Complex Query Search Using Inverted Index

October 31, 2012

1 Introduction

In the previous assignment (Inverted Index), we searched for only one-word queries in the documents. We will now extend it to handle multiple-word queries. For example queries like *dog cat* or *dog or cat*. For such queries, we need to extend our search method for single word queries and also redefine the concept of relevance. Like, for both the example queries we search for both words independently but while returning the relevant documents, we return the document containing both the words first followed by others whereas for the latter query, we rank the one having maximum of any or both of these as most relevant. Also there are queries like *"self discipline"* called "phrase queries" (note the difference between AND queries and PHRASE queries - no inverted commas in the first one) which are composed of two separate words but always occur together (back-to-back). So now we need to look into the inverted index for each of the words and see if they always appear together.

For this assignment, you would be provided with the inverted indexes for the documents. For example, assuming document was - "it is what it is" then corresponding inverted index in the file will be written as -

```
"it" 0 0 3
"is" 1 1 4
"what" 2 2
```

where first term is the word followed by its index value in the inverted index table and then a list of positions in the document (starting from 0). And now the corresponding trie and inverted index for this document would be same as we made in Assignment 5.

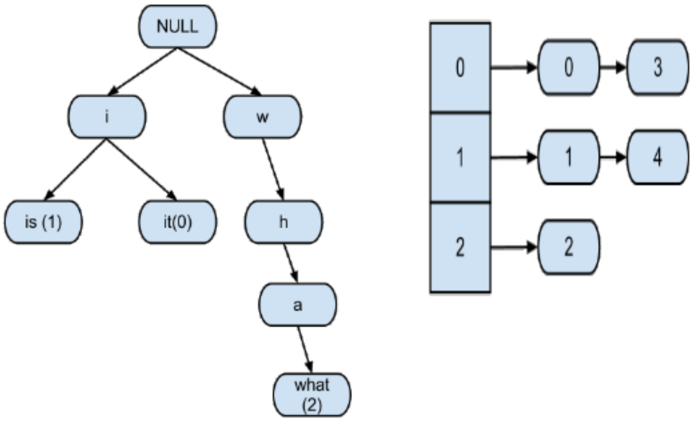

Figure 1: Trie and Inverted Index for Document 1

Possible query types are -

1. AND queries of the form: computer science. There can be any number of search terms.
2. OR queries of the form: computer or science. Here also there can be any number of search terms - each separated by an "or".
3. Phrase queries of the form: "computer science". Only two terms would be present in this.
4. Queries combining the above 3: "Computer science" or "information science" (here we have two phrases "Computer Science" and "Information Science", and an "or" between them)

Your program should proceed as follows -

1. Read the query (can be either from a file or from keyboard)
2. Parse the query (reduce into terms and check for keywords like "or", terms within " " or space separated terms)
3. Search for terms in the inverted indexes
4. Score each of the search results and present them in decreasing order of scores.

Before this, the program should read N inverted indexes from N files and build up the corresponding trie and the matrix-form of the index itself. The file names will be given at run time, example - java Assignment6 i1.txt i2.txt i3.txt

The Scoring scheme for different types of queries is -

1. The scoring scheme for individual search terms used in Assignment 5 should be used as a basis for single search term scoring.

2. For AND queries, higher score is to be given to those pages that contain all the search terms. Still higher score is to be given to pages that have the search terms in the order that they have been typed. If none of the the pages contain all the search terms then the program should delete at most 1 search term (chosen so that the results are maximized) and return the results for the rest (and indicate this to the user). If even deleting one term does not give any document containing all the remaining terms, then simply indicate this to the user.

3. For OR queries, first the pages should be scored using the AND score and then pages should be scored for individual terms.

4. For phrase queries, the location information stored in the inverted index has to be used to return the results. If there are not enough results with the phrase then AND results should be returned.