**Big Data Assignment some rough points before starting**

## **Language to be used:**

Python or Java

**Why** **Java?**

Because it is much better and more efficient with Hadoop boosting performance.

**Why Python?**

Because we are familiar with the language and have some practice and grip over it.

## **Basic Steps:**

Based on the given instructions and dataset description, here's how you can approach developing the Naïve Search Engine utilising MapReduce, focusing on the ARTICLE\_ID and SECTION\_TEXT columns and following the basic principles of the vector space model for information retrieval:

**Setup Environment:** Ensure you have Hadoop installed and configured properly in your environment. Set up a local development environment for testing MapReduce jobs.

**Data Preprocessing:** Extract the necessary columns (ARTICLE\_ID and SECTION\_TEXT) from the provided dataset. Perform basic text cleaning operations such as removing punctuation, converting text to lowercase, and tokenisation. Optionally, perform additional preprocessing steps like stopword removal and stemming.

**Indexing Phase:**

**Implement the Word Enumeration task using MapReduce:**

**Mapper:** Tokenize SECTION\_TEXT and emit (word, 1) pairs.

**Reducer:** Aggregate word frequencies to generate a set of unique words and assign unique IDs to each word.

**Implement the Document Count task using MapReduce:**

**Mapper:** Tokenize SECTION\_TEXT and emit (word, ARTICLE\_ID) pairs.

**Reducer:** Count the number of unique ARTICLE\_IDs for each word to calculate IDF values.

**Indexer Phase:**

**Implement the Indexer task using MapReduce:**

**Mapper:** Tokenize SECTION\_TEXT, calculate TF/IDF weights for each word, and emit (ARTICLE\_ID, word\_id: TF/IDF) pairs.

**Reducer:** Aggregate TF/IDF values for each ARTICLE\_ID to generate the vector representation of each document.

**Query Processing Phase:** Implement the Query Vectorizer function to generate the vector representation of a query using TF/IDF weights. Implement the Relevance Analyzer task using MapReduce.

**Mapper:** Receive the query vector and document vector pairs, calculate the inner product, and emit (ARTICLE\_ID, relevance\_score) pairs.

**Reducer:** Aggregate relevance scores and sort the documents based on relevance.

**Testing:** Conduct local testing on a smaller dataset to validate the correctness and performance of your MapReduce jobs. Use unit tests to ensure the individual components work as expected.

**Deployment:** Package your MapReduce jobs into a JAR file. Deploy your JAR file to your Hadoop cluster and run the indexing and query processing tasks using Hadoop Streaming. Integration and User Interface: Integrate the MapReduce jobs into a cohesive search engine application. Provide a user interface for users to input queries and display the relevant documents.

**Version Control and Collaboration:** Set up a public GitHub repository to track your project's progress and collaborate with your team. Use incremental commits to document changes and facilitate collaboration.

**Documentation and Reporting:** Document your implementation details, including design choices, algorithms used, and any challenges faced. Provide instructions for running the indexer and query tasks using Hadoop commands. Prepare a report summarizing your approach, results, and any insights gained from the project. By following these steps, you can develop a basic search engine utilizing MapReduce, adhering to the principles of information retrieval and leveraging the capabilities of Apache Hadoop's MapReduce paradigm.

# LEAVE THE ABOVE STUFF FOR NOW. WE WILL BEGIN ACCORDING TO WHAT IS WRITTEN BELOW:

The first step you should take is to set up your development environment and prepare your dataset for processing. Here's what you can do:

1. **Set Up Development Environment:** Install Apache Hadoop on your local machine or set up a Hadoop cluster if you have access to one. Ensure that Hadoop is properly configured and running without any errors. Set up any necessary dependencies for your Python MapReduce scripts, such as hadoop-streaming.
2. **Download and Extract Dataset:** Download the dataset provided in CSV format from the provided link. Extract the dataset files to a directory on your local machine where you'll be performing data processing.
3. **Inspect Dataset:** Open the dataset CSV file(s) using a text editor or a spreadsheet program to understand its structure and contents.
4. **Identify the columns relevant to your task:** ARTICLE\_ID and SECTION\_TEXT.
5. P**lan Data Preprocessing:** Decide on the specific text cleaning operations you'll perform on the SECTION\_TEXT column, such as removing punctuation, converting text to lowercase, and tokenisation. Determine if any additional preprocessing steps like stopword removal or stemming are necessary for your task.
6. **Set Up Project Structure:** Create a new directory for your project. Organize your project directory with subdirectories for input data, output data, code/scripts, and any other necessary files.