PES UNIVERSITY
(Established under Karnataka Act No. 16 of 2013)
100 Ft. Road, BSK III Stage, Bengaluru – 560 085
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SESSION: AUG-DEC 2020

Course Title: Algorithms for Information Retrieval		
Course code: UE17CS412		
Semester : VII sem	**Section: -**	**Team ID: 23**
SRN: PES1201700986	**Name: Pradyumna Y M**	
SRN: PES1201701604	**Name: Ajeya B S**	
SRN: PES1201701629	**Name: Pruthvish E**	
SRN: PES1201701435	**Name: Harish P B**	

ASSIGNMENT REPORT

1. Problem Statement

The aim of the assignment is to build a search engine on a static corpus of the Environmental News NLP archive , and compare its performance with a standard search engine like Elasticsearch.

2. Description

The search engine is built with python. The following are the files implemented:

- preprocess.py - This file preprocesses the data from the dataset and writes it to a binary that can be read and used by subsequent modules. This module has various options for preprocessing, such as stemming, lemmatization, stop word removal, etc. **Multiprocessing** is being used in order to reduce processing times. Here, the following steps are carried out:
  - The CSV files are read, and each snippet is cleaned using various regular expressions to remove non ASCII characters, special characters and other UTF-8 characters.
  - Case folding is done as a normalization step in order to have a uniform set of words.
  - Based on the config file, other normalization techniques are applied to the raw data.
  - The following mappings are saved to the binary, to help the creation of indices.
    - rowdict - A dictionary mapping a unique row_id to (row_number, document_name, station_name, show_name)
    - rowtems - A dictionary mapping a unique row id to the processed list of tokens to be used by the indices.
    - rowsnip - A dictionary mapping a unique row_id to the snippet contained in the row. Used by the flask app to display results.
    - word_corpus: A set of unique words found in the corpus that will be used later for tolerant retrieval.(spell correction using Levenstein distance.)
- query.py -
  - This file takes in the user query and based on the index mentioned in the config.py file, queries the respective index and returns the results.
  - The snippets, terms and details of each row is retrieved from the pickle file.
  - The user can query within a particular document by enclosing the document name in angular brackets. In such a case, only snippets of that document are filtered before querying.
  - The user can also query for a particular station and show by enclosing them in backticks. In that case, after the query results are returned , they are filtered such that the results are from the particular station/show.
  - After the results are returned from the index, they are formatted such that the format is similar to that of Elasticsearch.

- indexes.py - This file is the heart of the search engine. This file constructs the index for the corpus. Before implementing the index, isolated spelling correction is done.
  3 types of indexes are implemented.
  1. Vector Space model :
     - The snippets returned are the ones which are the most similar to the query. This is done with the help of cosine similarity.
     - A TAAT approach is used. The tf-idf score is calculated for each term in the query for the query and each document, and the dot product is updated.
     - After tf-idf scores for all terms are calculated, it is normalized with the product of the query magnitude and the document magnitude.
     - The resulting number is the cosine similarity and the best results are returned based on the top score.
  2. Boolean Query :
     - A simple boolean query model is implemented, i.e, without nested conditions.
     - Wild card queries are supported for Boolean queries. A BST is used to implement wild card queries. For wild card queries where * has a prefix and suffix, a reverse BST is used with the BST, where the terms fed to the reverse BST are the reversed terms of the corpus. The words which are returned from both BST and revered BST are considered.
     - Whenever OR or NOT is encountered, the query is split and for OR, the union of the 2 results are returned, and for NOT the difference of the 2 results are returned.
  3. Positional Index :
     - Positional index matches the snippets which have the terms in the same order as that of the query. Initially, the positions of the terms of each document are stored in the inverted index.
     - The query terms are checked, and the dictionary of all the terms are retrieved.
     - Next based on the position of the query term, the position number is subtracted from each entry of the dictionary of that term. The advantage of this is that, instead of checking the positions by incrementing the preposition number each time, we can look out for the same number for each term.
     - For each document , the following is done:
       - Check if all terms are present. If not, the document is dropped.
       - If the terms are present, the position number of the terms are checked. If the position numbers are equal for all terms , the document is returned.

- bstree.py - A helper module with a function to create a balanced BST from a list of tokens. This is done using a recursive algorithm. This module also has some functions used to create the Reverse Index, to search for wild card queries. This is done using two functions:
  - Search(root, value) - returns the lower bound of value in the BST rooted at root. This is used to search for the first term for a wild card query.
  - inOrderSuccessor(root, node) - returns an inorder successor(next element) in the BST, that is greater than the current element “node”. This is repeatedly called till the tree is exhausted or the wildcard query no longer matches.
- colorize.py and timer.py - helper module to colorize the outputs and time the execution times of various query and index construction functions.
- config.py - A file that holds all the config details for the 2 search engines. Some of the config parameters are:
  - 'preprocess_type': Type of normalization to be applied. (stemming, lemmatization, etc).
  - "stopword_removal" : A boolean flag to turn stopword removal on or off.
  - "index" : The type of index to be used for the query
    - Vector space model
    - Boolean query
    - Positional index
  - "tf_scheme" : The formula to be used for computing the TF terms.
    - 1+log(tf)
    - log(1+tf)
    - raw tf values
  - "threshold_score" : The minimum score (TF IDF score) that a document should have in order to be considered for results.
  - "spell_check" : A boolean flag to toggle spelling correction for query.

    - "es_index" : Name of the ElasticSearch index to be used.
    - "result_size" : The maximum number of documents to be retrieved by the indexes.
    - "es_host" : Host IP address for the elastic search client.
    - "es_port" : Port for the elasticsearch client.
- metrics.py - A module that is used to calculate the performance metrics for a given query. This is done using a function metrix(query), which returns a confusion matrix for the search engine for the given query.(TP, FP, FN, TN). This module is used to calculate metrics and draw plots.
- flaskapp.py - A module that is used to create a flask server. It serves a static HTML page that can be used to query our search engine. It also has a route, /search, that can be used to query the search engines using HTTP requests.

Elasticsearch setup and indexing

- Elasticsearch/ES.py -
    - This file builds an index in Elasticsearch if it does not exist. the generate_actions() reads the csv files through csv.Dictreader() and yields a single document for each row. This function is passed into the streaming bulk API to insert many documents in sequence
    - It insert unique identifier "id" and document name "document_name" that is used to compare results
    - It uses search_snippet that takes the search_object and searches it in the index mentioned in config.py. It uses elasticsearch.search that returns the matched documents as JSON object
    - It has other helper functions to delete an index, list all the indexes
- Elasticsearch/compare.py -
    - This file generates random queries from the snippets and computes metrics such as F1-score, recall, precision. The scores along with the queries have been dumped into pickle file.
- extract.py -
    - This file reads the queries and scores from the pickle file and is organized into a dataframe that has been downloaded as a csv file.

In summary, we have implemented the following:

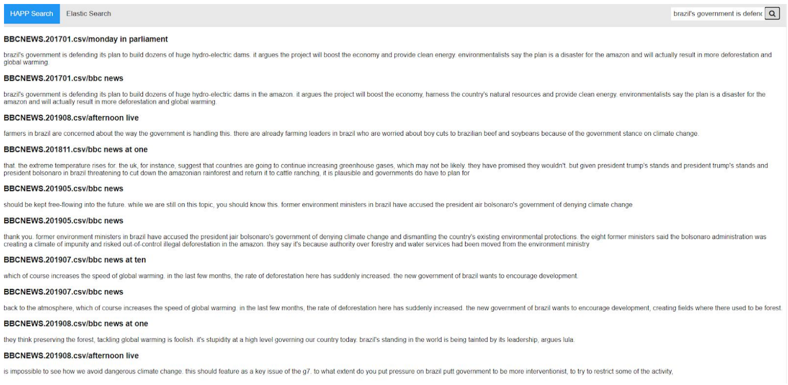
1. Text normalization, cleaning, stemming, lemmatization.
2. Various Indices:
    a. TFIDF index - using TAAT partial scores and normalization.
    b. boolean query index - using BST and Reverse BST. Supports AND,. OR and NOT queries.
    c. positional index - supports phrase queries.
3. Tolerant retrieval:
    a. Wild Card query: using BST and reverse BST with the help of iterators.
    b. Spelling Correction: Using Levenshtein Distance, and a corpus of english words, along with a corpus of words extracted from the given dataset.
4. A query log: that stores the details of the previous queries, the time taken, and other details, which can be used to improve performance on repeated queries, or to view query and result history.
5. Custom filters for searching:
    a. Search By document
    b. Search by show (name of the TV show)
    c. Search By channel.
6. Elasticsearch setup and retrieval
    a. Build one index containing each record as a document
    b. Perform search on different queries to compare the documents returned
    c. Generate random queries to compare the results against our search engine
7. Plots for comparison

8. A flask based REST API to perform queries on both elasticsearch, and our search engine
    This is used to build the backend section of the user interface.

9. A web based tool that can be used to query and index: Please refer to the output screenshots below.
    We have created a static page that queries our search engines using HTTP requests.

3. Output Screenshots

Search Engine Interface



brazil's government is defenc 🔍

BBCNEWS.201701.csv/monday in parliament

brazil's government is defending its plan to build dozens of huge hydro-electric dams. it argues the project will boost the economy and provide clean energy. environmentalists say the plan is a disaster for the amazon and will actually result in more deforestation and global warming.

BBCNEWS.201701.csv/bbc news

brazil's government is defending its plan to build dozens of huge hydro-electric dams in the amazon. it argues the project will boost the economy, harness the country's natural resources and provide clean energy. environmentalists say the plan is a disaster for the amazon and will actually result in more deforestation and global warming.

BBCNEWS.201908.csv/afternoon live

farmers in brazil are concerned about the way the government is handling this. there are already farming leaders in brazil who are worried about boy cuts to brazilian beef and soybeans because of the government stance on climate change.

BBCNEWS.201811.csv/bbc news at one

that. the extreme temperature rises for. the uk, for instance, suggest that countries are going to continue increasing greenhouse gases, which may not be likely. they have promised they wouldn't. but given president trump's stands and president trump's stands and president bolsonaro in brazil threatening to cut down the amazonian rainforest and return it to cattle ranching, it is plausible and governments do have to plan for

BBCNEWS.201905.csv/bbc news

should be kept free-flowing into the future. while we are still on this topic, you should know this. former environment ministers in brazil have accused the president air bolsonaro's government of denying climate change

BBCNEWS.201905.csv/bbc news

thank you. former environment ministers in brazil have accused the president jair bolsonaro's government of denying climate change and dismantling the country's existing environmental protections. the eight former ministers said the bolsonaro administration was creating a climate of impunity and risked out-of-control illegal deforestation in the amazon. they say it's because authority over forestry and water services had been moved from the environment ministry

BBCNEWS.201907.csv/bbc news at ten

which of course increases the speed of global warming. in the last few months, the rate of deforestation here has suddenly increased. the new government of brazil wants to encourage development.

BBCNEWS.201907.csv/bbc news

back to the atmosphere, which of course increases the speed of global warming. in the last few months, the rate of deforestation here has suddenly increased. the new government of brazil wants to encourage development, creating fields where there used to be forest.

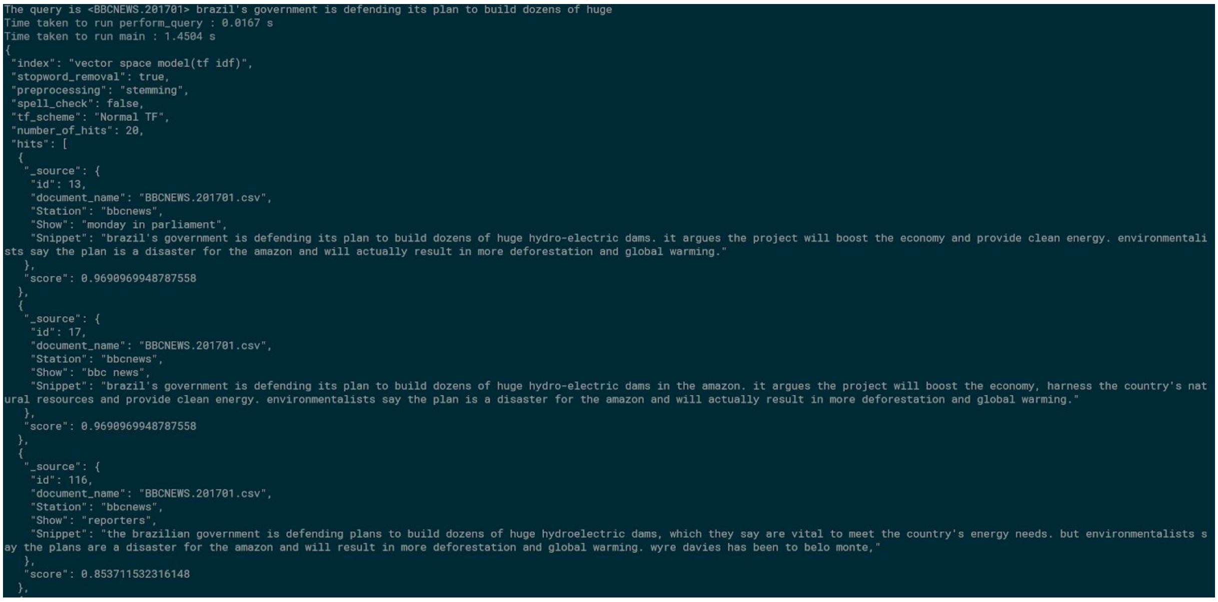
BBCNEWS.201908.csv/bbc news at one

they think preserving the forest, tackling global warming is foolish. it's stupidity at a high level governing our country today. brazil's standing in the world is being tainted by its leadership, argues lula.

BBCNEWS.201908.csv/afternoon live

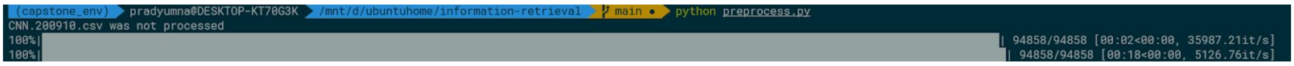
is impossible to see how we avoid dangerous climate change. this should feature as a key issue of the g7. to what extent do you put pressure on brazil putt government to be more interventionist, to try to restrict some of the activity,

Terminal: JSON response for TF-IDF(vector space) model.



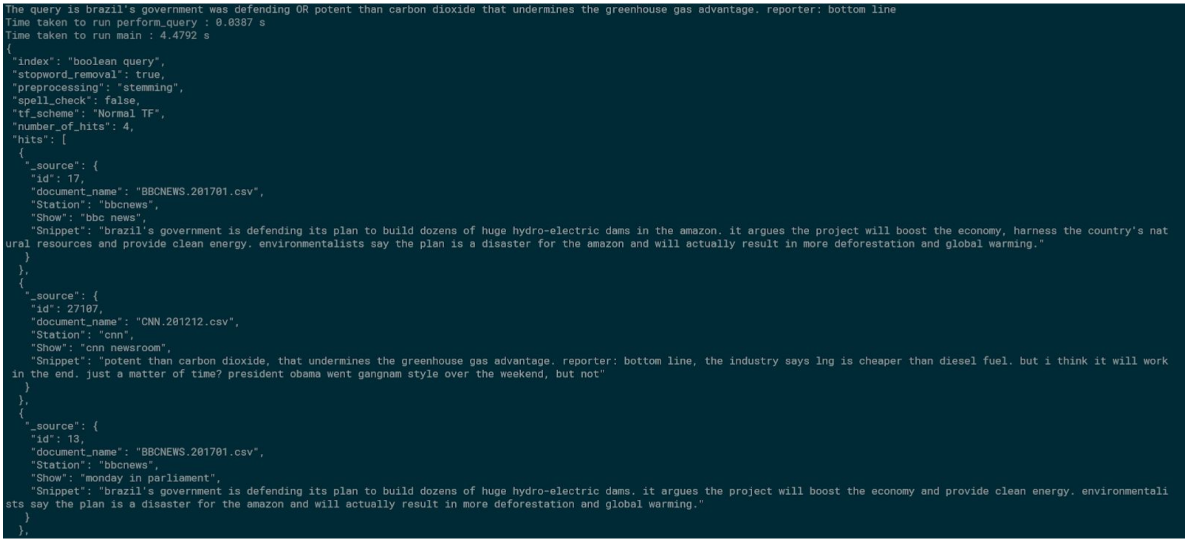
```
The query is <BBCNEWS.201701> brazil's government is defending its plan to build dozens of huge
Time taken to run perform_query : 0.0167 s
Time taken to run main : 1.4504 s
{
 "index": "vector space model(tf idf)",
 "stopword_removal": true,
 "preprocessing": "stemming",
 "spell_check": false,
 "tf_scheme": "Normal TF",
 "number_of_hits": 20,
 "hits": [
  {
   "_source": {
    "id": 13,
    "document_name": "BBCNEWS.201701.csv",
    "Station": "bbcnews",
    "Show": "monday in parliament",
    "Snippet": "brazil's government is defending its plan to build dozens of huge hydro-electric dams. it argues the project will boost the economy and provide clean energy. environmentali
sts say the plan is a disaster for the amazon and will actually result in more deforestation and global warming."
   },
   "score": 0.9690969948787558
  },
  {
   "_source": {
    "id": 17,
    "document_name": "BBCNEWS.201701.csv",
    "Station": "bbcnews",
    "Show": "bbc news",
    "Snippet": "brazil's government is defending its plan to build dozens of huge hydro-electric dams in the amazon. it argues the project will boost the economy, harness the country's nat
ural resources and provide clean energy. environmentalists say the plan is a disaster for the amazon and will actually result in more deforestation and global warming."
   },
   "score": 0.9690969948787558
  },
  {
   "_source": {
    "id": 116,
    "document_name": "BBCNEWS.201701.csv",
    "Station": "bbcnews",
    "Show": "reporters",
    "Snippet": "the brazilian government is defending plans to build dozens of huge hydroelectric dams, which they say are vital to meet the country's energy needs. but environmentalists s
ay the plans are a disaster for the amazon and will result in more deforestation and global warming. wyre davies has been to belo monte,"
   },
   "score": 0.853711532316148
  },
```

Terminal: Preprocessing of data:



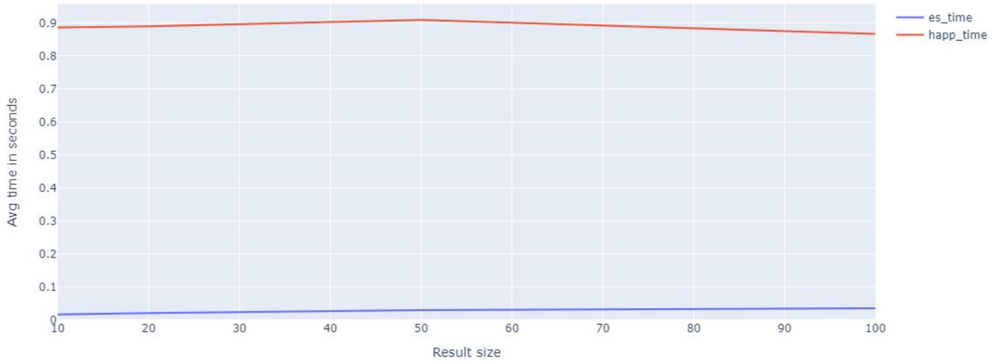
```
 (capstone_env)  pradyumna@DESKTOP-KT70G3K  /mnt/d/ubuntuhome/information-retrieval  main ●  python preprocess.py
CNN.200910.csv was not processed
100%|                                                                                | 94858/94858 [00:02<00:00, 35987.21it/s]
100%|                                                                                 | 94858/94858 [00:18<00:00, 5126.76it/s]
```

Terminal: Boolean query (positional index had a similar result, but the input query is without boolean conditions).

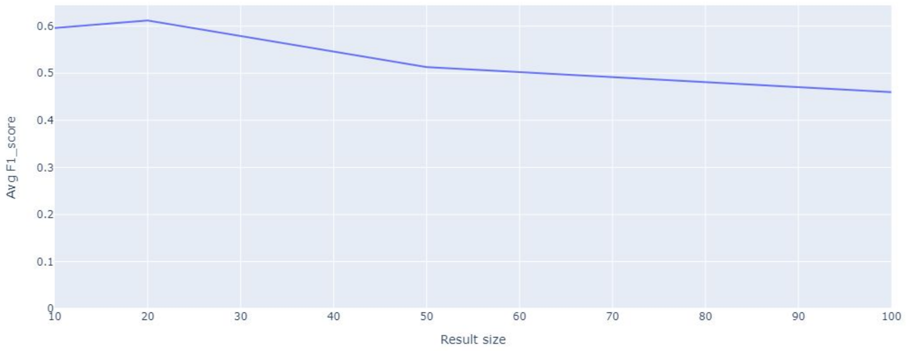


```
The query is brazil's government was defending OR potent than carbon dioxide that undermines the greenhouse gas advantage. reporter: bottom line
Time taken to run perform_query : 0.0387 s
Time taken to run main : 4.4792 s
{
 "index": "boolean query",
 "stopword_removal": true,
 "preprocessing": "stemming",
 "spell_check": false,
 "tf_scheme": "Normal TF",
 "number_of_hits": 4,
 "hits": [
  {
   "_source": {
    "id": 17,
    "document_name": "BBCNEWS.201701.csv",
    "Station": "bbcnews",
    "Show": "bbc news",
    "Snippet": "brazil's government is defending its plan to build dozens of huge hydro-electric dams in the amazon. it argues the project will boost the economy, harness the country's nat
ural resources and provide clean energy. environmentalists say the plan is a disaster for the amazon and will actually result in more deforestation and global warming."
   }
  },
  {
   "_source": {
    "id": 27107,
    "document_name": "CNN.201212.csv",
    "Station": "cnn",
    "Show": "cnn newsroom",
    "Snippet": "potent than carbon dioxide, that undermines the greenhouse gas advantage. reporter: bottom line, the industry says lng is cheaper than diesel fuel. but i think it will work
 in the end. just a matter of time? president obama went gangnam style over the weekend, but not"
   }
  },
  {
   "_source": {
    "id": 13,
    "document_name": "BBCNEWS.201701.csv",
    "Station": "bbcnews",
    "Show": "monday in parliament",
    "Snippet": "brazil's government is defending its plan to build dozens of huge hydro-electric dams. it argues the project will boost the economy and provide clean energy. environmentali
sts say the plan is a disaster for the amazon and will actually result in more deforestation and global warming."
   }
  },
```

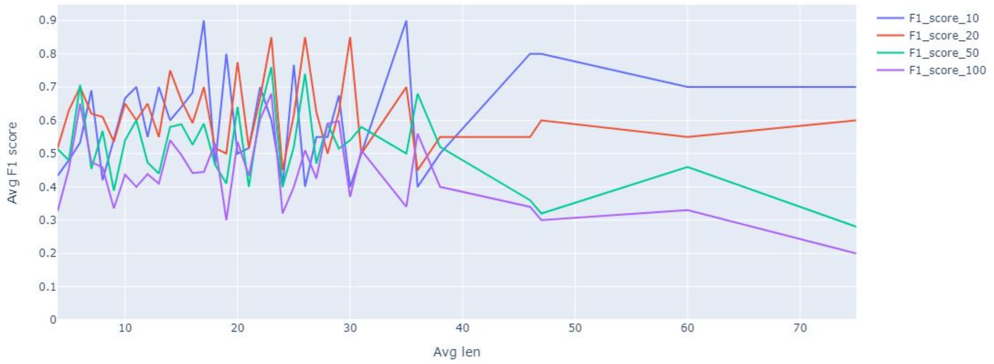
4. Interpretation of efficiency



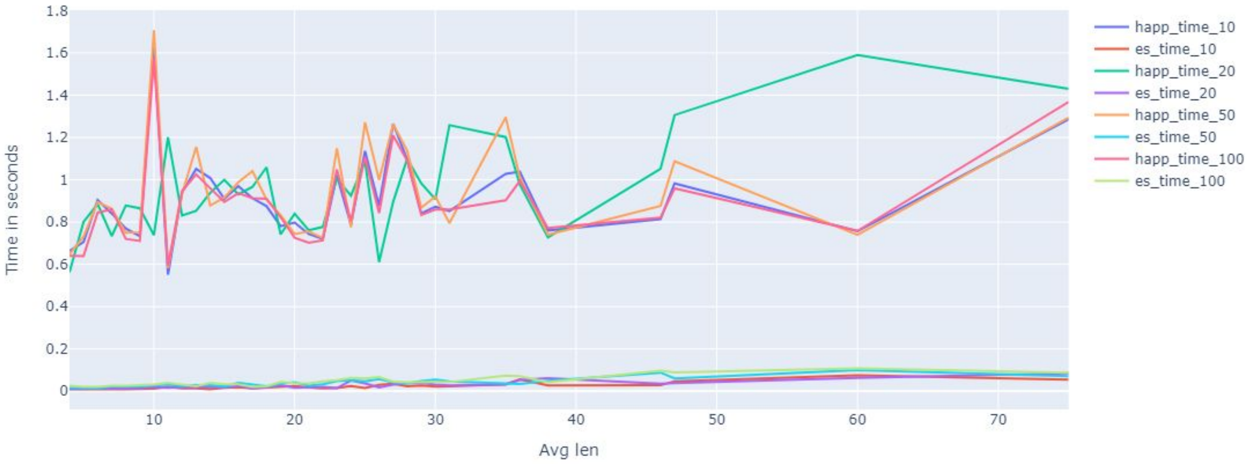
From the above plot, we see that the average time taken for elastic search is better than our search engine for results returned of all sizes, but the difference is only about 1s, which is acceptable for a non-production system,



From the above plot, we see that F1 score starts to reduce after 20 results returned, as the results returned after 20 would not be relevant ones. But still, we notice that the performance doesn't drop by a lot, which indicates that our search engine is in agreement with elasticsearch for a large number of retrieved documents as well.



From the above plot, we can infer that the length of the query does not affect the efficiency of the search engine by a large margin. This tells us that our search engine's results are consistent with elasticsearch even for larger queries.



From the above plot, we find that elasticsearch takes less time than our search engine consistently. But, we see that the performance of our search engine does not degrade with the number of documents to be retrieved.

5. Learning Outcomes

- We understood the various algorithms used in creating and querying indexes and also gained hands-on experience in handling a set of documents efficiently.
- We understood how Elasticsearch works and how a python client can be used to easily insert and manipulate documents in Elasticsearch, and obtain production quality query results and query times.
- We understood how older methods of boolean query and positional indexes return exact results, which has its own set of advantages and disadvantages.
- We understood the effects of various preprocessing and retrieval techniques on the performance and query times, through various experiments.

  We would like to thank each of the faculty members for giving us an opportunity to learn more about search engines and get hands-on experience through this project.

Name and Signature of the Faculty