Building a Search Engine: Part 2
Report
Jesse Geman and Ken Noh

1. If a word appears in every document the score will equal zero. Log(1) is zero.
2. Our inverted index is a dictionary; each key in the dictionary is a token obtained from the title or text tag of the XML file (filtered and stemmed), and each value for that key is also a dictionary. These 'sub' dictionaries, the values corresponding to the tokens, have page ids for keys (the pages the token showed up in) and lists of locations (the location of the token in the stream obtained from that page) for values. The structure of our inverted index allowed us to use json to dump it into a text file that was easily retrieved in our query program in the same format (dictionaries in dictionaries). In addition, Python dictionaries use hashing so everything is of order O(n).
3. Out create function iterates through each term, t, of each page, in the given corpus stemming and filtering where applicable. It then iterates over each term, t, again two more times, once when it creates the inverted index, and then again when it computes the idf scores (though here it only iterates through the set of words). Thus each term is iterated over and acted on multiple times when create.py is run, hence the program is of order O(t). The space complexity is order O(dt) where d can be thought of as the average number of elements in each dictionary for term t.
4. We would do it piecemeal, load in a bit of it at a time, and calculate the cosine similarity for the pages in that chunk, and then load in the rest of the index.
5. The tf-idf method for ranking search results is useful for two big reasons: one it allows for preprocessing, two, it takes into account word frequency in both the document and the overall corpus and gives more weight to rarer words. Accounting for NOT's would be simple, first would generate the documents that meet the query (and DO NOT contain the specified words). Then you would remove the "NOT words" from your query vector and proceed as as normal using tf-idf and cosine similarity.
6. It's hard to find queries that generate bad results. For one word queries (where we used the '-v' flag) our search engine returns every document that contains the query and they all are given the same score, 1.0. Obviously, this is no good, we should still somehow rank the pages in terms of relevance.

<u>Bad Free Text Query:</u>
Bornomala Partakes in The fair
"User:Bornomala" 0.6545551086171758
"User talk:Bornomala" 0.6545551086171758
"Sheriff" 0.6272313000941216
"File:Announcement Trailer – Disney Infinity 3.0 Edition" 0.6272313000941216
"File:Inside Out Play Set - Disney Infinity 3.0 Edition" 0.6272313000941216
"Pixar Wiki:Copyrights" 0.42206457558695337"...

Good Free Text Query:
Dory Nemo Finding
"Finding Nemo Home Video" 0.9914944740209516
"User:Pizzahut101" 0.9910370162122675
"File:Finding nemo dory marlin angler fish.jpg" 0.9910370162122675
"File:Finding-nemo-dory-squishy.jpg" 0.9910370162122675
"File:Marlin-and-Dory-finding-nemo-1003067 1152 864.jpg" 0.9910370162122675
"File:Finding nemo dory marlin swallowed.jpg" 0.9910370162122675
"File:Dory-white.jpg" 0.9910370162122675

Rare One Word Query:
Bornomala
"User:Bornomala" 1.0
"User talk:Bornomala" 1.0

Popular One Word Query
Dory
"Pixar Wiki" 1.0
"Finding Nemo" 1.0
"John Ratzenberger" 1.0
"Andrew Stanton" 1.0
"Brad Garrett" 1.0
"Bob Peterson" 1.0
"Albert Brooks" 1.0
"Marlin" 1.0
"Ellen DeGeneres" 1.0
"Willem Dafoe" 1.0
"Austin Pendleton" 1.0

Good Phrase Query
"Finding Dory"
"Finding Dory Credits" 0.9999344849349667
"Hank (Finding Dory)" 0.9997725616556139
"File:Finding Dory Charlie.png" 0.9997725616556139
"File:Finding Dory Jenny.png" 0.9997725616556139
"File:Happy Mother's Day from Finding Dory! - In Theatres in June 17 in 3D!"
0.9997725616556139
"Marine Life Interviews" 0.9997725616556139
"User talk:Rigby3000" 0.9994162533776751

Bad Phrase Query (was hard to find)

"talk adult humor"
"Talk:Adult Humor" 0.5361773495312574
"User talk:Bald pig cool" 0.442520495442415

Good Bool Query
Dory and Nemo
"Adult Humor" 0.9997920175427881
"Seagulls" 0.9995037324841063
"East Australian Current" 0.9980321368002423
"Talk:Finding Dory" 0.9980321368002423
"Chum" 0.9968691634569693

Bad Bool Query
(Zannabanna OR shakespeare) AND (TOY)
"User:Zannabanna" 0.5255028511771326
"Franklin" 0.44930755571230246
"User talk:RaptorWiki" 0.3242949687331056
"Buttercup" 0.2527166624439541