This program is on information retrieval; namely, given a collection of documents we build a positional index and when a query arrives, we retrieve top *k* relevant documents.

1: Term Proximity Score

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

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

Let q be a query and d be a document. A way of measuring the relevance of query q to document d is 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 in document d. If neither of the terms appear on the document or only one term appears 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 the form $\langle d : p_1, p_2, \cdots\rangle$. Let $\langle d : p_1, p_2, \cdots, p_l\rangle \in postings(t_1)$ and let $\langle d : r_1, r_2, \cdots, r_k\rangle \in postings(t_2)$.

$$dist_d(t_1, t_2) = \min\{\min\{r_i - p_j | r_i > p_j, 1 \le i \le k, 1 \le j \le l\}, 17\}$$

For example, if $\langle d, 6, 18, 21, 46\rangle \in postings(t_1)$ and $\langle d, 5, 9, 11, 20, 34\rangle \in postings(t_2)$, 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 $q = t_1, t_2, \cdots, t_l$ be a query. Then the *term-proximity score* of q with respect to d is

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

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

2: Vector Space Model Score

In the vector space model, every document is represented as a vector. Given a collection of documents D_1, D_2, D_N, we first preprocess the documents to extract all terms. Let $T = \{t_1, \cdots, t_M\}$ be the collection of all terms in the collection. The weight of a term with respect to a document is as the following:

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

where TF_{ij} is the frequency of t_i in d_j and df_{t_i} is the number of documents in which t 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, representing it as a vector v_q, that is,

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

[1] 17 is an arbitrary choice

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 the positional index, then we can compute $VSScore$.

3: Positional Index

The class ***PositionalIndex*** has the following constructor and methods. This class builds 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(Stirng t) returns the number of documents in which `term` appears.

postingsList(String t) returns string representation of the $postings(t)$. The returned string is in the following format.

$$[\langle DocName_1 : pos1, pos2, \cdots \rangle, \langle DocName_2 : pos1, pos2, \cdots \rangle \cdots]$$

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)`**

We built a program that pre-process a collection of documents, and when a query arrives it outputs top 10 documents that are relevant to the query. The relevance is calculated by using the following combination of $TSScore$ and $VSScore$:

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

Every word is a term and the program converts every word into lowercase and removes the following symbols from the text:

.,”?[]‘′{} :; ()

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

We ran the program on files from `data`, containing around 11,000 files crawled from wiki about baseball, with 5 distinct queries. The results are listed in the following.

query file ($TPScore$, $VSScore$, $Relevance$)

chicago

1. 1908_Major_League_Baseball_season.txt (0.0, 0.05652, 0.02261)
2. 1885_World_Series.txt (0.0, 0.03929, 0.01572)

3. Bob_Carpenter_(baseball).txt (0.0, 0.0259, 0.01036)
4. 1919_in_baseball.txt (0.0, 0.02408, 0.00963)
5. 1906_in_baseball.txt (0.0, 0.02323, 0.0093)
6. The_Heckler_(newspaper).txt (0.0, 0.02292, 0.00917)
7. 1905_Chicago_Cubs_season.txt (0.0, 0.02224, 0.0089)
8. 1907_Chicago_Cubs_season.txt (0.0, 0.02141, 0.00856)
9. 1876_St._Louis_Brown_Stockings_season.txt (0.0, 0.02071, 0.00829)
10. 1904_Chicago_Cubs_season.txt (0.0, 0.0207, 0.00828)

new york

1. 1958_Major_League_Baseball_season.txt (2.0, 0.04984, 1.21994)
2. 1957_Major_League_Baseball_season.txt (2.0, 0.01628, 1.20651)
3. 1933_New_York_Giants_season.txt (1.0, 0.04323, 0.61729)
4. 1937_New_York_Giants_(MLB)_season.txt (0.2, 0.03081, 0.13232)
5. 2002_New_York_Yankees_season.txt (0.11765, 0.06696, 0.09737)
6. Tommy_Clarke.txt (0.11765, 0.06, 0.09459)
7. 1958_New_York_Yankees_season.txt (0.11765, 0.05748, 0.09358)
8. 1936_New_York_Giants_(MLB)_season.txt (0.11765, 0.05136, 0.09113)
9. 1997_New_York_Yankees_season.txt (0.11765, 0.04513, 0.08864)
10. 1923_New_York_Giants_season.txt (0.11765, 0.039681, 0.08646)

chicago cubs season

1. 1925_Chicago_Cubs_season.txt (0.6, 0.01749, 0.367)
2. 1923_Chicago_Cubs_season.txt (0.1875, 0.04377, 0.13001)
3. 1922_Chicago_Cubs_season.txt (0.14286, 0.06556, 0.11194)
4. 1905_Chicago_Cubs_season.txt (0.16667, 0.00252, 0.10101)
5. 1921_Chicago_Cubs_season.txt (0.16667, 0.00214, 0.10085)
6. 1924_Chicago_Cubs_season.txt (0.16667, 0.00206, 0.10082)
7. 1926_Chicago_Cubs_season.txt (0.16667, 0.00185, 0.10074)
8. 1904_Chicago_Cubs_season.txt (0.16667, 0.00178, 0.10071)
9. 1911_Chicago_Cubs_season.txt (0.16667, 0.00172, 0.10069)
10. 1907_Chicago_Cubs_season.txt (0.16667, 0.00168, 0.10067)

1923 washington senators season

1. 1923_Washington_Senators_season.txt (0.11111, 0.03815, 0.08193)
2. 1955_Washington_Senators_season.txt (0.07843, 0.07687, 0.07780)
3. 1960_Washington_Senators_season.txt (0.11429, 0.01108, 0.07300)
4. 1923_New_York_Giants_season.txt (0.10256, 0.02863, 0.07299)
5. 1933_World_Series.txt (0.11429, 0.00741, 0.07153)

6. 1933_Washington_Senators_season.txt (0.07843, 0.05665, 0.06972)
7. Tilly_Walker.txt (0.07843, 0.03436, 0.0608)
8. American_League_Park.txt (0.08163, 0.02752, 0.05999)
9. 1932_Washington_Senators_season.txt (0.07843, 0.028, 0.05826)
10. Washington_Senators_(1891%E2%80%9399).txt (0.07843, 0.02646, 0.05764)

colored world series in 1924

1. 1927_Colored_World_Series.txt (0.11628, 0.01249, 0.07477)
2. 1924_Colored_World_Series.txt (0.07353, 0.07575, 0.07442)
3. 1926_Colored_World_Series.txt (0.11364, 0.0092, 0.07186)
4. 1924_World_Series.txt (0.09615, 0.02943, 0.06946)
5. Alex_Pompez.txt (0.09615, 0.02502, 0.0677)
6. 1924_Major_League_Baseball_season.txt (0.07353, 0.05622, 0.0666)
7. Bullet_Rogan.txt (0.09615, 0.02171, 0.06638)
8. Art_Nehf.txt (0.08929, 0.02933, 0.0653)
9. 1925_Colored_World_Series.txt (0.09615, 0.01438, 0.06344)
10. 1933_World_Series.txt (0.09615, 0.01384, 0.06323)