Project

Implementing an inverted index using Spark and a relational database

It is recommended you form groups of 2 people.

This project is about implementing an inverted index using Apache Spark for building the index and a relational database (e.g. SQLite) for storing the index. You should use Python (PySpark) for this project. Storing the index in a database offers the benefit of using the B-Tree data structure offered by a relational database instead of building it from the scratch.

To get a general idea you can start by reading the following paper:

"Using a Relational Database for an Inverted Text Index" by Steve Putz
http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.39.2655&rep=rep1&type=pdf

However, the project asks for a simplified version of the solution presented in that paper.

What you should do:

1. Build the index using a document collection.
2. Create database tables for storing the inverted index.
3. Implement the keyword search functionality.
4. Implement result ranking using the TF-IDF measure.
5. Implement a simple interface for giving keyword queries and showing results.

What you should submit:

Python notebooks containing your work.

Notes

1. Two datasets are given; a real one from Reuters which contains more than 19,000 documents, and a small sample of 5 documents in order to help with the writing and testing the code. Once you make sure that the code is correct, run the code on the real dataset. Processing the real dataset will take some time, in the order of few minutes.

2. The naming of documents is done as 1.html, 2.html, etc.

3. A skeleton notebook is given to get started with building the index using PySpark. There are TODO places where you should put your code. Also, you should replace the creation of dummy RDDs with constructs that build the needed RDDs. The dummy RDDs are only given to illustrate the structure of the RDDs you should create.

4. The suggested platform to do the project is Google Colab.

5. Compression of posting lists is not required for this project.

6. The results of the Spark processing will be saved first to disk in the form of two text files (see posted notebook). Then these files need to be processed (by some other notebook) and loaded to the database.

7. You can use SQLite for this project. The table schemas are as follows.

8. When creating the index, you should remove the *stop-words*. A list of the English stop-words can be obtained from the NLTK package (see posted notebook).

```
CREATE TABLE IF NOT EXISTS postings (
      word VARCHAR(100) PRIMARY KEY,
      postinglist_freq_tfidf TEXT
);
```

```
CREATE TABLE IF NOT EXISTS docmag (
      docid INT PRIMARY KEY,
      maxf INT,
      mag FLOAT
      );
```