - terminal based
Data Acquisition and Cleaning:
Article Downloads via Wikipedia API
Remove Irrelevant content
Exclude stubs & low-quality articles
Preprocessing
Eliminate Stop Words
Stemming/lemmatization for uniformity
Semantic Analysis:
Compare document vectors for semantic similarity
Optimization and User Feedback
Caching for speed improvement
Feedback loop for result rating
Query Processing and Ranking:
TF-IDF for word significance
BERT
Testing and Evaluation:
Benchmark against standard datasets
Usability and relevance feedback

Code Description

This Python script is designed to create a semantic search engine specifically for retrieving computer science-related articles from Wikipedia. The process begins with the get_best_articles function, which uses Wikipedia's API to fetch articles from predefined computer science categories such as algorithms, programming languages, and machine learning, among others. It handles pagination to ensure all relevant articles within each category are retrieved, capped at 2000 articles in total. Subsequently, the write_article_paragraphs function fetches the introductory paragraph of each article using the Wikipedia API and stores these paragraphs locally in a pickle file. This setup allows for efficient data management and quick access during the search process.

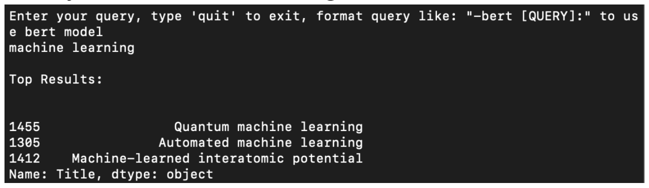
Once the data is collected and stored, the script transforms this dataset into a DataFrame for easier manipulation. The main function checks for the existence of the local data file and initiates the data collection process if the file isn't found. After loading the data, preprocessing steps are applied to prepare the text data for search queries. Users can interact with the system through a command-line interface that allows them to input search queries. The script supports two search methodologies: traditional keyword-based search using the BM25 algorithm and a semantic search using BERT embeddings. For semantic search, users can prefix their query with "-bert" to engage the BERT model, which computes cosine similarity between the query and document embeddings to find the most relevant articles. The top results are then displayed to the user, providing a direct and user-friendly way to explore computer science topics on Wikipedia.

Evaluation

In our comprehensive evaluation of the semantic search engine designed for Wikipedia's computer science articles, we discovered some intriguing results: traditional search algorithms such as TF-IDF not only hold their ground but also outperform the BERT model in certain scenarios. This finding emphasizes the enduring relevance and effectiveness of traditional methods even amidst the burgeoning AI revolution. Despite this, the pretrained BERT model demonstrated notable advantages in speed, processing queries significantly faster than its traditional counterparts, showcasing its practical utility in enhancing real-world applications. This project, specifically tailored to facilitate the querying of complex computer science topics, proves to be an invaluable tool for CS/CE majors who depend on Wikipedia as a critical learning resource. By improving both the speed and accuracy of search results, this search engine is poised to significantly aid users seeking to deepen their understanding of complicated subjects through Wikipedia. We are optimistic that this initiative will prove exceptionally useful to those who utilize Wikipedia to explore and comprehend advanced topics, offering a blend of traditional and modern techniques for optimal information retrieval.

Example Results (First Screenshot TF-IDF, Second Screenshot BERT)

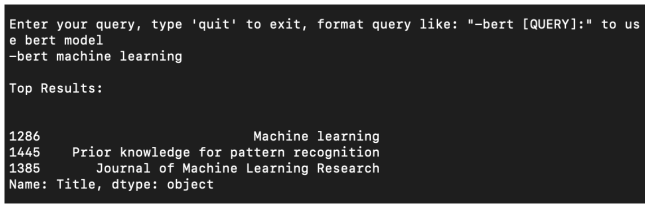
Query: “Machine Learning”



```
Enter your query, type 'quit' to exit, format query like: "-bert [QUERY]:" to us
e bert model
machine learning

Top Results:


1455                Quantum machine learning
1305              Automated machine learning
1412    Machine-learned interatomic potential
Name: Title, dtype: object
```

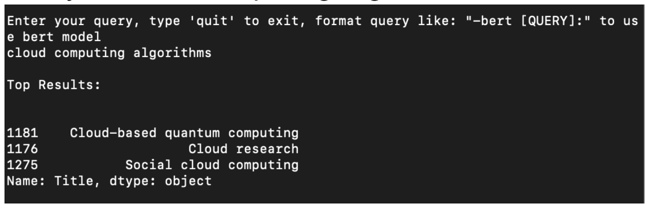


```
Enter your query, type 'quit' to exit, format query like: "-bert [QUERY]:" to us
e bert model
-bert machine learning

Top Results:


1286                          Machine learning
1445    Prior knowledge for pattern recognition
1385       Journal of Machine Learning Research
Name: Title, dtype: object
```

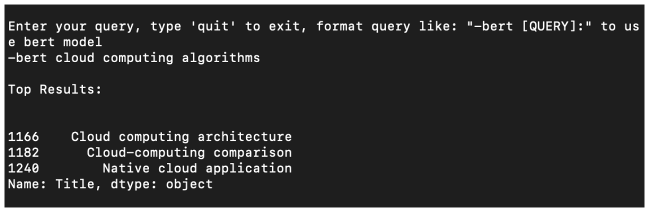
Query: “Cloud Computing Algorithms”



```
Enter your query, type 'quit' to exit, format query like: "-bert [QUERY]:" to us
e bert model
cloud computing algorithms

Top Results:


1181    Cloud-based quantum computing
1176                   Cloud research
1275           Social cloud computing
Name: Title, dtype: object
```

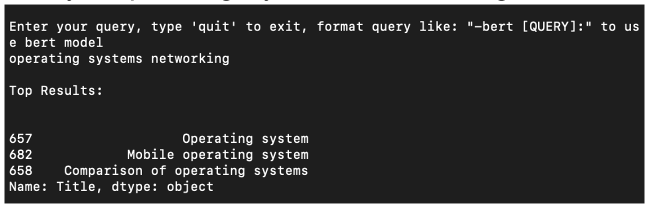


```
Enter your query, type 'quit' to exit, format query like: "-bert [QUERY]:" to us
e bert model
-bert cloud computing algorithms

Top Results:


1166    Cloud computing architecture
1182      Cloud-computing comparison
1240        Native cloud application
Name: Title, dtype: object
```

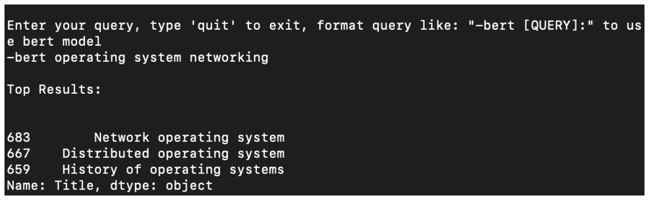
Query: “Operating Systems Networking”



```
Enter your query, type 'quit' to exit, format query like: "-bert [QUERY]:" to us
e bert model
operating systems networking

Top Results:


657                   Operating system
682            Mobile operating system
658    Comparison of operating systems
Name: Title, dtype: object
```



```
Enter your query, type 'quit' to exit, format query like: "-bert [QUERY]:" to us
e bert model
-bert operating system networking

Top Results:


683        Network operating system
667    Distributed operating system
659    History of operating systems
Name: Title, dtype: object
```