**CSI \_ FINAL\_REPORT**

### PREPARED BY

Abul Qaism - 8146981

Abdullah Baig – 7266683

## Dealing with Reuters Dataset

We were able to process all of the Reuters documents without too many problems through the use of Beautiful Soup 4 library. But we did have execution time issues, especially when creating the thesaurus. After optimizing the code were able to get it down to running in just one night. When it came to dictionary generation, we didn’t seem to have issues with run time, the only thing that took a little longer was getting the cosine similarities of documents. When running our code for cosine similarities it first returns the intersection of all sets in the list sets. Requires that the list sets contains at least one element, otherwise it raises an error. This resulted in a very large list. Finally cosine similarity is done between query and document.

## Query Expansion

In this module we generated a thesaurus using the Reuters documents. To do this we used K-nearest neighbours. This was super difficult as there are so many words, we were forced to leave our computer on for an entire night as it compared every word in every document to every other word in every document. Our code algorithm for this is running in n3 time.

## Boolean Retrieval Model

Figure 2 Boolean Retrieval of 'oil AND profit'

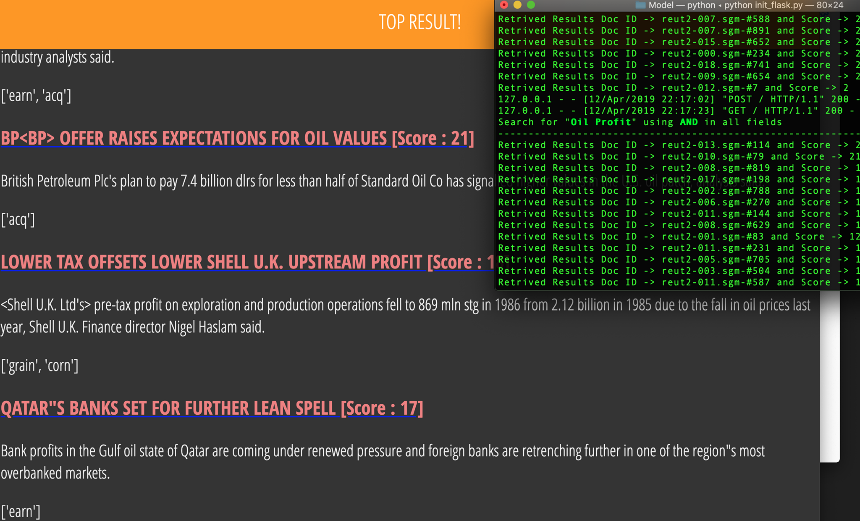
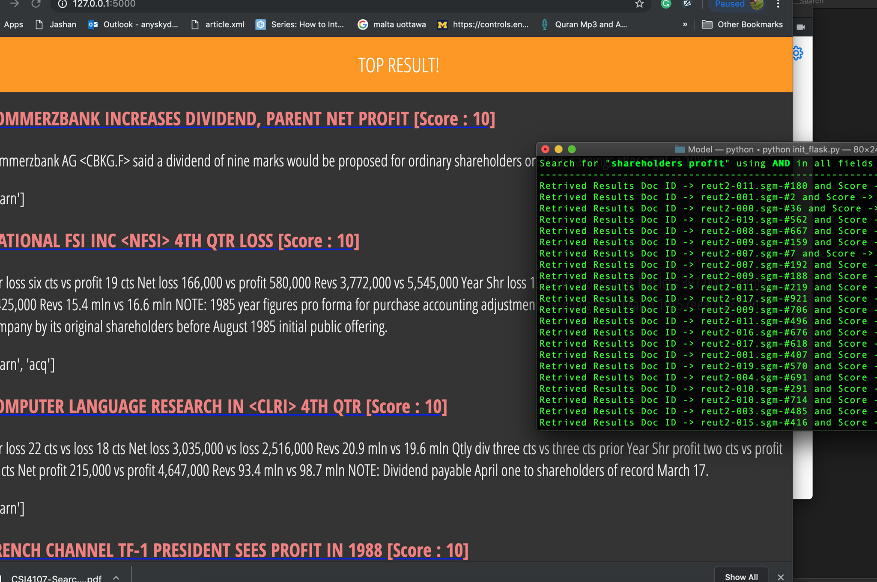


Figure 3 Boolean Retrieval of 'shareholders AND profit'

## Vector Space Model

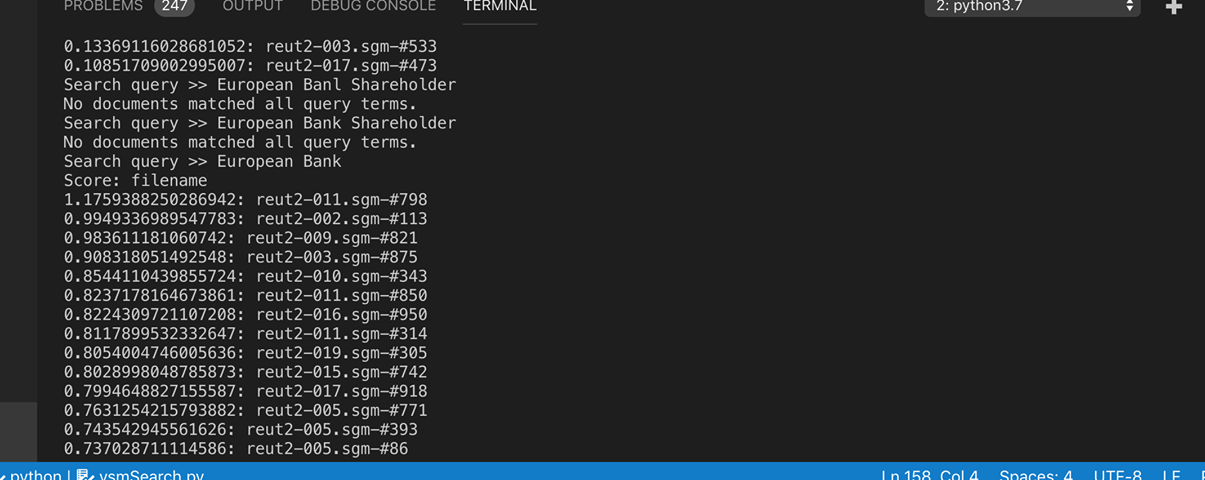
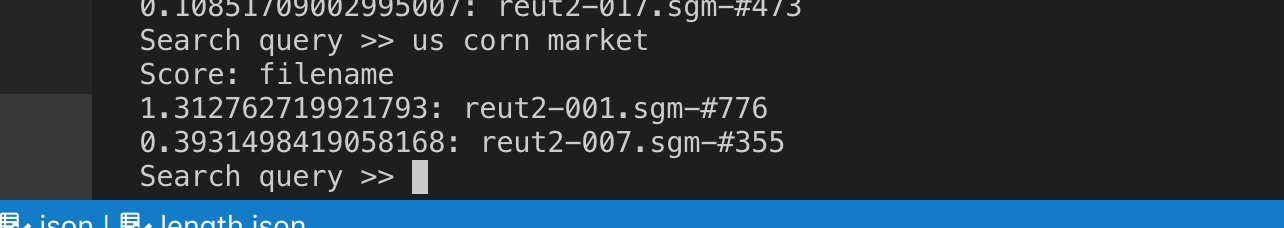
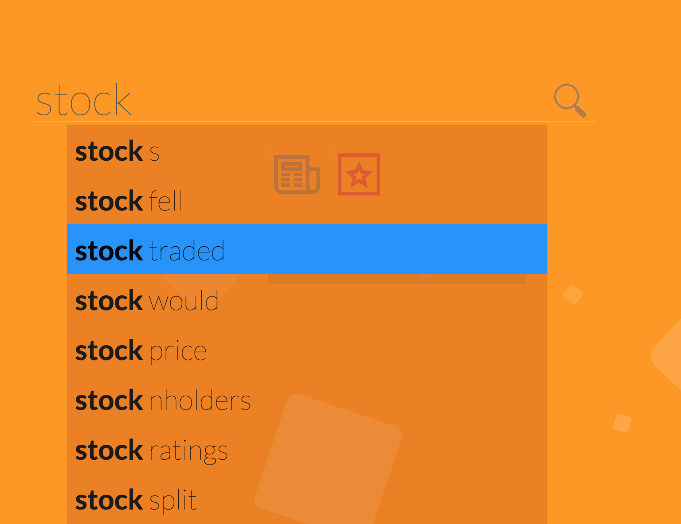
 

Figure 4 Vector Space Model Search Results on Reuters

## https://scontent.fymy1-2.fna.fbcdn.net/v/t1.15752-9/56942432_331846654183630_6504184063723569152_n.png?_nc_cat=111&_nc_ht=scontent.fymy1-2.fna&oh=d14b32f7d47c805ad843237f2e170cd9&oe=5D487C0CBigram Language Model and Query Expansionhttps://scontent.fymy1-2.fna.fbcdn.net/v/t1.15752-9/57382252_339665510068660_3348658251183947776_n.png?_nc_cat=105&_nc_ht=scontent.fymy1-2.fna&oh=b564484c036616a76a7d549d5bceb9e1&oe=5D2DA14A



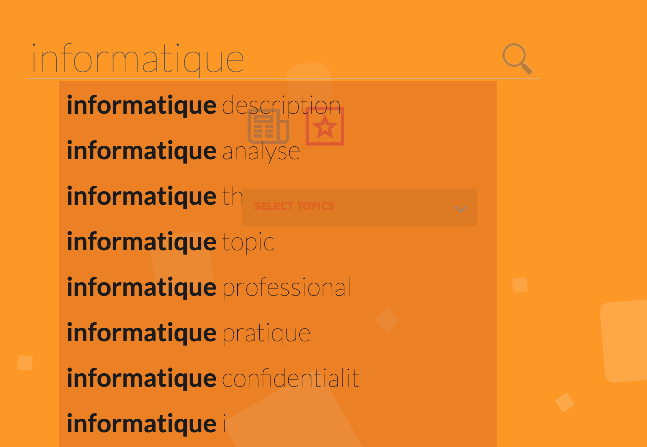


Figure 5 Example of top results for 'oil', ‘coffee’, ‘stock’, ‘police’, and ‘informatique’ using Bigram Language Model

## Topic Classification

We built a dictionary that had set of id with similar topics and then assigned topics to document with highest similarity score. For K we used the size of the training set. We found that k-nearest neighbor was not an effective method for classification as every single word we had to wait over 55 seconds. But at the end classification seemed to make some sense but was not perfect.

"doc\_id": "reut2-011.sgm-#214", topics are "money-fx", "coffee", "dmk", "dlr" title is "BROKERS SAY TOKYO STOCKS WILL RESUME RISE"

"doc\_id": "reut2-013.sgm-#54", : topics are "coffee", "acq", "earn" title is "UK SEEKS PACT WITH JAPAN FOR INFORMATION EXCHANGE"

"doc\_id": "reut2-000.sgm-#831", topics are "livestock", "earn", "acq", "hog" title is "GM <GM> OUTPUT FELL LAST MONTH"

"doc\_id": "reut2-021.sgm-#544", topics are "acq", "gold" title is "GM <GM> OUTPUT FELL LAST MONTH" doc 19015: topics are "interest", "money-fx" title is "BANK OF JAPAN TO SELL 800 BILLION YEN IN BILLS"

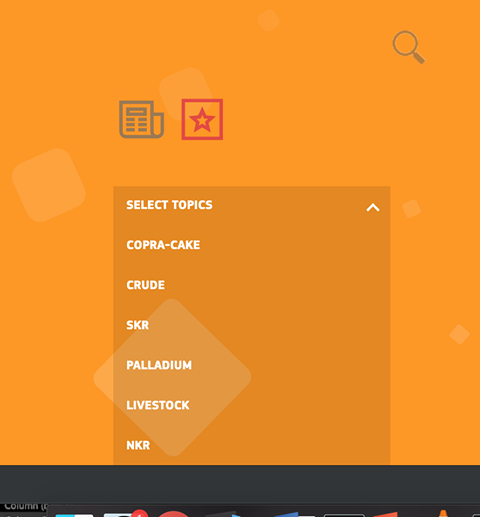


Figure 6 Topics Classified through the use of K-nearest Neighbor

## Thesaurus Visualization

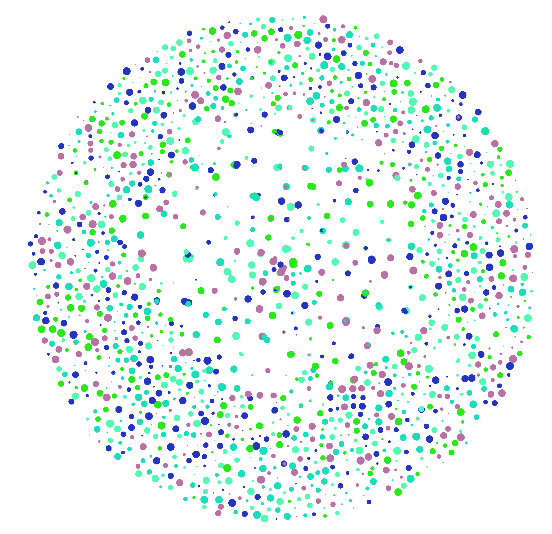
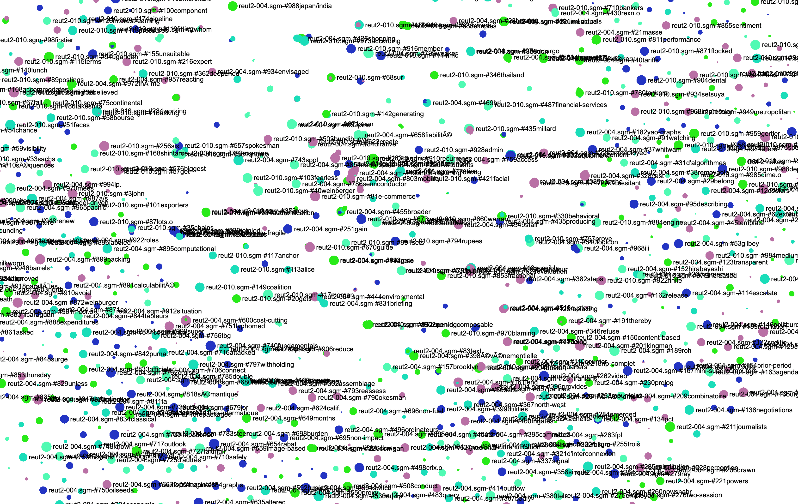


Figure 1 Visualization of Automatic Thesaurus using Java Script Sigma-js

*Dealing with Reuters ○ Were you able to process all the files? If not, why? What caused problems? How many documents did you end up with? Yes, we had no problem we used bs4*

*○ Did you have any execution time issues? (searches being too long for example). If yes, what did you do? Yes, see topic classification dn query expansion took a lot of time*

*○ How long does it take to generate the dictionary? Did you set up some constraints to make the dictionary generation faster? Dictionary was fast but it took around 15mis to get cosine similarities of document and queries*

*○ In general, describe how more challenging it is to work with the Reuters collection than with the CSI collection that was used for the vanilla system. yes the text categorization and knn took really long time to build and process*

# **References**

## Stop word removal

<https://stackoverflow.com/questions/46501481/my-function-remove-stopwords-removes-every-stopwords-in-a-word>

## Bigram

<https://www.tutorialspoint.com/python/python_bigrams.htm>

<https://stackoverflow.com/questions/21844546/forming-bigrams-of-words-in-list-of-sentences-with-python>

<https://gist.github.com/lizadaly/7071e0de589883a197433951bc7314c5>

<https://www.programcreek.com/python/example/86315/nltk.bigrams>

## VSM

<http://blog.josephwilk.net/projects/building-a-vector-space-search-engine-in-python.html>

<https://github.com/mnielsen/VSM/blob/master/vsm.py>

<https://github.com/IvoGoman/Vector_Space_Clustering_and_Link_Analysis>

## Boolean model

<https://github.com/MrAhmedSaeed/Boolean-Information-Retrieval-Model->

<https://github.com/itsnavneetk/booleanRetrieval>

<https://github.com/Erkanamen/CS3245-InfoRetrieval>

<https://stackoverflow.com/questions/19179706/boolean-retrieval-model-in-python>

## Global Query Expansion (with Wordnet)<https://stackoverflow.com/questions/13226629/wordnet-query-expansion-step-by-step>

<https://en.wikipedia.org/wiki/WordNet>

<https://pythonprogramming.net/wordnet-nltk-tutorial/>

<http://www.nltk.org/book/>

## Visualization

<https://www.visual-thesaurus.com/wordnet.php>

## Text Classification

<https://towardsdatascience.com/text-classification-using-k-nearest-neighbors-46fa8a77acc5>

<https://link.springer.com/chapter/10.1007/978-3-642-25734-6_142>

<https://www.datacamp.com/community/tutorials/k-nearest-neighbor-classification-scikit-learn>

<https://www.python-course.eu/k_nearest_neighbor_classifier.php>

## Flask

<https://blog.miguelgrinberg.com/post/the-flask-mega-tutorial-part-i-hello-world>

<https://www.youtube.com/watch?v=lj4I_CvBnt0>

<https://www.fullstackpython.com/flask.html>