Algorithm

First store all the tokens and their locations. This is done using the hashMapVector() function in Document.java, and adding a getter for this new field. Then calculate the proximity score for each document that was retrieved by the retrieve() function. The proximity score algorithm is as such.

- Proximity between 2 words is their distance in number of words.
- If the words are in reverse, multiply by twice the actual distance between the words in the query string.
- Given a query Q with a set of n tokens {q1, q2,...,qn}. Where qi is the i-th token when reading the query from left to right, and a function *f(D, qi)* that maps from token to a set of locations in Document D.

```
for every document D:
        maintain a set L
        L = Union of f(D, qi) for each qi in Q
        sort L in ascending order
        Let n be the window size for L
        Using a sliding window, calculate the proximity pair of adjacent words in window
        Let Si be the sum of said pairs in the window i
        Let ki be the number of unique words in the window i
        Proximity score = min(Si * 2 ^ (n-ki)/ki for i in 1...num of windows)
        Retrieval score = vector score / proximity score
```

The algorithm encourages proximity of words by adding a multiplicative penalty to document with words that are out of order. It adds another penalty to documents that do not contain all of the query tokens by dividing by the # of unique words and multiplying by 2 ^ # of missing words. So a document with less of the query words would have a higher proximity and thus a lesser overall score. The algorithm takes O(n) time and O(n^2) space due to the token locations.

Analysis

- The baseline system had only 3 out of the top 10 documents that contained the query “real time” or “real-time” while my proximity system had 8 out of 10.
- For the query “mass spring system”, my proximity system had 3 relevant documents of the top 10 chosen with 2 of them at the top 2 while the baseline returned only 2 relevant documents ranked at the bottom half of the top 10.
- For the query “real world” the baseline system’s top 9 results were all pages with only Hello world in them. While all 10 of my results contained “real world” or “real-world”
- For the query “information technology” the baseline system had only 2 documents in its top 10 with the query in it while my proximity had all top 10 documents with the query in it.

How to run

```
java ir.vsr.InvertedIndex -html -proximity <path to corpora>
```