a) UNI: hs2762(Hao-Hsin Shih), sw2848 (Siyang Wu)

b) We Implemented this porject by JAVA. A list of all the files:

Bing.java
Doc_format.java
Idf_file.java
Rocchio.java
Setting_constant.java
commons-codec-1.9.jar
jsoup-1.7.3.jar
makefile

c) How to run my program:
Firstly change the directory of /src, then type make (executing the makefile)

Then input the following:
java -cp jsoup-1.7.3.jar:commons-codec-1.9.jar: Bing
dFkQ9zzsXuD5Mvqsb3jyPvIuPYgTzzOGhiB33K9OPLo 0.8 snow leopard

[The format should be:
java -cp jsoup-1.7.3.jar:commons-codec-1.9.jar: Bing ACCOUNT_KEY PRECISION
QUERY_TERM_1 QUERY_TERM_2 QUERY_TERM_3…]

Note: the query terms are all case insensitive.

d) Internal design of my project:

Due to the language we use to develope this project is JAVA, this project is designed as object oriented model .The point for this project is indexing the terms and documents as well as using Rocchio algorithm to calculate the weight of term to choose the greatest term to expand the query. .

Data structures for each document(item fed back by Bing API):
Class Doc_format
[

```
        private String ID;                  \\"1"
        private String Title;               \\"project for ADB"
        private String Summary;             \\"query expanding method..."
        private String Body;                \\not used in  this project, reserved for extend version
                                            \\store the html document paresd from URL.
        private String URL;                 \\"http://..."
        private boolean Is_relevant;                \\"true/false"
        private HashMap<String, Integer>Tf_vector = new HashMap<String, Integer>();
        \\hashtablve with key = term, value = term frequency.
        .
```

```
        othrer function...
]
```
and all document are stored in a list order by document ID.

Data structure for object storing the IDF information.

HashMap<String, HashMap<String, ArrayList<Integer>>> hm

```
[
        “term” : [      //term strign is key of hah table, value is also hash table.
                        “document ID” : [ 0, 99, 105]       //document is key, value is ArrayList
                                                            //ArrayList of posiotions of terms in
                                                            specific
                        .                                   // document
                        .
                        other documents
                ]
        .
        .
        .
        other terms
]
```

Porter stem: we planed to implement port stem method to reduce the noises that actually could be classified to a term. But the experiment came out with a unsatisfying result that Port stem method tend to reduce the performance for some query. For instance, “OS” would be transferred to “O”; “Jobs” from Steve Jobs is transferred to job that doesn’t expand query used to search at all. So it was not applied to this project.

Html parser: If the information in summary isn’t considered as enough amount of data for Rocchio Algorithm to work, then the textual content of the web page seem to be good extra data to extend by. But there still exist lots of problems which could easily eliminate the good effect it comes with. First, the diversity of html tag for different web page which would generate ton of unexpected useless and meaningless tokens which could be seen as totally noises for Rocchio algorithm. Moreover, we are not sure if the texts we get from the web page is relevant information to its title due to ton of commercials. And we decided not to implement this function.

Accumulate scoring for all iteration and resorting of query:

The scoring for terms given by Rocchio algorithm would be accumulated from last iteration so that on the hyperplane we vectorize our model we would greatly appraoch our target point and get away from the irreleavat point (term) every iteration. Furthermore, before giving the new query expanded to Bing API we would resort the order of term in the new query by their scoring weight to make sure that the most important term is always at the beginging of query.

Adding one or two new terms to query for each iteration:

If the top1 scoring term is two times higher than the second term then we just add top1 scoring term to new query to reduce the bias new term may gives. Otherwise, we add top2 highest terms to new query.

e) Query-modification method:

Rocchio Algorithm.

For the Rocchio Algorithm:

$$\overrightarrow{Q_m} = (a \cdot \overrightarrow{Q_o}) + \left(b \cdot \frac{1}{|D_r|} \cdot \sum_{\overrightarrow{D_j} \in D_r} \overrightarrow{D_j}\right) - \left(c \cdot \frac{1}{|D_{nr}|} \cdot \sum_{\overrightarrow{D_k} \in D_{nr}} \overrightarrow{D_k}\right)$$

We set:
a = 1, b = 0.8, c = 0.1

$\overrightarrow{Q_{m*}}$	$\overrightarrow{Q_{o*}}$	$\overrightarrow{D_{j*}}$	$\overrightarrow{D_{k*}}$	a_*	b_*	c_*	D_{r*}	D_{nr*}
Modified Query Vector	Original Query Vector	Related Document Vector	Non-Related Document Vector	Original Query Weight	Related Documents Weight	Non-Related Documents Weight	Set of Related Documents	Set of Non-Related Documents

f) Our Bing Search Account Key:
dFkQ9zzsXuD5Mvqsb3jyPvIuPYgTzzOGhiB33K9OPLo

g) Any additional information that you consider significant.

For each round for expanding the query. We get at most 2 terms, and if one term's score is more than twice as the other term's score, then we only expand by one term, because we think adding too many terms may affect the accuracy of our program.

The order of the terms matter! For the first round, all the query terms we type in will have score 10. For the following iterations, the documents that we analyze will affect the score in the query list(including the query terms in the first round). And after expanding the query, we reorder the list according to their score so that terms with highest score will move to the first of the expanded query. For example, if for the first round, query is "San" and we want to find news about Bay Area, the query

after the first round may then be “Bay Area San” (if the scores of Bay and Area are higher than San so that they jump to the left)