SEARCH ENGINE GUIDE- Query Processing

NAME: VENKATA SAI VARUN UPPALA

COSC 488

PROJECT: Search Engine (Part 2)

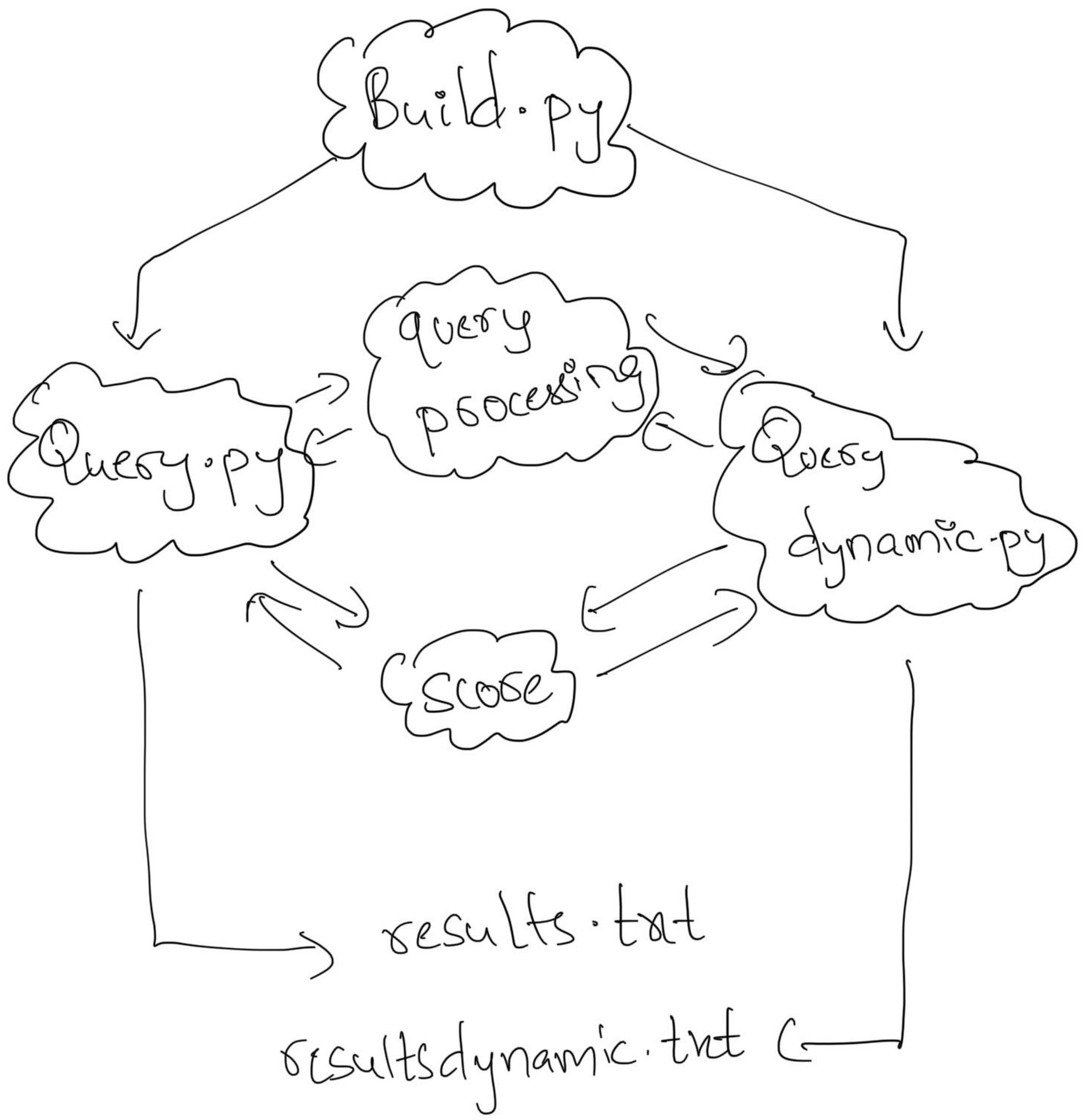
STATUS: Completed

TIME SPENT: 20 - 25 hrs

THINGS I WISH I KNEW BEFOREHAND:
1. I wish I knew that we require document lengths and the original document numbers as I had to include it in the part-1.

2. I wish I knew the proper requirements for each retrieval method,it would have made it easier for calculating the variables.

<u>Engine Design</u>:



- Above is the design plan for the program.

Language: Python
Libraries :

- argparse
- os
- time
- warnings
- pprint
- json
- pandas
- math

Note: Please make sure all the packages are installed before the program is run.

Programs in the directory:

1.build.py

Syntax : python build.py f t o

Description : Perform Indexing

positional arguments:

f	-	Please enter the filename to be indexed
t	-	Please specify the type of index
o	-	Please specify the output directory

2.query.py

Syntax : python query.py i q m t r

Description : Perform Querying

positional arguments:

i	-	index Path
q	-	query Path
m	-	retrieval Mode
t	-	index Type
r	-	results directory

3.querydynamic.py

Syntax : python querydynamic.py i q r

Perform Querying

positional arguments:

i	-	index Path
q	-	query Path
r	-	results directory

Flow for Processing:

- Firstly, the indexes have to be built using the build.py which expects the documents directory as the input and type of index to be built.
- The output of the build.py is a folder output in the same directory as the program is present.
    - Structure : Output
        - IndexName : final.json(inverted index),lexicon.json
        - documentlist.json

  Documentlist.json contains:
  Docid : [actual docid,length of document]
- The above has to be run for each index as required.

Example Command:
python build.py ../PART-1/BigSample single output

- Next, querying can be done as required.
    - Query static - query.py
    - Query dynamic - querydynamic.py
    - Options
        - single
        - stem
        - phrase
        - pos

Query Static

- Query static can be used to evaluate the relevant documents for single and stem indices as per user's method preference.
- Options for indices
    - Single
    - stem
- Options for Retrieval

        - bm25
        - cos
        - lm

Example Command :
python query.py output P2files/queryfile.txt lm stem results

Query Dynamic

    - Query dynamic has a flow to evaluate relevant documents,it uses BM25 retrieval method to find such documents.

Example Command :
python querydynamic.py output P2files/queryfile.txt results

How are queries being handled?

- Firstly, the queries file is run and all the queries are scanned using regex.
- Queries are processed as per the index mentioned as done during build.
- They are later stored into a dictionary.
    - Single: lowercase , stopwords, special tokens,punctuations.
    - Positional:lowercase,punctuations.
    - Stem:lowercase,porter stemmer,punctuations.
    - Phrase:lowercase,punctuations,bigram formation.

What relevance features does the application offer?

- Cosine (without normalising)
- BM25
- Dirichlet smoothing

How is ranking done?

- Scores for all the documents containing the query term are calculated and the top 100 are reported.

The MAP's and Query processing times when evaluated against the trec_eval are reported below as follows.

The MAP's and Query processing times when evaluated using ElasticSearch are reported for the following documents and queries.

REPORT 1 :

SINGLE TERM INDEX:

Retrieval Mode	MAP	MAP - Elasticsearch	Query Processing Time	Query Processing Time- Elasticsearch
Cosine	0.2026	-	1.05s	-
BM25	0.4361	0.3851	1.45s	0.15s
LM - (Dirichlet)	0.1490	0.4628	1.54s	0.15s

STEM TERM INDEX:

Retrieval Mode	MAP	MAP - Elasticsearch	Query Processing Time	Query Processing Time- Elasticsearch
Cosine	0.2077	-	1.07s	-
BM25	0.4414	0.4475	1.55s	0.16s
LM - (Dirichlet)	0.1265	0.4703	1.58s	0.16s

What is the flow of the dynamic retrieval method?

- Phrase index -> Positional index ->Single index -> Stem index.

What is the motive behind using the above flow?

- Firstly, I am going with phrase index so that I can report all the documents which have exactly the same phrase as formed with the query term with the hope that the retrieved documents are relevant and the BM25 score is calculated accordingly.
- Secondly, I am identifying the positions of the query terms,checking whether the document is relevant or not by making sure the query terms are within a window of 4 in the remaining documents and calculating the BM25 by using the single index.
- Thirdly, I am finding the remaining documents score using the single index.
- Finally, I am finding the remaining documents score using the stem index.

- The top 100 documents and their scores are reported.

What relevance features does the application offer?
- BM25 only.

The MAP and Query processing time for the dynamic model is calculated and reported below

REPORT 2 :

Retrieval Mode	MAP	Query Processing Time
BM25	0.4058	19.002s

DRAWBACKS:
- The system cannot normalise the cosine product values.

CONCLUSION:
- The system is efficiently processing queries.Queries are being processed as per the index specified the user in static and in the flow as mentioned in the dynamic method.
- Even though cosine doesn't offer normalisation,it doesn't affect the way the ranking for the documents is being done.
- The phrase selection does not have any threshold as to my evaluation results that if a phrase is present in a document, the following document is relevant.
- The threshold for positions is 4,which as been decided after checking for various sized windows.