CIS 492 Big Data

FINAL PROJECT PROPOSAL

John Kubas 2842254

Logan Graham 2864014

Intelligent Wikipedia Search Engine with Inverted Index and TF-IDF Ranking

1. **Description of Big Data in Size and Format, Data Collection Plan:**

   For This project, we plan to use the English Wikipedia dataset found here: https://dumps.wikimedia.org/enwiki/20250401/#:~:text=enwiki%2D20250401%2Dpages%2Darticles.xml.bz2. This data set is free and publicly available, so to collect the data all we had to do was simply download it. This dataset is 22.1GB compressed and approximately 100GBs uncompressed. It contains an XML document for approximately 7,000,000 wiki articles. For testing and validation purposes we will begin with smaller data set of the same structure found here: https://dumps.wikimedia.org/enwiki/20250401/#:~:text=enwiki%2D20250401%2Dpages%2Darticles1.xml%2Dp1p41242.bz2 which is approximately 1GB in size and contains ~20,000 documents and scale the project from there. Each document includes an article title, wikitext body and metadata. This format would be classified as semi-structured and suitable for text analytics. We plan to convert these semi-structured XML documents into a structured CSV document containing only the relevant data and discarding metadata.

2. **Goal of Your Big Data Analytic Project with What Kind of Intelligent Analytic Functionality, Features of Your AI/Big Data Analytic Application**

   The goal of this project is to build an intelligent search engine that processes a large collection of Wikipedia articles and allows the user to submit a query and find the most relevant articles. This application leverages *information retrieval (IR)*

techniques to provide *intelligent analytic functionality* through efficient document lookup and relevance ranking, addressing the project's intended focus on AI-driven analytics.

2.1 Intelligent Analytic Functionality

We plan to use a two level, fully structured inverted index stored in an SQL database to maintain relevant document information similar to the first extra credit opportunity in Lab 4_3 phase 1. We will use this document information to perform TF-DF vectorization on the documents to compute relevance scores in relation to the user's query. We will then apply a cosine similarity algorithm to choose the most relevant documents. This strategy mimics existing AI and search engine strategies in determining responses and information used to respond to user queries.

2.2. Features to be Implemented (Deliverables)

- Query Interface: a command line interface where the user can submit a query. The documents returned will be ranked by the cosine similarity algorithm and may return a set number of documents or any amount over a certain similarity threshold
- Scalable Indexing: the use of an inverted index allows the program to scale in size efficiently as we add more data to the set
- Evaluation metrics: we will use the industry standard precision@K (where K equals the sample size we want to test) formula to determine the quality of our results

3. Big Data Processing Plan, Methods

We plan to implement a multi-step pipeline to transform the raw big data in our dataset into a highly searchable system by leveraging the principles discussed above such as an NLP pipeline, an inverted index, and TF-IDF relevancy scores used to find cosine similarity. Here is the tentative plan in sequential order:

1. Extract article title and content using xpath/wikitextparser, and store these fields in a CSV file (semi-structured data → structured data)
2. Preprocess the text by passing each article's body content into an NLP pipeline utilizing tokenization, lemmatization, NER (named entity retrieval), and POS (part of speech) to extract context and relevant information in the article
3. Inverted Index construction:
    1. Dictionary table (Term, TotalDocsFreq, TotalCollectionFreq) and
    2. Posting table (Term, DocID Term_Freq)

       Where the DocID will be the article title
4. TF-IDF calculation: we can utilize the scikit-learn's TfidfVectorizer to create a document-term matrix as discussed in class lecture
5. Query processing: we will pass the user's query into the nlp pipeline to extract the relevant information, and use the data structures mentioned above to calculate the most relevant articles.

4. **Investigate on Platform/Systems/Tools/APIs to Use**

We plan to build our application in *python*, as it is what we have been using all semester in this class. We plan to store the inverted index in a *mySQL* server. The modules we will implement will be spacy for NLP, scikit-learn for TF-IDF calculation and cosine similarity matrices. Pandas to create and organize the CSV created by our pre-processing. Wikitextparser (https://pypi.org/project/wikitextparser/) to aid in our pre-processing (approved by Dr. Sunnie). We plan for this application to run on any machine with at least 8GB of ram, to accomplish this we will process the data in batches to conserve memory and improve performance.