Boolean Retrieval - IR weekly report

hello underworld

22th April 2021

1 Some Initial Concepts

1.1 What's Boolean Retrieval

The Boolean retrieval model is a model for information retrieval in which we can pose any query which is in the form of a Boolean expression of terms, that is, in which terms are combined with the operators AND, OR, and NOT. The model views each document as just a set of words.

1.2 What's Inverted Index

The inverted index, actually, is the base of the Boolean Retrieval. First, we have to know 2 models which could describe the relationships between the dictionary and doc_id(where the dictionary appears) and more information including the frequency and the offset.

The following model is Forward Index

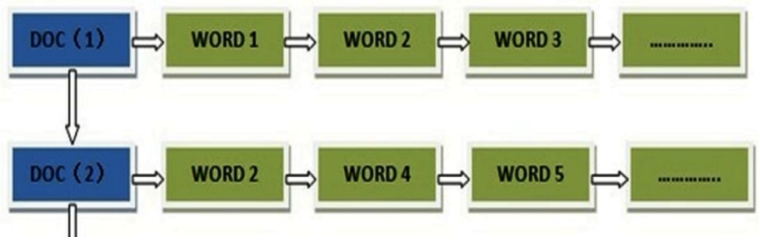

Figure 1: example of forward index

However, when this model is used to search for a word ,the whole text has to be scanned,costing a lot of time.
As a contrast, it will cost less time to get same result by using this model - Inverted Index ,as following.

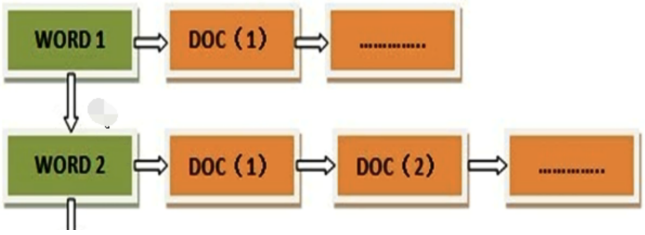

Figure 2: example of inverted index

In this model, just scan the dictionary once, until the word wanted is found, then the doc_id and more information about the word could be get in its postings. Obviously, it can't be

avoided to scan the whole document once to get two kinds of index. But it will be different when you search for some word using two distinct index.

2 Process of Boolean Retrieval

2.1 Step1:get the Inverted Index

1. Collect the documents to be indexed
2. Tokenize the text, turning each document into a list of tokens
3. Do linguistic preprocessing, producing a list of normalized tokens, which are the indexing terms
4. Index the documents that each term occurs in by creating an inverted in- dex, consisting of a dictionary and postings.

2.2 Step2: process the Boolean Query

1. Get the Boolean expression of the query
2. Scan the Inverted Index to get the postings of words in the expression.
3. Using the Boolean operators to process the postings, get the result of the query finally.

3 Some Algorithms in the Process of Boolean Retrieval

3.1 Intersection

A simple example of Intersection:Term_1 AND Term_2. Here is the algorithm for this operation(actually, we should get the posting lists of two words first):

```
INTERSECT(p1, p2)
 1  answer ← ⟨ ⟩
 2  while p1 ≠ NIL and p2 ≠ NIL
 3  do if docID(p1) = docID(p2)
 4        then ADD(answer, docID(p1))
 5             p1 ← next(p1)
 6             p2 ← next(p2)
 7        else if docID(p1) < docID(p2)
 8                then p1 ← next(p1)
 9                else p2 ← next(p2)
10  return answer
```

Figure 3: process of intersection of two postings lists

In the algorithm, it will cost less time than linear scan of these postings list. To use this algorithm, it is crucial that postings be sorted by a single global ordering. Using a numeric sort by docID is one simple way to achieve this.

We can extend the intersection operation to process more complicated queries like: Term_1

AND Term_2 AND Term_3 ... AND Term_i. Here is the algorithm for processing the query:

```
INTERSECT(⟨t_1, ..., t_n⟩)
1  terms ← SORTBYINCREASINGFREQUENCY(⟨t_1, ..., t_n⟩)
2  result ← postings(first(terms))
3  terms ← rest(terms)
4  while terms ≠ NIL and result ≠ NIL
5  do result ← INTERSECT(result, postings(first(terms)))
6     terms ← rest(terms)
7  return result
```

Figure 4: process of intersection of i postings lists

if we start by intersecting the two smallest postings lists, then all intermediate results must be no bigger than the smallest postings list, and we are therefore likely to do the least amount of total work.
And it is easier to do the operations of the OR and NOT. We could transfer all Boolean expressions into the expressions which only contains three opeartors including AND,OR,NOT, then deal with them with these algorithms.