Search the Bible - DESIGN

Data set:

Our dataset consists of 10000 documents. Each document has the Citation, Book name, Chapter number, verse number and the verse. This dataset was taken as CSV(Comma Separated Values) file.

Technologies:

Python
Flask Web framework
HTML
BootStrap 4(CSS + JS)

Implementation:

We used vector space model to rank the documents. The steps of the process which includes construction of data structures and ranking the documents with respect to a query are described here.

The dataset is read using python - CSV library. The text part in each document is tokenized and each term is stemmed using python - nltk package. The dictionary with keys as document names and values as the list of processed terms of its text is constructed while reading the CSV file.

Step 1 : **docs = {DocName : [term1, term2, term3….]}**

Also, all the processed terms are dumped into a list during the *Step 1*. The list is then converted into a set to get rid of any duplicates. For every term in the resultant set, the documents in which it is present along with the count in the document is stored as a dictionary. This dictionary is added to inverted index (which is another dictionary) with key as the term.

Step 2 : **terms = {term : { docName : count}}**

A tf-idf matrix (number of documents * number of unique-terms) with rows and columns as is constructed. The formula for computing tf-idf value of a term in a document is

```
(log(1+tf)* log(N/df)) * (cosine-normalisation)
```

Where tf is the Frequency of the term in the document.
df is number of documents having the term.
and cosine-normalisation = sqrt(sum(square(**(log(1+tf)* log(N/df))**)))

Step 3 : **[[(tf-idf).....].....]**

A query given is taken as a document,tokenized and stemmed and tf idf values are computed and stored in a list. If any of the words in the query are not present in any of the documents, its closer word takes its place in the query string.

Step 4 : **[(tf-idf).....] for the query**

Scoring is done by multiplying and adding the corresponding tf-idf values of query with each document. The scores are then sorted in reverse order and top 10 documents are returned.

Step 5 : **[doc1, doc2, doc3……, doc10]**

Only the relevant documents are returned always because even if the word is not present, the results are shown for the closest word. Hence, the recall is 100%.

Data structures :

Dictionary of Dictionaries : Inverted Index

`Terms = {term: {documentName: termCountInTheDocument}}`

(For quick look up of the count of a term in a document.

List 2D : tf-idf Matrix

Set : unique terms from all the documents in the dataset to enable suffix search and typo/closer-word searches.

Time:

For 1000 documents,

Reading Documents:	1s
Inverted index construction:	2s
Matrix Construction:	1s
Search:	~0s

For 5000 documents,

Reading Documents:	6s
Inverted index construction:	15s
Matrix Construction:	6s
Search:	3s

For 10000 documents,

Reading Documents:	13s
Inverted index construction:	43s
Matrix Construction:	19s
Search:	8.5s

For 30000 documents,

Reading Documents:	35s
Inverted index construction:	200s
Matrix Construction:	120s
Search:	40s