Domain Specific Information Retrieval System

(news articles retrieval system)

This is a search engine built using tf-idf model that can retrieve relevant news articles from a corpus of 2,00,000 news articles. We've tried to fetch more relevant by incorporating LDA (Latent Dirichlet Allocation) with the tf-idf model.

Architecture

The entire code can be broken down into following sub components:

- Pre-processing
- Index Construction
- LDA
- Data Retrieval

About Corpus

The corpus consists of headline and summary of news articles. We've used both of them but have given a weightage of 2 if a word is in headline compared to weightage of 1 if it is in summary.

Pre-processing

- Removed punctuation marks [!"#$%&'()*+,-./:;<=>?@[\]^_`{|}~]
- Convert everything to lowercase
- Tokenised sentences into words using word_tokenise() of nltk
- Remove stopwords
- Stemming using PorterStemmer

Index Construction

```
"""
example of an entry to master dictionary
master_dict =
{
  'dog':
  {
    doc_number:
    {
      'tf':10,
      'df':15,
      'tf-idf':0.6
    }
  }
}
"""
```

The master_dict hashmap is used to store for each unique word -> the documents in which it is present -> with tf, df and idf.

The matrix used for tf-idf is a very sparse matrix which wastes a lot of memory. The master_dict hashmap in very efficient in terms of memory.

Index construction for the entire master dictionary had the following steps involved

1. For each doc_id in our corpus we took all the pre-processed terms and for each term we updated the dictionary with that term as key if not present, then for that term we updated the doc_id as value and incremented the term frequency(tf) associated with the doc_id.
   We did this separately for headlines to increase the term frequency with a factor associated. Then we did it normally for short description of the news document and incremented tf by 1.
2. We updated the document frequency and tf-idf corresponding to each term -> doc_id in our master dictionary
3. We divided the tf-idf with the normalisation factor calculated using L2 Norm Normalisation

LDA

Rationale

The frequency of some of the proper nouns in articles like Trump, North Korea is quite high. Also, the occurrences of these are spread across a wide range of topics. So, we modelled the articles in corpus into 20 different topics. We've used the LDA scores of query and articles along with tf-idf scores to retrieve articles. We've used genism library for implementing LDA model.

How LDA helps

If we have a query "apple iphone":

Tf-idf model will give results related to apple fruit also

LDA model helps to filter out such results or gives them lower preference

Example of how a topic score would look like:

```
Topic: 18
Words: 0.056*"white" + 0.054*"hous" + 0.048*"call" + 0.036*"trump" + 0.026*"investig" + 0.025*"say" + 0.025*"attack" + 0.024*"cop" + 0.023*"poli
c" + 0.022*"kid"
```

The scores of each topic for a document is stored in the hashmap ‘doc_lda’.

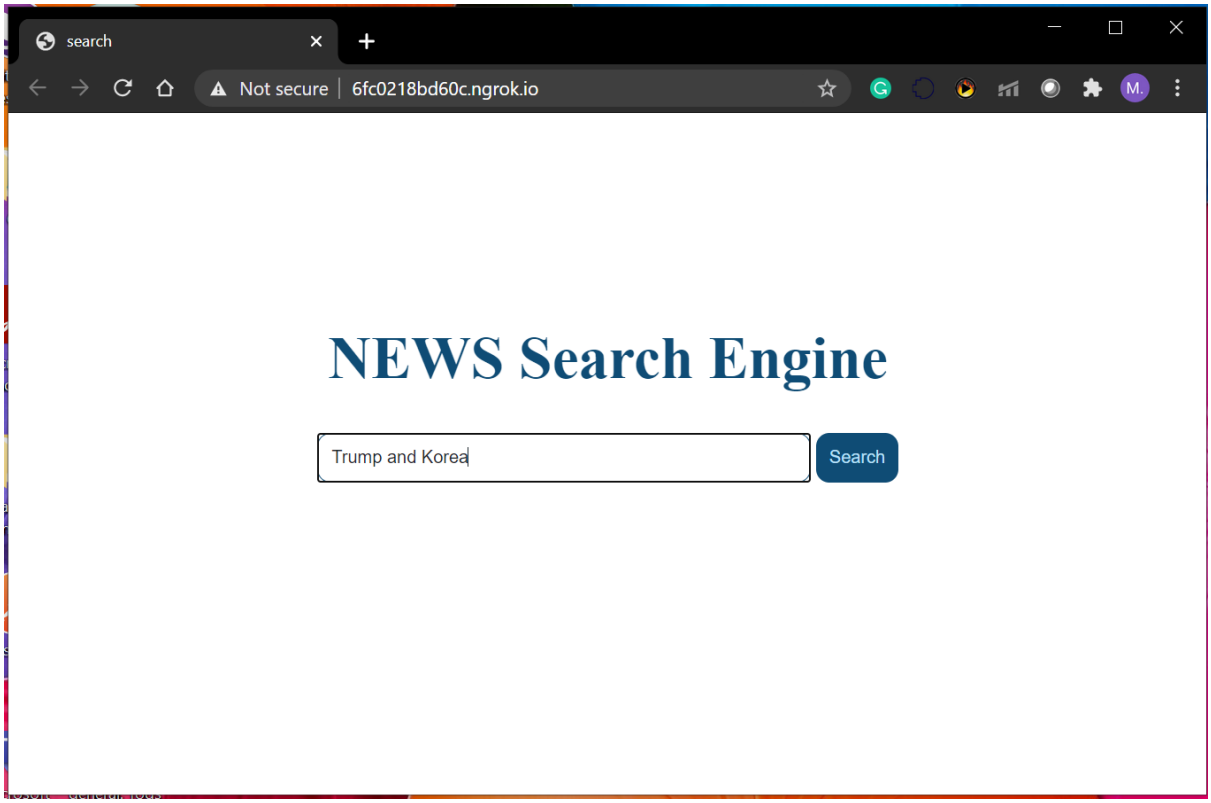
```
"""
example of an entry to doc_lda

doc_lda =
{
  doc_num_0:
  {
    topic_id_0:score,
    topic_id_1:score,
    topic_id_5:score
  }
}
"""
```

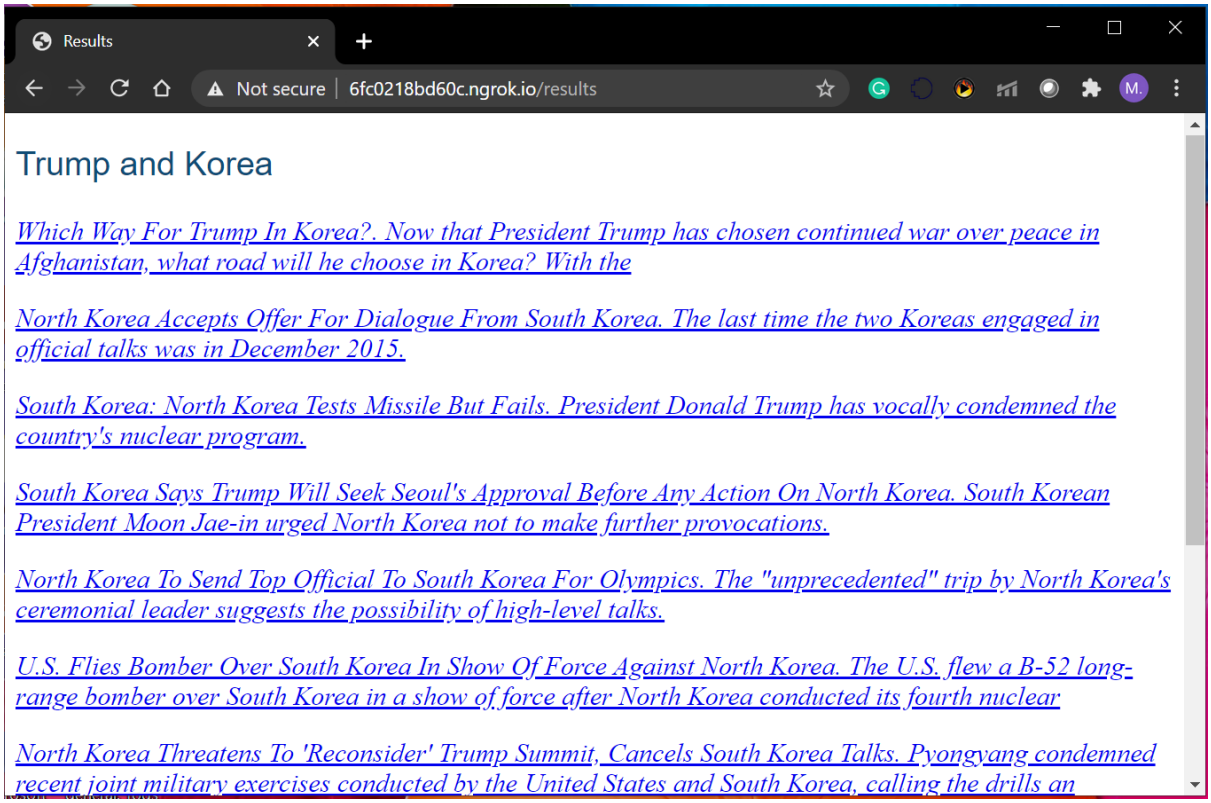
GUI

The GUI is made using html and CSS and flask framework is used to fetch and display results.

Query: Trump and Kim



From the results you can go to the website of news article by clicking on it.



Extracting Results from a Query

The query is pre-processed and query_dict is formed. For the words present in master_dict, tf-idf values are calculated and normalised. Then, LDA topic scores are calculated for query.

Then we find cosine similarity of the topic scores of query with the topic scores of all the relevant documents in our corpus. And normalize this score with a factor. Now we add the cosine similarity scores of tf-idf and topic modelling(LDA) and output the top results.

Results

Takes less than 1 sec to retrieve results

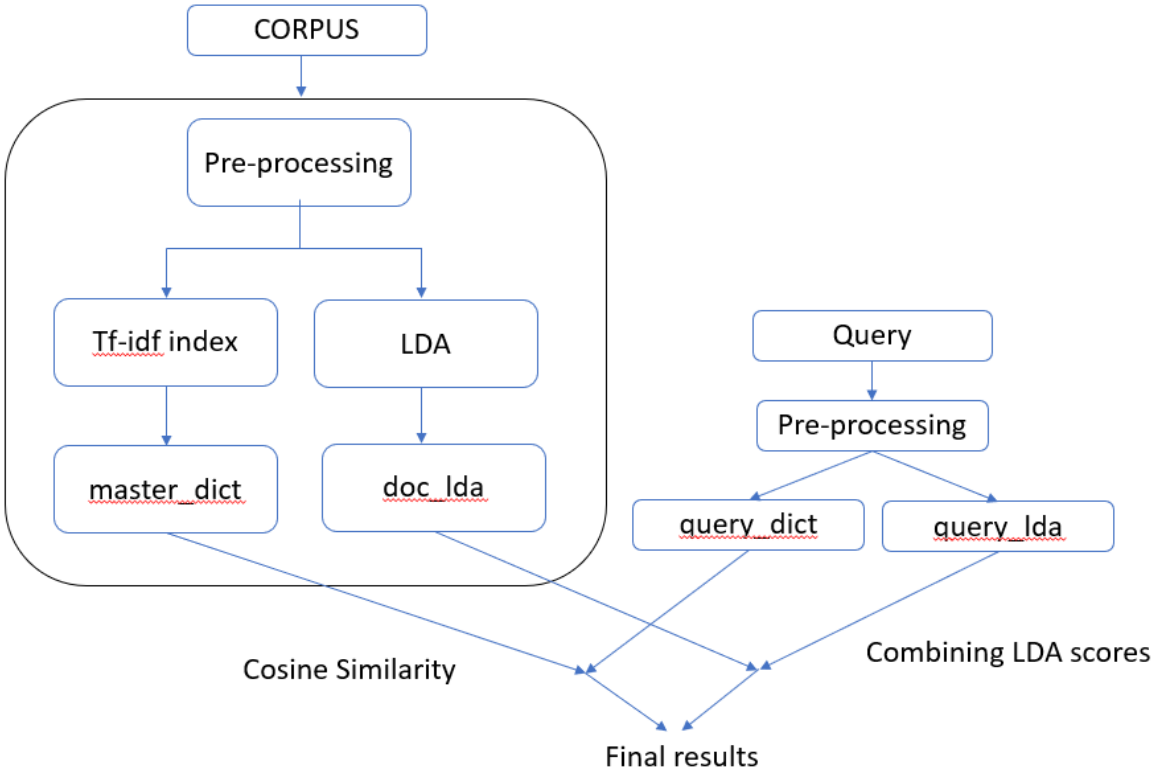
Total news documents = 188042

Total unique words in corpus = 81246

Time taken to create the entire model over Google Colab = 32mins

(Both for index and LDA model construction)

Flow Diagram



Group Members

Adesh Kumar Pradhan – 2017B3A70960H

Pranavi Marripudi – 2018A7PS0507H

Mereddy Aishwwarya Reddi – 2018A7PS0276H