Kevin Li (kl2855) Aliraza Punjani (amp2280)

PROJECT 1 GROUP 12

List of files:

1. README (this file)
2. bingFeedback.py (python source file)
3. Transcript.pdf (transcript of the runs of our program on the 3 test cases ‘musk’, ‘brin’, ‘Taj Mahal’)

Steps to run:

Execute: **python bingFeedback.py <BING_SEARCH_KEY> <TARGET_PRECISION> <QUERY>**

Query should be in double quotes (“”)

BING_SEARCH_KEY: mz6wWvhFVxhgbqlz+aDPIa/V1uaygzWZreeE3L3+7CA

Eg: python bingFeedback.py mz6wWvhFVxhgbqlz+aDPIa/V1uaygzWZreeE3L3+7CA 0.9 “musk”

P.S. transcript.txt will be generated in the same directory

Implementation:

We’re employing following techniques for query reformulation

1. A customized, more practical variant of **Rocchio’s algorithm** to score words
2. Accounting for **Document Zones** by assigning different weights to scores of words appearing in Title and Description
3. A simple variant of a **Stemmer** to group words stemming from similar roots
4. **Stop word elimination**

Pre-processing

The pre-processing step involves tokenization of the relevant and non-relevant text into words; removing spaces and punctuation marks and lower casing them. We carry out stop word elimination here. We also pre-process the Title to get rid of website specific info, since it's found to be not relevant to the phrase queried in most cases.

extractWords() is our pre-processing routine.

Scoring

Scoring is done using a variant of Rocchio's algorithm. We've modified it to account for document zones. Score S of a word is given by the formula,

$$S = \alpha . f_q + \beta_1 . f_{rt} + \beta_2 . f_{rd} + \Gamma_1 . f_{nrt} + \Gamma_2 . f_{nrd}$$

f_q = frequency of the word in the **query**

f_{rt} = frequency of the word in the **title** of **relevant** documents

f_{rd} = frequency of the word in the **description** of **relevant** documents

f_{nrt} = frequency of the word in the **title** of **non-relevant** docs

f_{nrd} = frequency of the word in the **description** of **non-relevant** docs

$\alpha = 1$

$\beta_1 = 1.5$

$\beta_2 = 1$

$\Gamma_1 = -0.75$

$\Gamma_2 = -0.5$

processFeedBack() is our scoring routine.

Stemming

We consider every word in our list as root and remove all words in the list which are derivatives of the root while adding their scores to the root word. We realize, this relatively simple implementation, in some cases, might eliminate words which are not derivatives of other words. However, we found such cases to be rare, especially since we've already eliminated the stop words. On average, we found this functionality improves the performance of our system for the better.

Eg: **robot** *with score* **1.5** and **robots** *with score* **1** gets stemmed to **robot** *with score* **2.5**

stemDerivatives() is our stemming routine.

Sorting

We sort the words in descending order of their scores.

Selection

Finally, we select the top 2 words from the list which are not part of the query already.