Com S 435/535 Programming Assignment 4
600 Points
Due: Dec 7, 11:59PM
No Late Submissions Please

This programming assignment is on information retrieval. Given a collection of documents you will to build a positional index and when a query arrives, you will retrieve top k documents.

Note that the description of a programming assignment is not a linear narrative and may require multiple readings before things start to click. You are encouraged to consult instructor/Teaching Assistant for any questions/clarifications regarding the assignment.

For this assignment, you may work in groups of two.

1 Term Proximity Score

Recall the positional index has a dictionary and a postings list corresponding to each term in the dictionary. Each entry of of the dictionary is of the form $\langle t, df_t \rangle$, where t is a term and df_t is number of documents in which t appears. The postings list for a term is of the following form:

$$[\langle DocID_1 : pos1, pos2, \cdots \rangle, \langle DocID_2 : pos_1, pos_2, \cdots \rangle \cdots]$$

Let q be a query and d be a document. In the lectures, we have seen how to assign $score(q, d)$ that measures the relevance of query q to document d. This score is based on the vector space model. Using positional index, we could arrive at a different way of scoring called *term-proximity score*. For this, we need to first define the notion of distance between two terms in a document. Let t_1 and t_2 be two terms and d be a document. If neither of the terms appear on the document or only one term appear in the document, then $Dist_d(t_1, t_2)$ is 17. If t_2 does not appear after t_1 in d, then $Dist_d(t_1, t_2) = 17$[1]. Otherwise, look at $postings(t_1)$ and $postings(t_2)$. Both these lists will have tuples of form $\langle d : p_1, p_2, \cdots \rangle$. Let $\langle d, p_1, p_2, \cdots, p_\ell \rangle \in postings(t_1)$ and let $\langle d, r_1, r_2, \cdots r_k \rangle \in postings(t_2)$.

$$dist_d(t_1, t_2) = \min\{\min\{r_i - p_j \mid r_i > p_j, 1 \leq i \leq k, 1 \leq j \leq \ell\}, 17\}$$

For example, if $\langle d, 6, 18, 21, 46 \rangle \in postings(t_1)$ and $\langle d, 5, 9, 11, 20, 34 \rangle \in posting(t_2)$, then $dist_d(t_1, t_2) = 2$. Note the function $dist_d$ is not symmetric, i.e, $dist_d(t_1, t_2)$ may not be equal to $dist_d(t_2, t_1)$.

Let $q = t_1, t_2, \cdots t_\ell$ be a query and d be a document. Then the term-proximity score of q with respect to d

$$TPScore(q, d) = \frac{\ell}{\sum_{i=1}^{\ell-1} dist_d(t_i, t_{i+1})}$$

If q has exactly one term, then $TPScore(q, d) = 0$.

[1] 17 is an arbitrary choice

2 Vector Space Model Score

Recall that in the vector space model, every document is represented as a vector.

Given a collection of documents $D_1, D_2, \cdots, D_N$, first preprocess the documents to extract all terms. Let $T = \{t_1, \cdots, t_M\}$ be the collection of all terms in the collection.

In lectures, we talked about weight of a term with respect to a document. For this assignment, use the following

$$w(t_i, d_j) = \log_2(1 + TF_{ij}) \times \log_{10}(N/df_{t_i})$$

where TF_{ij} is the frequency of t_i in d_j and df_{t_i} is the number of documents in which t_i appears.

Now, every document d_j corresponds to the following vector:

$$v_j = \langle w(t_1, d_j), w(t_2, d_j), \cdots, w(t_M, d_j) \rangle$$

Given a query q, we can view it as a (very short) document represent it as a vector v_q.

Giben a query q, form a vector corresponding to q

$$v_q = \langle w(t_1, q) \cdots, w(t_M, q)$$

Now, the vector space score of q with respect to d is

$$VSScore(q, d) = CosineSim(V_q, V_d)$$

Note that if we have access to positional index, then we can compute $VSScore$.

3 Your Task

You will build a program that pre-processes a collection of documents, and when a query arrives it will output top 10 documents that are relevant to the query. The relevance is calculated by using the following combination of $TSScore$ and $VSScore$.

$$Relevance(q, d) = 0.6 \times TPScore(q, d) + 0.4 \times VSScore(q, d)$$

Your program will consist of following classes and methods.

3.1 PositionalIndex

This class should have following constructor and methods. This class build an index for single words.

`PositionalIndex` Gets the name of a folder containing document collection as parameter.

`termFrequency(String term, String Doc)` Returns the number of times `term` appears in `doc`.

`docFrequency(String term)` returns the number of documents in which `term` appears.

`postingsList(String t)` Returns string representation of the $postings(t)$. The returned String **must** be in following format.

$$[\langle DocName_1 : pos_1, pos_2, \cdots \rangle, \langle DocName_2 : pos_1, pos_2, \cdots \rangle \cdots]$$

`weight(String t, String d)` Returns the weight of term t in document d (as per the weighing mechanism described above).

`TPScore(String query, String doc)` Returns $TPScore(query, doc)$.

`VSScore(String query, String doc)` Returns $VSScore(query, doc)$.

`Relevance(String query, String doc)` Returns $0.6 \times TSScore(query, doc) + 0.4 \times VSScore(query, doc)$.

This class may have additional methods.

4 QueryProcessor

This program will have a method named `topKDocs` that gets a query and an integer k as parameter and returns an arraylist consisting of top k documents that are relevant to the query.

5 Pre Processing

Do not remove any STOP words. Every word is a term convert every word into lowercase. Remove ONLY the following symbols from the text:

. , ” “ ? [ ] ′ { } : ; ( )

However, do not remove period if it is part of a decimal number, i.e, for example do not remove period from 9.23.

6 Data

`IR.zip` contains around 11, 000 files crawled from wiki about Baseball.

7 Report

Include the following in your report. Run your program on files from `IR.zip`. Pick 5 distinct queries so that

- One query with exactly one word
- One query with exactly two words
- One query with exactly 3 words
- Two queries with more than 3 words.

Please ensure that at least one of the (3 or more word) queries have propositions such as `of, for, to` etc.

For each query, list top 10 files along with TPScore, VSScore, and Relevance Score.

8 What to Submit

- PositionalIndex
- QueryProcessor
- Source files of additional classes that you used.
- Report

Only one submission per group please.