PA3 Report

Name(s): Emmanuel S Thomas & Sumit Gupta

Due Date: 05/15/2014

Class: CS276

Professors: Chris Manning & Pandu Nayak

Overview

For this assignment we implemented various ranking algorithms (Cosine Similarity, BM25 and Smallest Window) and benchmarked their performance using parameter tuning for each ranking algorithm. Few key points for our algorithms design:

1) IDFs for each term are computed using corpus data.
2) If stemming flag is enabled, we store both term and its root word to compute IDFs for both. In this case the term frequency for root word will be different than the actual word.
3) Base class Ascorer.java is used to compute raw TF for terms in different zones in document and Query to compute Document and Query Vectors.
4) Porter Stemmer algorithm is used for stemming (http://tartarus.org/martin/PorterStemmer/).
5) Cosine Similarity uses
    a. Sub-linear scaling for both Query and Document Vectors
    b. La-place smoothing is used for IDFs
    c. Normalization is used only for Document Vectors

Task 1: Cosine Similarity Score:

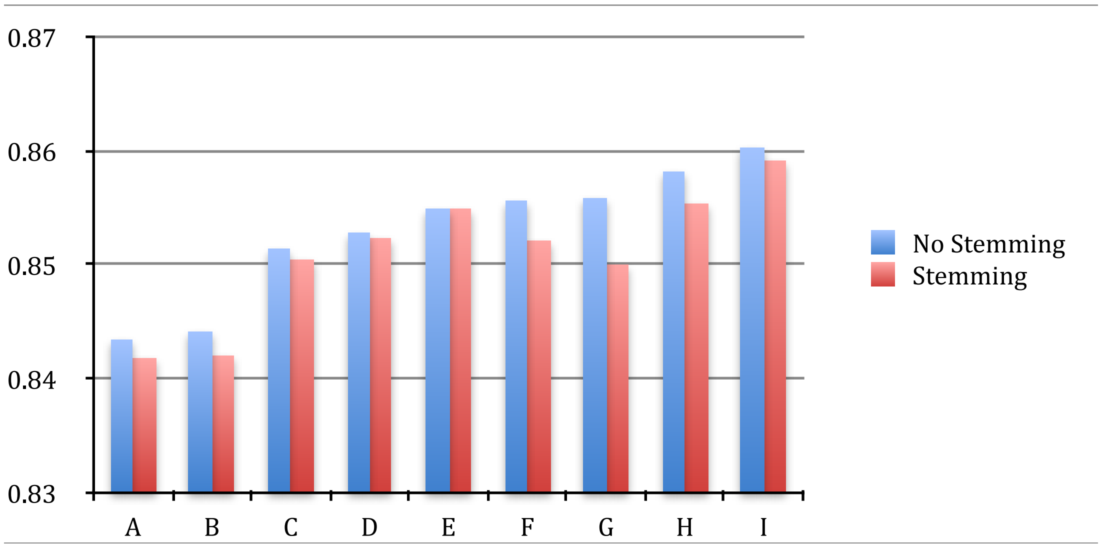
During experiments we observed the following:

1) Anchors should have the highest weights as they are the most relevant as more the other documents point to this document the more relevant this document will be.
2) Title and body of the document are the next factor for relevance, as they reflect the actual contents matching the query. For new documents added in the corpus the anchors will build up over time but still they can be relevant and important w.r.t. Query.
3) Url and the header might not even have terms from the query.
4) As the average document length is 1000 words per doc, it worked well with our experiments.

Smoothing Body Length = 1000

Configuration	task1_W_title	task1_W_anchor	task1_W_url	task1_W_header	task1_W_body	NDCG Score Stemming (OFF)	NDCG Score Stemming (ON)
A	0.7	0.5	0.7	0.5	0.9	0.8434358044463822	0.8418796038114148
B	0.9	0.5	0.7	0.5	0.9	0.8441296523293544	0.8419657766646911
C	0.9	0.8	0.7	0.5	0.7	0.8513226937963415	0.8505141382302487
B	0.7	0.5	0.7	0.5	0.3	0.852843258072535	0.8522564729368541
E	0.9	0.8	0.7	0.5	0.3	0.8549393428353194	0.8548264227242311
F	0.3	0.5	0.7	0.5	0.3	0.8556805094411905	0.8521853065148774
G	0.3	0.5	0.1	0.5	0.3	0.8559162836626797	0.8500909537476006

H	0.3	0.8	0.1	0.5	0.3	0.8581345664736532	0.8552621153983023
I	0.3	0.9	0.1	0.1	0.3	0.8602954983044914	0.8591685996952332



The other metrics that were not used in this assignment that could potentially increase overall relevance score are:

1) **Authoritativeness** – Boosting the scores using the source (url) of information. Giving more weightage to the documents from the trusted information store like Wikipedia, News Websites, official homepages, etc.
2) Implementing the **Phrase queries** instead of treating each term independent.
3) **URL sharing** on social networks can be another signal that can be used to boost document's ranking proportional to number of shares (sub-linear scaling). Tweets and/or Facebook shares can have more weights than bitly. Some sort of logic is required to distinguish between shares and spams.
4) **Time sensitivity** can be added to score down the documents that are older as compared to the new ones. Example: Reddit has a good implementation for the new stories on the web.
5) Using other schemes from SMART can be used to benchmark our ranking algorithms.

Task 2: BM25F Score:

Parameter Tuning

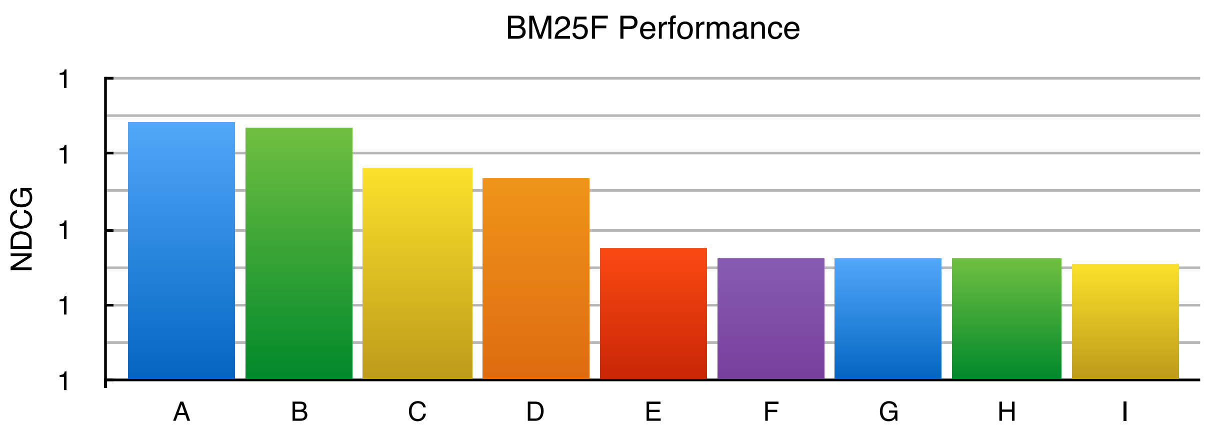
The logic behind estimating the values was simple. We looked at the average length of the field and tried the weight the shortest the higher. By intuition, the smaller the field, less room there will be for noise. That's why the body was ranked the lowest in our model. We ran extensive testing and came up with the values below:

task2_W_url: 10.0
task2_W_title: 2.0
task2_W_body: 1.0
task2_W_header: 8.0
task2_W_anchor: 7.0
task2_B_url: 0.4
task2_B_title: 0.2
task2_B_header: 0.5
task2_B_body: 0.1
task2_B_anchor: 0.1
task2_k1: 50.0
task2_pageRankLambda: 1.0
task2_pageRankLambdaPrime: 1.0

Optimization

One other variable that we could use would be the reachability of the document. The harder it is to find a document, the less likely it would be relevant. However we were not able to implement this due to lack of time.

In BM25F we chose log(lambda' + page rank). The reason is simple; we assumed that the pagerank was pretty significant



value. Even if this is not necessarily the case in real life, it turned out pretty well for us.

Task 3: Smallest window Score:

We tried to implement the smallest window algorithm based on the bm25f. However, we were not able to achieve improvements no matter our choice of parameters. This is probably due to errors in our logic. More testing and debugging would be needed to determine.

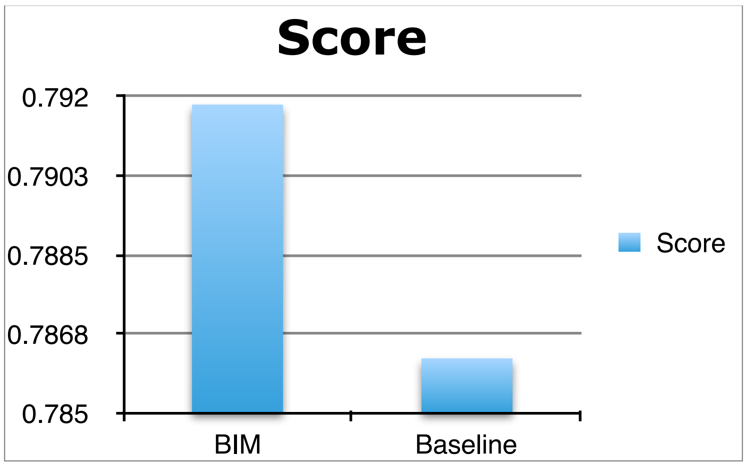
Extra Credit: Binary Independence Model (BIM):

Here we tried to implement the BIM algorithm, where we calculate score by summing IDFs for the matched terms in the document (any zone level). We ignore the TF for a term and just check if term exists in the document or not (matched with the Query) and compute IDF for each matched term and taking the SUM of IDFs as a final Net Score for ranking.

In this case we make following assumptions:

P(i) = 0.5 (50 % chances for a term to occur in the relevant documents)

RSV = Σ log (N/n(i))) for all matched terms. (Assuming relevant documents are significantly less as compared to the total number of documents in the corpus. Then we can boil down RSV to the above equation.)


Score
BIM
Baseline
Score