1. Our Name and UNI

Andrew Shu [ans2120@columbia.edu](mailto:ans2120@columbia.edu)

Ran Bi [rb2651@columbia.edu](mailto:rb2651@columbia.edu)

1. Files that we submit
2. Jave code files
3. Makefile
4. Readme
5. How to run our program
6. Internal design of our project
7. Set up a way to talk to Yahoo! Search BOSS API through Java.
8. Afer the user input the query, there will be a resultset returned from Yahoo! BOSS. If the specific value of presicion is reached, we will stop. Otherwise, we do the query expansion.
9. Return the new resultset to the user.
10. Query-Modification Method
11. According to the feedback of the user, we can divide the original resultset into two parts, relevant and nonrelevant resultsets.
12. Through Apache Lucene, we can find all the terms which appear in the resultset. Then for a given term, we compute its frequency, which is the number of documents in which this term appears. We do this computing for both relevant and nonrelevant resultsets.
13. Set the frequencies of terms in nonrelevant resultset to be negative. Then we combine the terms in two resultsets. If the terms are the same in two resultsets, we add their frequency and return only one term vecor, which consists of termname and frequency.
14. We sort all the terms by their frequencies and pick up top two terms.
15. Add these two terms into the old query. To put them in the best order, we compute the positions for all terms which consist of the new query. Then we pick up the best document which contains the most of terms.
16. At last, we put the terms in the order the same as their natural order in that best document and return it to Yahoo! Search BOSS.
17. yahoo BOSS Application ID

8ZdPHgrV34GqaKRLc2FEMULwfYT9\_rn1xE0swnm6JcfB\_IflTBEdzba7HAuPJcDwGA