CSI 4107 Search Engine

Final System

April 10th 2020

Group 15

Armin Zolfaghari
8715116

This project is already deployed on Heroku, so there is no need to compile the project. The project utilizes Django to create a web application where the scripts folder pregenerates processed data, index and dictionary that the project will utilize while live.

https://one-search.herokuapp.com

Note that Heroku puts the server to sleep after 30 minutes of inactivity, so first accessing the site may take up to 20 seconds.

It is to note that running any of the files in the *scripts* folder would require the virtual environment as described in the vanilla project.

Changes from Vanilla Project

Based on suggestions made two key things were modified:

- A tooltip was added to VSM search results that display VSM scores when hovering above title of result
- ID's of documents were changed to integers from strings

Also: Wildcard queries were not changed even though it was noted that the search results were not as expected. This is because as explained in the Vanilla report that this was happening because of the stemming of the word psychology and on the wildcard word so this was expected behaviour.

The project also now shows word weight for each cleaned word (stemmed, normalized…) in a VSM query, and the query for a boolean search at the top of the results page.

Working with the Reuters Collection

Much of the code was very easily modified to accomodate for the Reuters collection. The code was modified to place all the processed JSON files in the *processed* folder under the respective collection (documents, dictionary, index, bigrams). This means that all prior functionality from the vanilla system also works on the final system, with the same assumptions.

The *preprocess_reuters.py* script successfully processed all 21 Reuters (reut2) files and generated 19043 documents (IDs 0 - 19042). There was an issue in *reut2-017.sgm* there was an error caused by the ü character which was manually deleted:

```
UnicodeDecodeError: 'utf-8' codec can't decode byte 0xfc in position
1519554: invalid start byte
```

There were execution time problems at first as there were also memory problems when loading the index. With the old way the index was done, the index was 202.3MB which is too big for GitHub to even upload (GitHub has a max file size of 100MB). This was resolved by reducing how many characters were actually used in the index. The key “document” in the dictionary was changed to “docs”, “doc_id” was changed to “id”, “frequency” was changed to “tf”, and precalculated tf-idf scores were completely removed. All the indentation was also removed from the file which made it difficult to read and load in a text file, but all of that combined reduced the file size to 34.1MB. Overall the dictionary and index generation for the Reuters collection take less than 10 minutes (*scripts/dict+index.py*).

Before Example:

```
"adm": {
      "documents": [
            {
                  "doc_id": 1,
                  "frequency": 1,
                  "tf-idf": 0.49633151327271724
            }
      ],
      "idf": 0.49633151327271724
}
```

After Example (but without indentation):

```
"adm": {
      "docs": [
            {
                  "id": 1,
                  "tf": 1
            }
      ],
      "idf": 0.49633151327271724
}
```

Overall, it was only slightly more challenging to develop for the Reuters collection because it was much more difficult to check if the results were correct manually with such a large index.

Bigram Model + Query Completion (Modules 1 & 2)

The bigram model is generated with *scripts/word_bigrams.py* and finds all the word pairs in the collections (title and body) then stores them in *word_pairs.json* in the respective processed collection folder. To limit the size of the bigram model, a couple key restrictions were added:

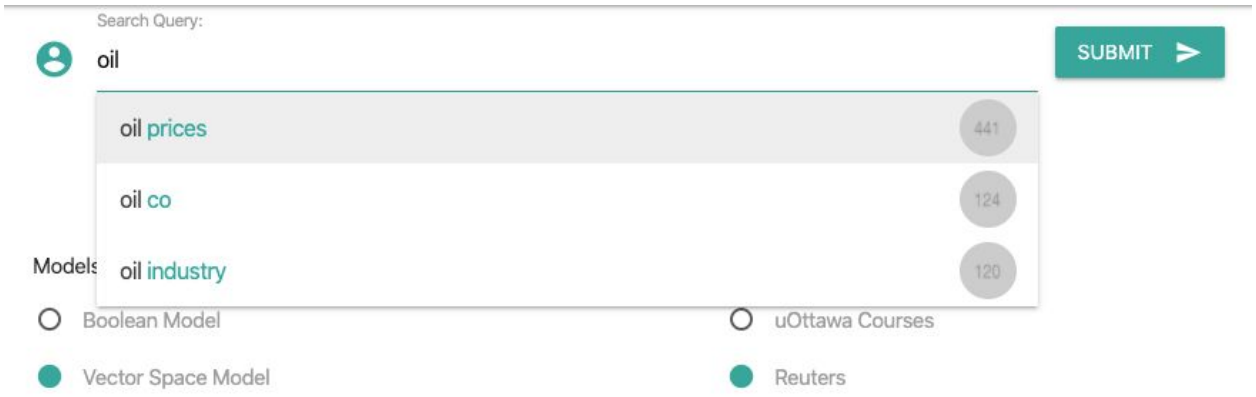
- The first word must have occurred at least 5 times throughout the collection
- Both the first and second word must be more than one character and neither can be stopwords
- Only the top 3 bigrams of each word will be stored

The query completion module then displays suggestions in the view for the next word based on the prior word when the space bar is clicked. This functionality only works when the VSM model is selected in the view, and the results are dependent on the collection (e.g. stock will have no results if the courses collection is selected). The user can select any of the suggestions and the textbox will be filled. The number of times the bigram has occurred in the collection is also displayed on the right hand side in the circle image.

Query Completion Results with *coffee* input for Reuters collection

Query Completion Results with *stock* input for Reuters collection

Query Completion Results with *oil* input for Reuters collection



Query Completion Results with *continued* input for Reuters collection

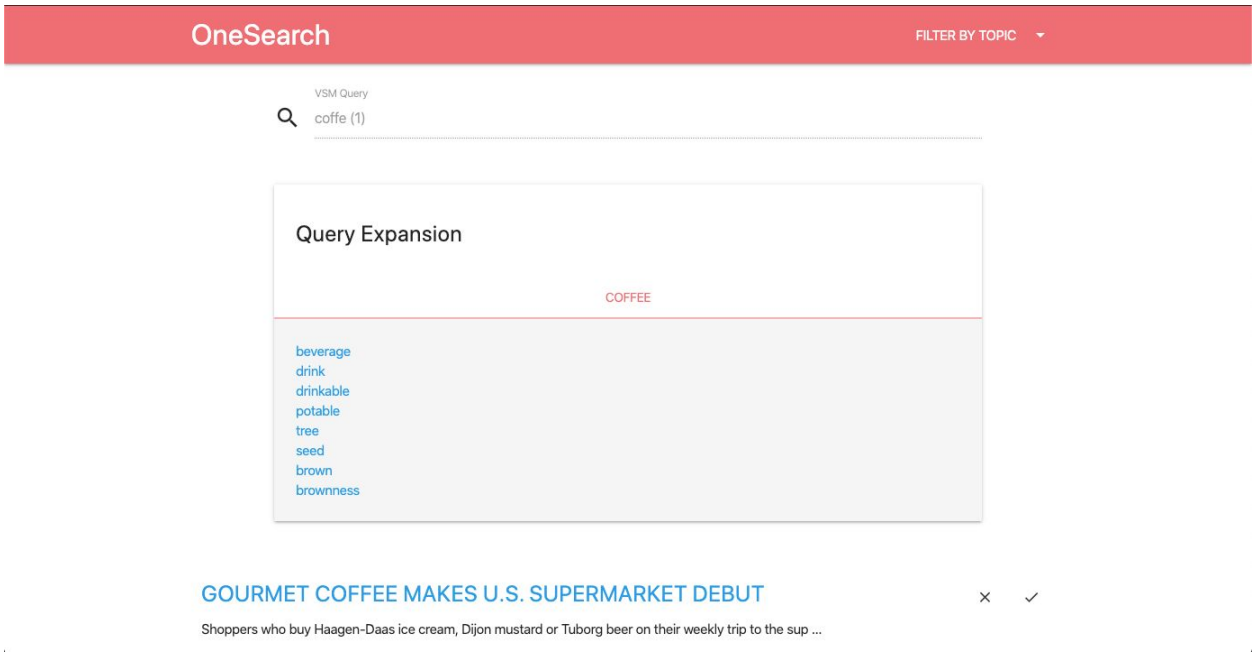
Query Completion Results with *ealier* input for Reuters collection

Query Expansion (Module 3)

The query expansion module with WordNet (located in *engine/views.py* in *search_results* function) offers the user some simple choices to expand their query. The module will show

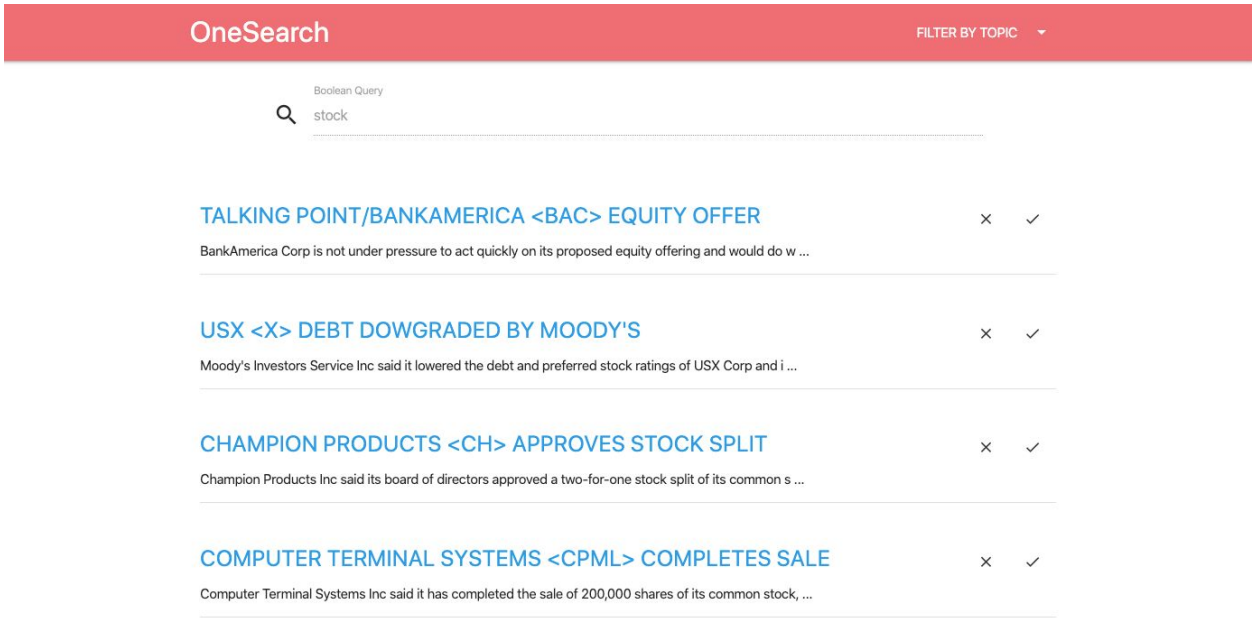
options for each word with less than 10 synsets, and it will only show one word hypernyms of those synsets. For VSM query expansion, it will always just have the word as a score of 1, and for boolean it will add an OR to the word it is expanding.

Query Expansion Options expansion with VSM Query “coffee”



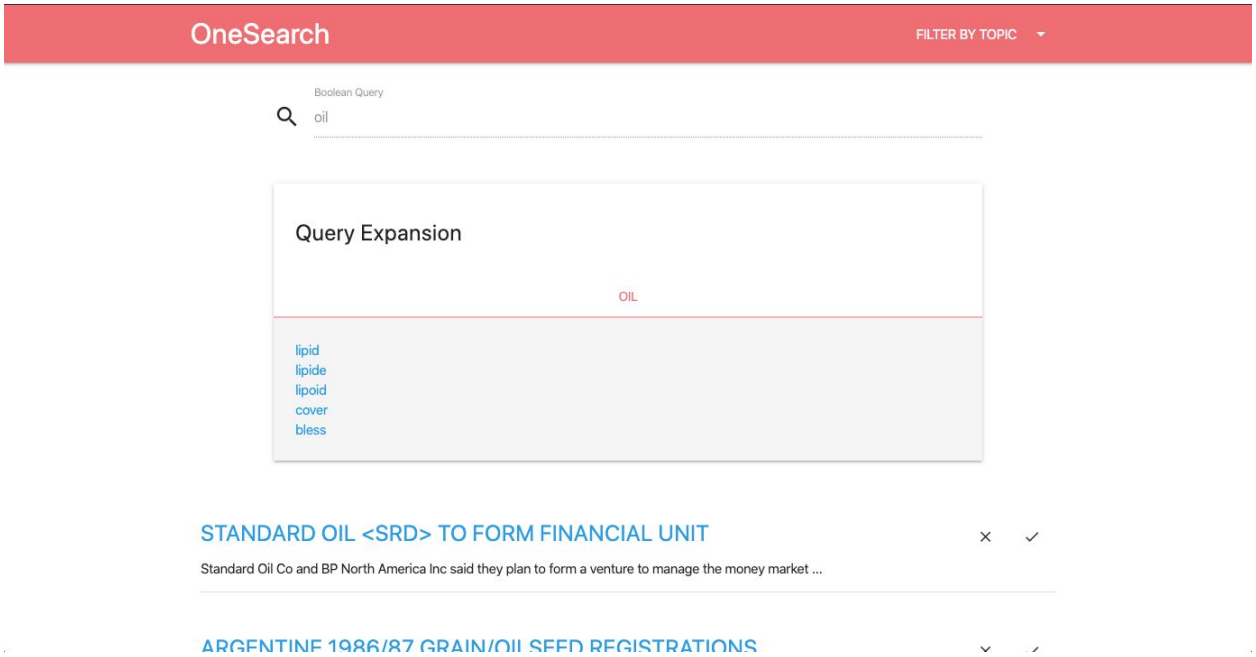
Can potentially add *beverage* and *drink* to the query.

Query Expansion Options expansion with Query “stock”



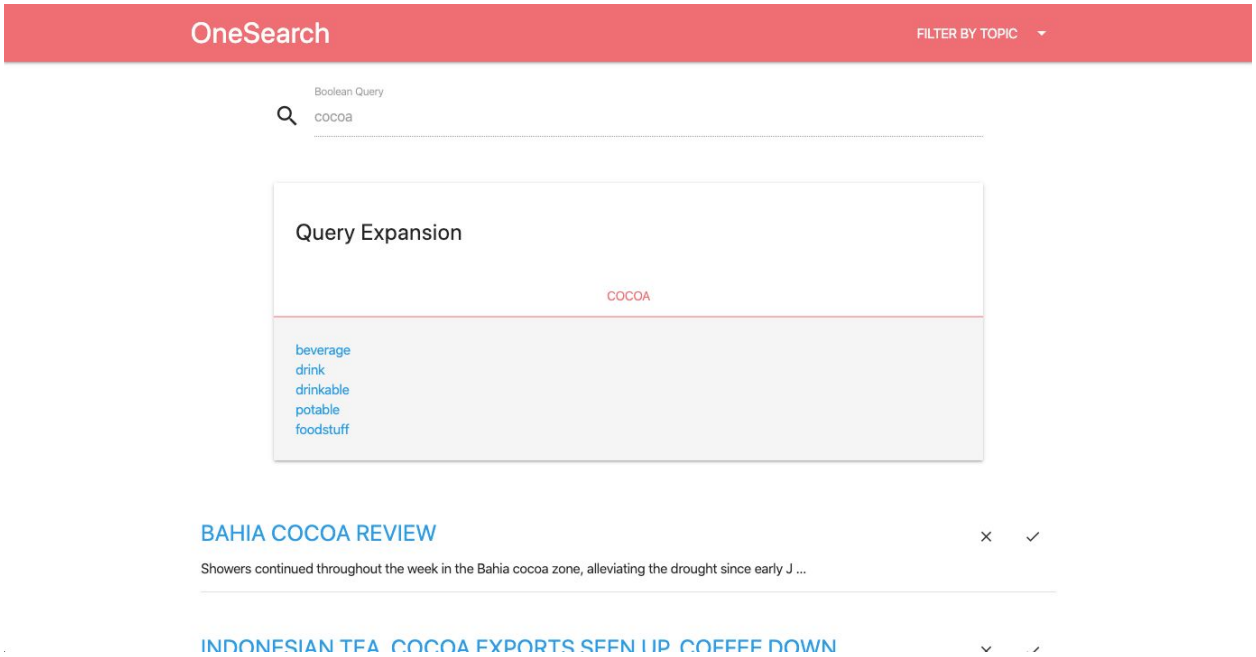
The word stock had too many synsets and therefore did not offer any suggestions.

Query Expansion Options expansion with Query “oil”



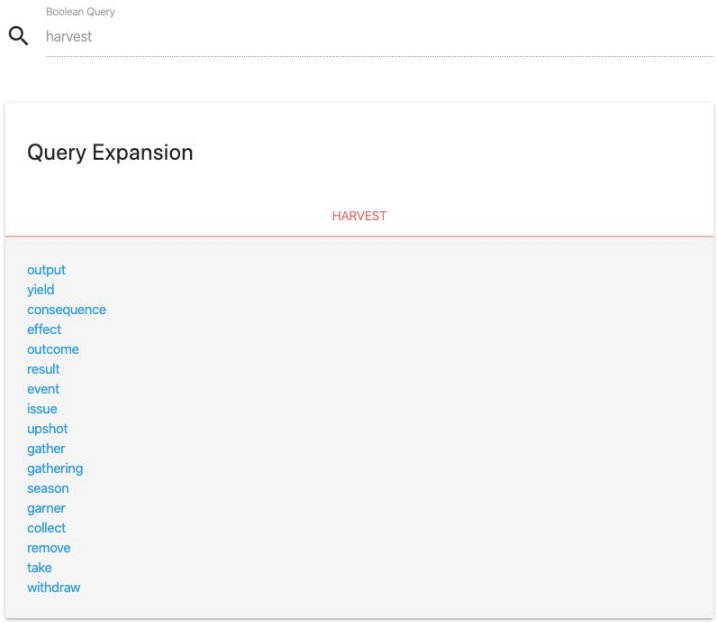
There are no options of value with the word *oil*.

Query Expansion Options expansion with Query “cocoa”



The words *beverage* and *drink* are potentially options to expand *cocoa*

Query Expansion Options expansion with Query “harvest”



Can potentially use *output* or *yield* to expand harvest depending on information need

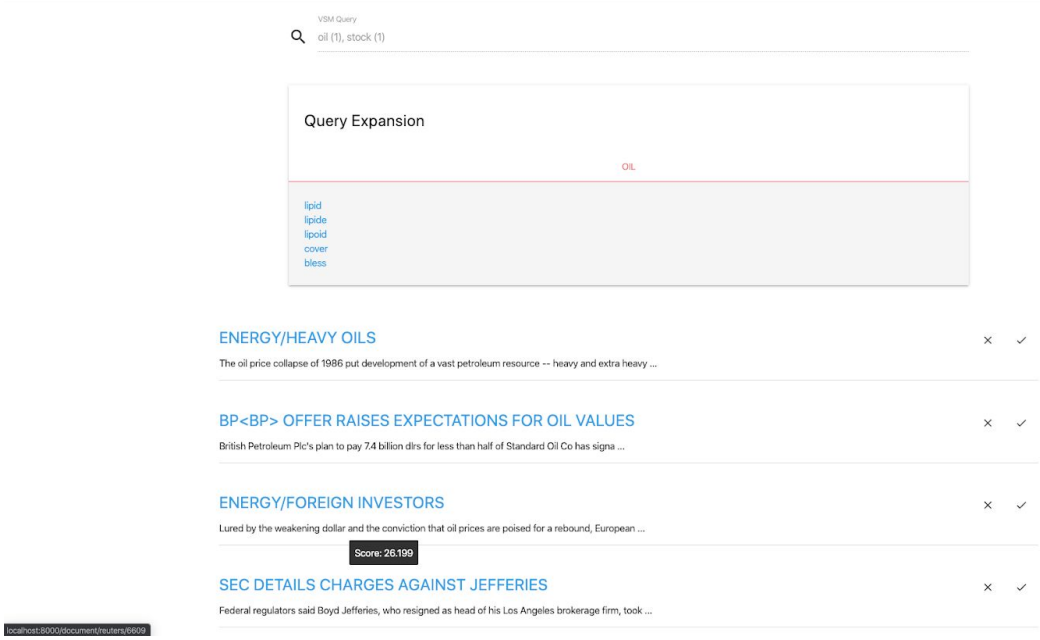
Relevance Feedback + Rocchio (Modules 4 & 5)

The relevance feedback part was designed to be fairly intuitive. Beside each VSM search result, there is a button with a Checkmark and an X to mark a document as relevant or irrelevant for that query. This will then store the documents relevance along with the query and collection in the browsers cookies. The cookies are currently set to the default timeout which means that a session in this case would be when the browser is closed. This is a good definition of a session for relevance feedback because it is likely the user will have a single information need from open to close of a browser thus providing more relevant search results.

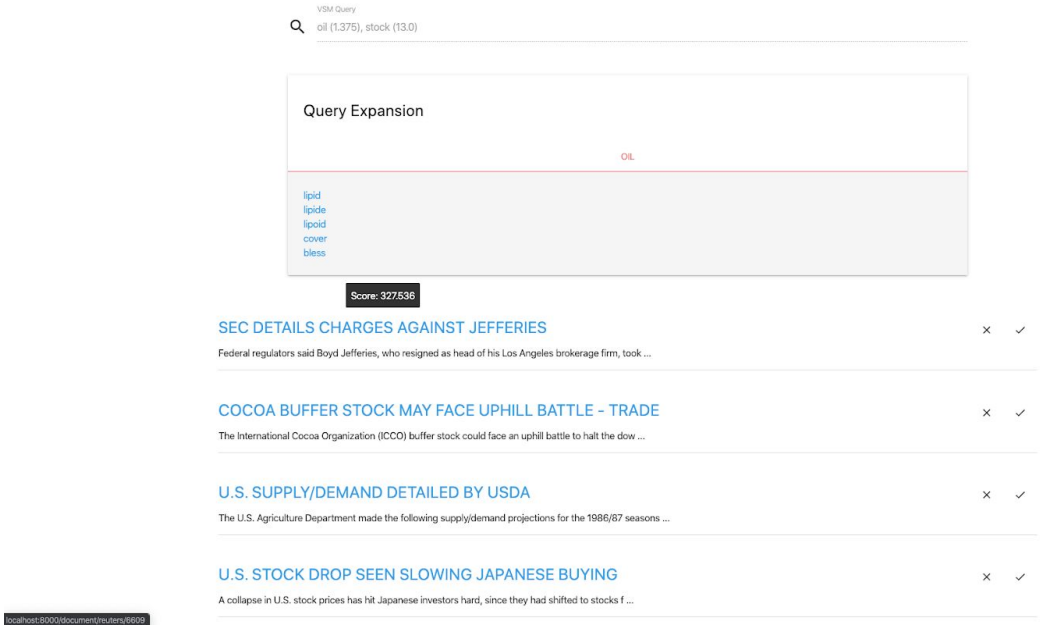
For Rocchio, the formula is used with an alpha of 1, beta of 0.75, and lambda of 0.15. The expansion is also limited to the query terms of the query where the document was found relevant/irrelevant. For example if a document was found relevant with the query “oil”, then only the word oil will be implicitly expanded to in future search queries.

This means that for the search queries “coffee”, “stock”, or “oil” if a document is marked as relevant or irrelevant then only the tf of the search query word in that document is used for expansion.

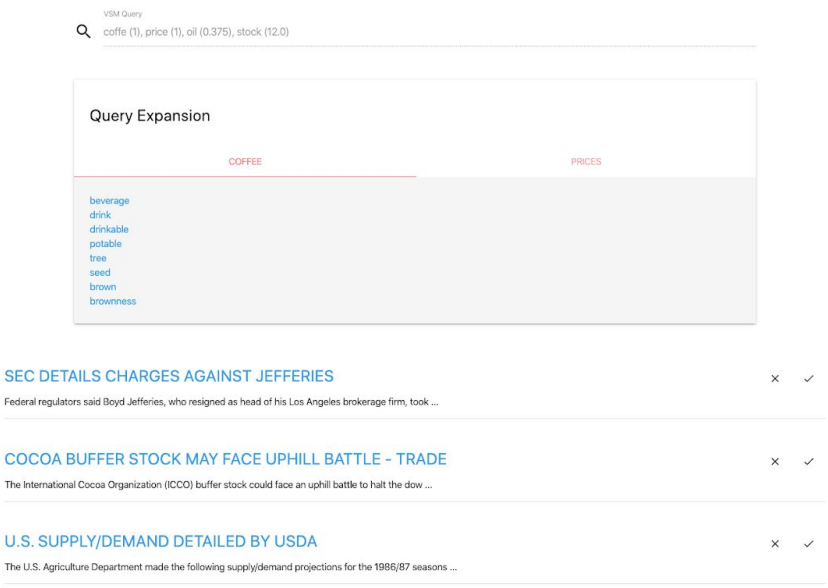
For example for the query “oil stock” shown below, if I mark the first result “SEC DETAILS CHARGES AGAINST JEFFERIES” as relevant, then only those two terms will impact future searches.



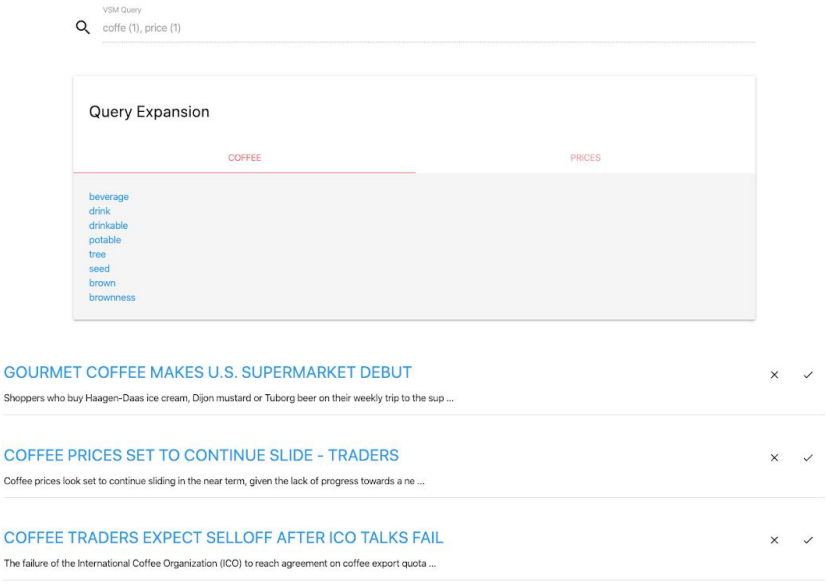
Once it is marked as relevant, you see below that the score of the document changes from 26 to 26 to 327, and the VSM query changes drastically with a large shift towards the keyword stock.



Then if i search something such as “coffee prices”, the keywords oil and stock implicitly expand the search which shifts the search results towards stocks.



Below you see the results without the implicit query expansion and the results are vastly different with no focus on stocks.



Topic Classification (Module 6)

Each of the topics for Reuters are stored in *processed/reuters/topics.json*. The file is initially made in the preprocess_reuters.py script storing the list of topics of each document. If the document did not have topics set then the value “assigned” would be set to false.

```
{
  "id": 0,
  "assigned": true,
  "topics": [
    "cocoa"
  ]
}
```

Within *scripts/generate_topics.py*, topics are assigned to the documents that did not have topics. The script finds the 3 nearest neighbours to the unassigned documents. Since it is possible to have many or no topics, it will then assign the document all topics that occur in at least 2 of those neighbours (if there are no common topics, then the document will have no topics).

Examples of documents that were assigned topic *crude*:
DOC # 1: STANDARD OIL <SRD> TO FORM FINANCIAL UNIT
DOC # 504: POGO <PPP> CONSOLIDATES TWO DIVISIONS
DOC # 557: OPEC PRESIDENT SAYS OIL MARKET BEING MANIPULATED
DOC # 616: BAYER <BAYRY> MAKES U.S. ACQUISITION
DOC # 656: GULF OF MEXICO RIG COUNT FALLS THIS WEEK

This subset of documents that were assigned to the topic crude are very accurate.

Examples of documents that were assigned topic *earn*:
DOC # 2: TEXAS COMMERCE BANCSHARES <TCB> FILES PLAN
DOC # 174: JURY FINDS FOR DOW <DOW> IN BIRTH DEFECT CASE
DOC # 205: TAIWAN OFFSHORE BANKING ASSETS RISE IN JANUARY
DOC # 531: HARNISCHFEGER INDUSTRIES INC <HPH> 1ST QTR NET
DOC # 533: ALBERTSON'S INC <ABS> 4TH QTR JAN 29 NET

This subset of documents that were assigned to the topic earn are mostly accurate. In my opinion only document 174 does not belong there.

Examples of documents that were assigned topic *veg-oil*:
DOC # 3: TALKING POINT/BANKAMERICA <BAC> EQUITY OFFER
DOC # 15: ECONOMIC SPOTLIGHT - BANKAMERICA <BAC>
DOC # 19: CREDIT CARD DISCLOSURE BILLS INTRODUCED
DOC # 27: TOWER REPORT DIMINISHES REAGAN'S HOPES OF REBOUND
DOC # 40: SHULTZ SAYS NO RESIGNATION OVER IRAN REPORT

This subset of documents that were assigned to the topic veg-oil are not at all accurate.

Examples of documents that were assigned topic *interest*:
DOC # 6: RED LION INNS FILES PLANS OFFERING
DOC # 427: PRIME COMPUTER <PRM> UNVEILS PC SOFTWARE
DOC # 435: UNOCAL <UCL> PLANS LUBE CENTERS AT TRUCKSTOPS
DOC # 965: BANGOR HYDRO <BANG> SEEKS RATE CUT
DOC # 1147: AMERICAN NETWORK <ANWI> REDUCES CUSTOMER RATES

This subset of documents that were assigned to the topic interest are somewhat accurate with documents 427 and 435 seeming out of place.