# MSCI 541 HW5 Report

Ramandeep Farmaha 20516974

## Problem 1

For an *m x n* matrix *W*, where there are *m* rows of terms and *n* columns of documents, in order to find the terms most similar to a term *t*, we would first have to compute $M = WW^T$. This resulting square matrix shows the relevance of terms using their term weights. To find terms most similar to term *i*, we first find the value of element $M_{ii}$; this, along with all of the other weights on the diagonal, show the relevance of a term versus itself. Thus, it'll be the highest value for a term in its column or row. In order to find the next most similar words, simply traverse either up/down the column *i* or left/right the row *i* and look for the next highest weight value.

## Problem 2

### 2a

The MLE model of the document collection is represented by $P(w|C)$, and an be computed by:

```
P(w|C) = count of word w in collection of docs
         -------------------------------------
         total word occurences in collection
```

### 2b

Since Dirichlet smoothing is a specific instance of Jelinek-Mercer smoothing where:

$$\lambda = 1 - \frac{|D|}{|D| + m}$$

Lambda is the Jelinek Mercer smoothing parameter, while *m* is the Dirichlet smoothing parameter. Lambda and *m* are set to 100 and 0.5 respectively, and thus we can solve for document length $|D|$:

$$\lambda = 1 - \frac{|D|}{|D| + m}$$
$$\lambda(|D| + m) = |D| + m - |D|$$
$$\lambda|D| + \lambda m = m$$
$$|D| = \frac{m(1 - \lambda)}{\lambda}$$
$$|D| = \frac{100(1 - 0.5)}{0.5}$$
$$|D| = 100$$

Thus, documents with a length of 100 will be smoothed exactly the same by the Jelinek-Mercer or Dirichlet smoothing algorithms using the given parameters.

# Problem 3

Smoothing shorter documents more than longer documents makes sense because there is a less chance of the query tokens appearing in a shorter document than a longer one. For example, a short 30-word document about the Mars chocolate bar could contain only phrase "Mars bar". However, a query for "Mars chocolate" wouldn't return the document because although it's relevant, there may be other, longer documents that contain more instances of the word "Mars". Increased smoothing on shorter documents means that less emphasis is placed on the probability of the tokens appearing in a specific document, and more is placed on query tokens appearing in the entire corpus.

# Problem 4

## Query-biased Summary Explanation

In order to compute the query-biased summaries, the document text and graphic text are used for each LA times article. The following pseudocode summarizes the query-biased summary algorithm:

```
Concatenate document text and graphic text
Split on any stops [?!.] that precede either a space or quotation ['"], indicating
the end of a sentence or quote
Remove escaped characters from sentences
Discard sentences that are less than 5 words
For each sentence S:
    l = 2 if S is the first sentence, 1 if it is the second, 0 otherwise
    c = the number of words in the sentence that are query terms, including
repetitions
    d = the number of distinct query terms that are in the sentence
    k = the length of the longest contiguous run of query terms
    score = 2*c + 5*d + 3*k + l, i.e. place a greater emphasis on the query terms
appearing in the sentence and their frequency in the sentence
    NOTE: Since headline is not used in the summarization, the h term as discussed
in class is not used in computing the score
    max_heap.push(score)
max_heap.pop() twice to retrieve top 2 most important sentences
```