|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Factors/parameters** | **Experiment #1** | **Experiment #2** | **Experiment #3** | **Experiment #4** |
| Preferred block size (*SB*) | Not Applicable | 50, 250 (a typical page), 1000 | 50, 250 (a typical page), 1000 | 50, 250 (a typical page), 1000 |
| Maximum number of blocks (*BM*) | 1 | 50, 100 | 50, 100 | 50, 100 |
| False positive rate (*RFP*) |  |  |  |  |
| Number of query terms (*QT*) | 1, DT(2, 8) | 1, DT(2, 8) | 1, DT(2, 8) | 1, DT(2, 8) |
| Number of words per query term (*QW*) | DW1(1, 2), DW2(3, 8) | DW1(1, 2), DW2(3, 8) | DW1(1, 2), DW2(3, 8) | DW1(1, 2), DW2(3, 8) |
| Number of documents (N) | 500, 5000 | 500, 5000 | 500, 5000 | 500, 5000 |
| Text corpus | {Small Documents, Large Documents } | {Small Documents, Large Documents } | {Small Documents, Large Documents } | {Small Documents, Large Documents } |
| Proximity Scoring | No | No | Yes | Yes |
| Term Weighting | No | Yes | No | Yes |

**Table 3** – summary of the four experiments

NOTES:

(a)

This will consist of 4 experiment types, and 608 variations of the experiments in total.

Let me know what you think, but I’ll be making an automated tester to go through however many parameters we choose. The only question is, how long will it take? Once I construct this tester, I’ll evaluate how long it’s taking and adjust it accordingly.

(b)

Text corpus implicitly defines document size (*SD*). A text corpus will just be a set of documents. I will have the following two text corpuses:

1. A text corpus of small documents. This will not be as demanding on the proximity measure, as the document can be divided into a smaller number of blocks of size *SB*.
2. A text corpus of large documents. These will be very demanding on the proximity scoring.

I will sample N={1000, 10000} from the designated text corpus to make a database, repeats allowed.

(c)

N(5, 2) in *QT* and *QW* is a truncated normal distribution, with a minimum value of 1 and a maximum value of 8. A maximum of 8 seems reasonable as this will allow exact phrases up to 8 words long as query terms. I will construct M queries (M is just some constant that does not change) according to *QT* and *QW*.

When *QT* is 1 no proximity scoring needs to be used, so in experiment #3 I do not explore *QT*=1.

Let me know if this seems reasonable.

(d)

I removed an experiment. The typical way proximity scoring is done: use term weighting to calculate a term weight score. Call this A. Then, use proximity scoring to calculate a proximity score, B. Then, combine these two in the following way: (1-t)\*A + t\*B, where 0<=t<=1. I’m not sure it makes a lot of sense to just do term weighting with the proximity scoring system I have in mind, but I will resolve this question today or tomorrow.

(e)

I am not going to use Lucene; I’m going to use Terrier, which is an academic IR system.

(z)

In experiment number 1, there is less need for segmenting documents into blocks since no relevancy scoring will be used. However, having only a single block for the entire document will (probably) increase the false positive rate on phrase search terms. A phrase search term is decomposed into a conjunction of bigram terms. If all of the bigrams exist in a block, then it is assumed the phrase is present. Of course, it may not be – all the bigrams may be present but not as a chain of bigrams. When the block is extremely large, the probability of this occurring is increased.

On the other hand, large blocks may decrease the chances of false negatives. To save time and simplify things, when doing a phrase term search, currently we just check for the presence of all the bigrams in a block. If a block has all the bigrams then, like stated previously, it is assumed the block contains the bigram. However, if a phrase spans two blocks, said phrase will not be detected by this approach.

Solutions:

1. Put the first K-1 bigrams from block N in block N-1 also. Any phrase that is less than or equal to K terms will thus be found by this approach. The downside is that these K-1 bigrams will be shown as present in both block N-1 and N, which may ironically increase the rate of false positives on phrases and harm the proximity measures.
2. When checking for the presence of bigrams in a phrase consisting of n bigrams, then if block N has bigrams 1, 2, …, k, but does not have k+1, then check block N for k+1, k+2, …, n. This logic can be extended to support phrases that span three or more blocks, but there is not much expected value in that. The important case of checking if a phrase spans two blocks will cover all or nearly all needs.