Text Search in Wade

Kabir Shah

1 Overview

Text search with Wade searches a set of documents for a query to find the most relevant set of documents. To do this, the following must be taken into consideration:

1. The amount of documents.
2. The amount of terms in a search query.
3. The significance of the term.

2 Index

An index can be generated to allow for optimized searches within a text. This index can be generated by:

1. Preprocessing each document to make them lowercase, free of punctuation, and free of stop words.
2. Splitting each document into a multiset of terms.
3. Generating a trie of the terms and storing the indexes of the corresponding documents containing the term.
4. Storing the weighted significance of the term in the documents.

2.1 Significance

The significance of the term t in the set of documents d can be represented by the function:

$$wm(t,d) = 1.5 - \frac{\sum_{i=0}^{|d|}[\frac{\mu(t)}{\sum_{p \in d_i}(\mu(p))}]}{|d|}$$

1. d is a set of multisets d_i
2. $t \in d_i$ for at least one d_i

3. $\mu(t)$ is the multiplicity of t in the multiset d_i
4. $\sum_{p \in d_i}(\mu(p))$ is the cardinality of the multiset d_i
5. $\frac{\mu(t)}{\sum_{p \in d_i}(\mu(p))}$ is the ratio of occurrences of the term t in the document d_i to the total amount of terms in the document d_i

This works by finding the average of how often the term appears within a document. After this, the significance is normalized between 0.5 and 1.5, allowing it to become higher when the average occurrence is lower. This allows for rarer terms to be amplified in significance.

3 Search

Searching for a query must follow the same preprocessing step used when building the index. Searching for a query can be done by:

1. Preprocessing the query to make it lowercase, free of punctuation, and free of stop words.
2. Splitting the query into a multiset of terms.
3. Searching the index for each term in the query.

3.1 Relevance

To find how relevant each term is to a document, the length of the query must be taken into consideration. The relevance of the term t in the query q to the set of documents d can be represented by the function:

$$wr(t, q, d) = wm(t, d)[\frac{1}{\sum_{p \in\ q}(\mu(p))}]$$

1. q is a multiset
2. $\mu(p)$ is the multiplicity of p in the multiset q
3. $\sum_{p \in\ q}(\mu(p))$ is the cardinality of the multiset q

This can be used to represent how much each term should affect the score of the query. It works by taking significance of the term and the length of the query into account.

3.2 Exact Search

To search the index for each term *except for the last*, the term can be split into characters to use as a key for the trie. Each character will be used as a key to look up inside the previous node (with the root being the index). As a result, a new node in the trie is found for the next character to be looked up in.

At the end, if every key was found in the index, and it has a list of documents with the term, then there was an exact match of the term. As a result, the score for the documents can be updated.

The score for documents found for the term t in the query q searched within the documents d can be updated by adding:

$$wr(t, q, d)$$

3.3 Depth-First Search

For the last term, a different process is used to update the scores. Since a user might still be typing the last term, it is treated as a prefix and a depth-first search is used.

The index is searched in the same manner as described above, but when an ending point is found, a different process is followed.

Instead of aborting the search, all child nodes are traversed for indexes and the documents are all updated for the term. The score is updated for the term t in the query q searched within the documents d by adding:

$$wr(t, q, d)$$

As a result, all of the documents that have a term with the same prefix are updated.