# README

## a. Team Members

1. Aishwarya Sivakumar - as6418
2. Sairam Haribabu - sh4188

## b. List of files

| File name | Usage |
|---|---|
| main.py | Main file which performs google api search, takes relevancy input from user and passes to required methods to get new query. |
| content.py | Constructs content object, calculates bigrams, tf-idf vectors for all search result docs. |
| rocchio.py | Implements updated rocchio's algorithm and query augmentation. |
| transcript.pdf | Test run outputs. |

## c. Dependencies

1. pip3 install --upgrade google-api-python-client
2. pip3 install nltk
3. python3 -m nltk.downloader stopwords
4. python3 -m nltk.downloader corpus
5. python3 -m nltk.downloader punkt
6. python3 -m nltk.downloader wordnet
7. python3 -m nltk.downloader omw-1.4

**Instructions for running the program**

```
python3 main.py <precision> "<query>"
```

## d. Internal Design

Main.py

This file contains the google cloud credentials and takes as input from the user the required precision and search query. It then calls google search api and returns results. If the number of results is less than 10 it terminates. Otherwise, call the process_search method within to scrape the title and snippet and store it in a list of data dictionaries. It also takes relevancy input from the user and returns the number of relevant results. If the number of relevant results is 0 we terminate. Otherwise, calls Content to create a Content object. Rocchio.augment_query is called with the created Content object to obtain the new query. The new query is passed to repeat the search.

Content.py
This contains the Content class definition, constructor and member functions.
The member functions include

1. get_terms - uses nltk libraries to strip, tokenize, make to lowercase, remove stop words and return terms in the provided data item
2. get_bigrams - user nltk libraries to get possible bigrams in the provided data item and update the member variable bigrams.
3. populate_vocab_bigrams - member function to traverse through all data items and populate member variables vocab, bigrams.
4. calculate_tf_idf - member function to find the term frequencies, document frequencies and construct the tf-idf vectors of size vocab for relevant and irrelevant data items.
5. constructor will set the query, number of relevant documents, and processed search results as data. It creates bigrams, vocab, relevant and irrelevant member variables and calls member functions to populate them

Rocchio.py
This contains the augment_query method which is called in main and an implement_rocchio method.

1. implement_rocchio - Defines beta and gamma as 0.8, 0.2. It gets the relevant and irrelevant vectors from passed Content object and implements Rocchio's algorithm. It then traverses the list of bigrams and increases the final weights of certain words which occur in bigrams. This part will be explained more in detail in part E of the document.
2. augment_query - calls implement_rocchio to get final weights of terms. Then it chooses the two highest weight words and orders them with the already existing query words and returns the query. This part will be explained more in detail in part E of the document.

## e. Query Modification

Query modification process used involves finding term frequencies, document frequencies, tf-idf vectors for all data items, rocchio’s algorithm, bigrams enhancement and inplace reordering of the chosen words.

Preprocessing - every data item is tokenized, converted to lowercase and stripped off of stop words.

Bigrams Construction - bigrams are found in every data item using nltk libraries and stored in the Content.bigrams member variable. If the data item was termed relevant by the user, then the found bigrams are added to Content.bigrams. If it was termed irrelevant by the user, then the found bigrams are subtracted from Content.bigrams.
The member variable finally only contains positive bigram values.

TF-IDF - The data items are iterated and term frequencies, document frequencies are calculated. The size of these vectors is the whole vocab for normalization purposes. Tf and idf values are combined to form tf-idf vectors. These vectors are now stored separately for relevant and irrelevant data items.

Rocchio - We use alpha as 1, beta as 0.8 and gamma as 0.2. The relevant and irrelevant vectors constructed in Content are used to implement basic rocchio's algorithm.

Bigram enhancement - We traverse through all the bigrams and do the following

1. for each word in a bigram, increase the weight of the word to the max(2, its bigram count) times.
2. If a part of the bigram was in the query words, then we increase the weight of the other word to 1.5 times.

The final weights are used to choose 2 augment words.

Choosing words and ordering - We choose two of the words with highest weight in the vector returned by rocchio. For ordering, we keep the initial original query words as is and reorder the others in descending order of weight.

**f. Google Cloud Credentials**

| | |
|---|---|
| Google Custom Search Engine JSON API Key | AIzaSyAo07nnsj2_blbE3Yh_N0d1eL8vSth73Pg |
| Engine ID | acc8d36446995d94c |