Muhammad Owais Bawany -- 7993647

Felix Singerman -- 7970742

Tweet IR System

We implemented an Information Retrieval (IR) system based for a collection of documents (Twitter messages). The collection is a subset of the data used in the TREC 2011 Microblog retrieval task. Each line contains the id of the Twitter message and the actual message. We completed 5 main steps.

1. **Preprocessing**: We implemented preprocessing functions for tokenization and stopword removal. The index terms will be all the words left after filtering out punctuation tokens, numbers, stopwords, etc.
2. **Indexing**: We built an inverted index, with an entry for each word in the vocabulary.
3. **Retrieval and Ranking**: We used the inverted index (from step 2) to find the limited set of documents that contain at least one of the query words. We computed the similarity scores between a query and each document
4. **Results file:** We ran our system on the set of test queries. We included the output in our submission as a file named Results.The file has the TREC format. We included the top-1000 results for each query.
5. **Evaluation:** For evaluation, we used the trec\_eval script. The main evaluation measures was MAP (mean average precision) and P@10 (precision in the first 10 documents retrieved). We compared our results with the expected results, from the relevance feedback file. Evaluation results are found in a file named evaluation.

Installation:

* Edit the doit.sh script, and the config file and set the correct path to your download of the repository.
  + data-path: path to the tweet data (Trec\_microblog11);
  + index-path: a new directory that will hold the contents of the index;
  + stop-words: the stop words file (stopwords.txt);
  + icsv: the queries (topics\_MB1-49);
  + ocsv: new file that will contain the results of the IR system;
  + tweet\_fuzzy: the path to the jar.
* Run the doit.sh script. `./doit.sh`
  + Doit runs the indexing, searching, then workflow
* You can also run the program manually by running the Indexer.java, followed by Workflow.java scripts with the appropriate command line arguments (Ex: --icsv path/to/directory --data-path /path/to/directory). You can also run a manual query that is not listed in the query doc by running Indexer.java followed by Searcher.java and entering the query in the command line. The results you will receive will be outputted to the command line in JSON format.

We made the IR system using Lucene-6.5.1 -- a free and open sourced information retrieval library.

* Indexer.java: the indexer, and preprocessor.
  + The indexer uses the stop-words document provided in order to omit these common words. Lucene’s built-in Standard Analyzer is used, which builds an analyzer with the given stop words. An Ngram Analyzer is used for the tweets.
  + Lucene’s built-in IndexWriter is used in order to create the index and maintain it. It does its own tokenization. The documents (tweet info) are added to it. Documents are the unit of indexing and search. A Document is a set of fields. Each field has a name and a textual value. Lucene also provides its own stemming on the documents, however we did not use it as it is not proven to increase results. Due to time constraints we do not do any normalization.
  + The tweet is indexed using Lucene’s TextField which is used on a field that is indexed and tokenized, without term vectors and is stored in index. For example this would be used on a 'body' field, that contains the bulk of a document's text. The tweets are also indexed using an Ngram Analyzer.
  + The vocabulary was a large collection of tweets. There was 45899 tweets which have a 120 character limit. The algorithm indexed them in 4742 msecs.
  + The format of the indexed documents is almost unreadable. I have posted an example here for reference (I will not include 100 since it is mostly gibberish): Kids Fashion Agency gaat Peaple vertegenwoordig0Het7 Little Known Secr§T\*W%AMake-Òa Guy Magnet Shes better known for things that she does on the mastress!NP/ P(†Û`0589529088àCentral New York Creating Wellness Expo in Oneida highlights local health ...: Once people try one `y food&Û Guys was pretty delicious,vÒway too expensive. DefinitelyÑ0getÜP frieJ!xt¥s, only πÅa burgerÉ§peanuts.ì'ó/ ñm,0ƒ968942592mMy i y‡really weren't00mak¨Äyone ups®îêsorry if Fêtopic cam#≠Qa bad-<¿your life.Ä É/ ÉZ7998728208384?Red≈Pte Mo/\*
* Searcher.java: the algorithm for searching the indexed documents against a query.
  + The searcher runs a single query at a time against the indexed words. We have provided a command line option to run custom queries.
  + Our search uses an LciEditDistance metric with an ngram overlap.
  + We use the Lucene TopDocs in order to store our results returned by IndexSearcher
  + Lucene uses a built in TF-IDF scoring. We then ran our modified scoring based on that result. In the JSON output, you can see both Lucene’s score, and our modified score.
* Workflow.java: the algorithm for parsing the queries document and return a results file in the proper format.
  + Calls the Searcher.java in order to run the searching on each given parsed query.
  + Prints out the relevant information in the required format for analysis with TREC.
* Config.java: code for reading from the command line (recycled code used in past project)
* AnalyzerUtils.java: Lucene code taken from guidelines for the analyzer.
* LciEditDistance.java: Algorithm for the LciEditDistance. Help from online resources.
* NgramAnalyzer.java: Algorithm for performing the Ngram Analysis. Help from online resources.

Top 10 results to query 1: BBC World Service staff cuts

1. RT @davelength: Anyway, while Twitter goes wild about Andy Gray, a quarter of the BBC World Service staff gets laid off and nobody notices.
2. Quarter of BBC world service staff to go, uk foreign office grant reduction of 17.5%.
3. A statement on the BBC World Service, ahead of staff briefings/ further details on Weds http://bbc.in/dFfXIW #bbcworldservice #bbccuts
4. BBC World Service to cut [...] a quarter of its staff - after losing millions in funding from the Foreign Office. http://bbc.in/hyGSHi
5. BBC World Service forecast to lose 30m listeners as cuts announced http://gu.com/p/2mkqh/tf
6. BBC World Service plans 650 job cuts (AP) - AP - The BBC said Wednesday that it plans to cut 650 jobs, more tha... http://ow.ly/1b2u20
7. BBC world service faces 650 job losses and it get a prime slot on BBC news. What about the larger cuts in many local authorities! #notfair
8. BBC World Service outlines cuts to staff http://bbc.in/f8hYAT
9. BBC News - BBC World Service cuts to be outlined to staff http://www.bbc.co.uk/news/entertainment-arts-12283356
10. Save BBC World Service from savage cuts http://www.petitionbuzz.com/petitions/savews

Top 10 results to query 25: TSA airport screening

1. TSA shuts door on private airport screening program (CNN) <http://feedzil.la/gD1tt6>
2. TSA shuts door on private airport screening program. Utter BS! - http://bit.ly/fx6Dgw #cnn
3. TF - Travel RT @Bitter\_American TSA shuts door on private airport screening program - http://bit.ly/fx6Dgw #cnn:... http://bit.ly/eADg2G
4. TSA Shuts Door on Private Airport Screening Program – Patriot Update http://patriotupdate.com/2451/tsa-shuts-door-on-private-airport-screening-program?sms\_ss=twitter&at\_xt=4d45868911137f91,0 … via @AddThis
5. Breaking #news #tcot Jesse Ventura takes on airport screening pat-downs in lawsuit: Former Minnesota governor ... http://twurl.nl/c0rene
6. TSA Shuts Down Private Airport Screen Program is headline now on www.fedsmith.com.
7. TSA chief: airports won't hire private screeners (Associated Press): Share With Friends: | | Top News - To... http://feedzil.la/e7oo6n
8. #TSA denies airports right to private screeners, prefers #SecurityTheater over effectiveness: http://bit.ly/e02tIs #OptOut
9. Up to 31 killed in Moscow airport attack. Police seeking 3 men, according to report http://wapo.st/eA61BZ
10. ロシアの空港で爆弾爆発 ... 人々をおかしく ... a bomb exploded on Russian airport today ... freaking people who made it ... do not understand it at all

We ran `./trec\_eval -q -m num\_q -m map -m P.10 ../Trec\_microblog11-qrels.txt ../Results.txt >> ../Evaluation.txt` in order to get these following results.

Mean Average Precision (MAP) Scores:

**Map Num Score**

map 1 0.4419

map 2 0.0329

map 3 0.5468

map 4 0.2455

map 5 0.0806

map 6 0.2423

map 7 0.3373

map 8 0.1867

map 9 0.5674

map 10 0.0267

map 11 0.2663

map 12 0.2563

map 13 0.3592

map 14 0.0000

map 15 0.0024

map 16 0.5000

map 17 0.3840

map 18 0.2500

map 19 0.3144

map 20 0.5566

map 21 0.2892

map 22 0.4008

map 23 0.0993

map 24 0.2185

map 25 0.0219

map 26 0.2463

map 27 0.0607

map 28 0.4313

map 29 0.1413

map 30 0.2401

map 31 0.5902

map 32 0.0672

map 33 0.0269

map 34 0.1570

map 35 0.5986

map 36 0.4279

map 37 0.3579

map 38 0.1787

map 39 0.0769

map 40 0.1475

map 41 0.2024

map 42 0.0006

map 43 0.5006

map 44 0.0446

map 45 0.0652

map 46 0.3505

map 47 0.0239

map 48 0.0459

map 49 0.5000

map all 0.2471

Precision @ 10 Scores:

**P\_10 Num Score**

P\_10 1 0.4000

P\_10 2 0.1000

P\_10 3 0.5000

P\_10 4 0.5000

P\_10 5 0.0000

P\_10 6 0.2000

P\_10 7 0.7000

P\_10 8 0.6000

P\_10 9 0.9000

P\_10 10 0.0000

P\_10 11 0.2000

P\_10 12 0.1000

P\_10 13 0.5000

P\_10 14 0.0000

P\_10 15 0.0000

P\_10 16 0.1000

P\_10 17 0.4000

P\_10 18 0.1000

P\_10 19 0.8000

P\_10 20 0.9000

P\_10 21 0.5000

P\_10 22 0.7000

P\_10 23 0.0000

P\_10 24 0.6000

P\_10 25 0.0000

P\_10 26 0.6000

P\_10 27 0.2000

P\_10 28 0.4000

P\_10 29 0.3000

P\_10 30 0.3000

P\_10 31 0.4000

P\_10 32 0.1000

P\_10 33 0.1000

P\_10 34 0.2000

P\_10 35 0.6000

P\_10 36 0.4000

P\_10 37 0.9000

P\_10 38 0.3000

P\_10 39 0.1000

P\_10 40 0.3000

P\_10 41 0.2000

P\_10 42 0.0000

P\_10 43 0.6000

P\_10 44 0.0000

P\_10 45 0.2000

P\_10 46 0.5000

P\_10 47 0.0000

P\_10 48 0.2000

P\_10 49 0.1000

P\_10 all 0.3224

The work was divided with Felix doing part 1, 2, and 3. Muhammad doing part 4, and 5.

We are happy with the results we received through our IR system. Most of our P@10 are good. We would need to further investigate a few results that are 0.00. We were happy with the results for Q1 and Q25 as both were relevant. It was a good learning experience, Lucene was a very helpful library and we are glad we chose it. We know more about information retrieval systems now, from pre-processing to indexing and then using inverted index to retrieve documents that are most relevant and ranking accordingly. We learnt how to apply our lessons from class into a real world project. In the future we would like to try Stemming, using machine learning, and using more context sensitive spell correction.

In this assignment we made an information retrieval system for text documents but in the future we would like to work on a project where we can retrieve voice notes/documents, text to speech etc. This was a challenging but fun assignment.