Final Project DSAI 201

Name : Alhassan Ali Ahmed ID : **202200681**

➢ Introduction and Description of project:

Building a search engine using an inverted index, where it takes a query and searches for relevant documents after processing the documents and also processing the query.

➢ Data Collection:

I used vaswani dataset from pyterrier as following :

```
vaswani_dataset = pt.datasets.get_dataset("vaswani")
topics = vaswani_dataset.get_topics()
data = topics
qrels = vaswani_dataset.get_qrels()
```

```
Downloading vaswani topics to /root/.pyterrier/corpora/vaswani/query-text.trec
query-text.trec: 10.7k/? [00:00<00:00, 516kiB/s]
Downloading vaswani qrels to /root/.pyterrier/corpora/vaswani/qrels
qrels: 24.3k/? [00:00<00:00, 1.13MiB/s]
```

```
[5] data
```

	qid	query
0	1	measurement of dielectric constant of liquids ...
1	2	mathematical analysis and design details of wa...
2	3	use of digital computers in the design of band...
3	4	systems of data coding for information transfer
4	5	use of programs in engineering testing of comp...
...	...	...
88	89	tunnel diode construction and its electrical c...

The code :

```
vaswani_dataset = pt.datasets.get_dataset("vaswani")
topics = vaswani_dataset.get_topics()
data = topics
qrels = vaswani_dataset.get_qrels()
```

- **Indexing:**
  - Build an inverted index
    - Create a data structure that maps each unique word (or term) to the
  - documents that contain that word.
    - For each term, maintain a list of document IDs where the term appears along
  - with the frequency of occurrence.

```
[17] indexer = pt.DFIndexer("./DatasetIndex", overwrite=True)
     index_ref = indexer.index(topics["query"], topics["docno"])
     print(index_ref.toString())
     index_ref.toString()
```

```
./DatasetIndex/data.properties
'./DatasetIndex/data.properties'
```

```
index = pt.IndexFactory.of(index_ref)
```

Code :

```
indexer = pt.DFIndexer("./DatasetIndex", overwrite=True)
index_ref = indexer.index(topics["query"], topics["docno"])
print(index_ref.toString())
index_ref.toString()
index = pt.IndexFactory.of(index_ref)
```

- **Preprocessing:**

```
[8]  stemmer = PorterStemmer()
     def Steem_text(text):
         tokens = word_tokenize(text)
         stemmed_tokens = [stemmer.stem(word) for word in tokens]
         # print (tokens)
         return ' '.join(stemmed_tokens)
```

```
[9]  def remove_stopwords(text):
         tokens = word_tokenize(text)
         filtered_tokens = [word.lower() for word in tokens if word.lower() not in stop_words]
         print('Tokens are:',tokens,'\n')
         return ' '.join(filtered_tokens)
```

```
def clean(text):
    text = re.sub(r"http\S+", " ", text) # remove urls
    text = re.sub(r"RT ", " ", text) # remove rt
    text = re.sub(r"@[\w]*", " ", text) # remove handles
    text = re.sub(r"[\.\,\#_\|\:\?\?\/\=]", " ", text) # remove special characters
    text = re.sub(r'\t', ' ', text) # remove tabs
    text = re.sub(r'\n', ' ', text) # remove line jump
    text = re.sub(r"\s+", " ", text) # remove extra white space
    text = text.strip()
    return text
```

- **Query Processing:**

```
[23] def preprocess(sentence):
       sentence = clean(sentence)
       sentence = remove_stopwords(sentence)
       sentence = Steem_text(sentence)
       return sentence
```

Code :

```
def preprocess(sentence):
  sentence = clean(sentence)
  sentence = remove_stopwords(sentence)
  sentence = Steem_text(sentence)
  return sentence
```

- **Query Processing:**

```
query = input("Please Enter you query: ")
query = preprocess(query)
results_tfidf = tfidf_retr.search(query)
results_tfidf
```

```
Please Enter you query: digital computers
```

```
query = input("Please Enter you query: ")
query = preprocess(query)
results_tfidf = tfidf_retr.search(query)
results_tfidf
```

```
Please Enter you query: digital computers
Tokens are: ['digital', 'computers']
```

	qid	docid	docno	rank	score	query
0	1	43	44	0	4.819617	digit comput
1	1	42	43	1	4.342149	digit comput
2	1	58	59	2	4.022643	digit comput
3	1	2	3	3	3.623218	digit comput
4	1	45	46	4	3.451844	digit comput
5	1	4	5	5	2.318627	digit comput
6	1	40	41	6	2.318627	digit comput
7	1	41	42	7	2.318627	digit comput
8	1	35	36	8	2.165504	digit comput

```
bm25 = pt.BatchRetrieve(index, wmodel="BM25",num_results=10)

results_bm25 = bm25.search(query)
results_bm25
```

	qid	docid	docno	rank	score	query
0	1	43	44	0	8.117148	digit comput
1	1	42	43	1	7.288686	digit comput
2	1	58	59	2	6.774892	digit comput
3	1	2	3	3	6.102185	digit comput
4	1	45	46	4	5.813559	digit comput
5	1	4	5	5	3.826587	digit comput
6	1	40	41	6	3.826587	digit comput
7	1	41	42	7	3.826587	digit comput
8	1	35	36	8	3.573876	digit comput

- **Query expansion:**

```
[28] if not pt.started():
        pt.init(boot_packages=["com.github.terrierteam:terrier-prf:-SNAPSHOT"])
     rm3_expander = pt.BatchRetrieve(index, fb_terms=100, fb_docs=1000)

     #output of the BM25 will be fed into the RM3 expander for query expansion.
     rm3_qe = bm25 >> rm3_expander
     expanded_query = rm3_qe.search(query).iloc[0]["query"]

     expanded_query
```

```
BR(DPH): 100% 1/1 [00:00<00:00, 24.37q/s]
'digit comput'
```

```
for s in expanded_query.split()[1:]:
  print(s)

print("\n" + query)
```

```
comput

digit comput
```

```
expanded_query_formatted = ' '.join(expanded_query.split()[1:])

results_wqe = bm25.search(expanded_query_formatted)

print("   Before Expansion    After Expansion")
print(pd.concat([results[['docid','score']][0:5].add_suffix('_1'),
            results_wqe[['docid','score']][0:5].add_suffix('_2')], axis=1).fillna(''))
```

```
   Before Expansion    After Expansion
   docid_1   score_1  docid_2   score_2
0        3  2.500542       42  3.873215
1       83  2.500542        4  3.826587
2       81  2.345635       40  3.826587
3       21  2.208801       41  3.826587
4       57  2.208801       35  3.573876
```

+ Code + Text

```
[31] retrieved_Doc = topics[['processed_text']][topics['qid'].isin(results_wqe['qid'].loc[0:10]
     retrieved_Doc
```

	processed_text
0	measur dielectr constant liquid use microwav t...

```
result_merged = results_tfidf.merge(topics, on="qid")[["score", "processed_text"]]
result_merged = result_merged.sort_values(by="score", ascending=False)
result_merged
```

	score	processed_text
0	4.819617	measur dielectr constant liquid use microwav t...
1	4.342149	measur dielectr constant liquid use microwav t...
2	4.022643	measur dielectr constant liquid use microwav t...
3	3.623218	measur dielectr constant liquid use microwav t...
4	3.451844	measur dielectr constant liquid use microwav t...
5	2.318627	measur dielectr constant liquid use microwav t...
6	2.318627	measur dielectr constant liquid use microwav t...

- **Bert :**

```
from transformers import AutoTokenizer, AutoModel
from xpmir.models import AutoModel
model = AutoModel.load_from_hf_hub("xpmir/monot5", as_instance=True)
```

```
/usr/local/lib/python3.10/dist-packages/huggingface_hub/utils/_token.py:88: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (https://huggingface.co/settings/tokens), set it as secret in your
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public models or datasets.
  warnings.warn(
config.json: 100% 2.00/2.00 [00:00<00:00, 138B/s]
definition.json: 100% 18.1k/18.1k [00:00<00:00, 1.36MB/s]
path: 100% 990M/990M [00:07<00:00, 94.6MB/s]
config.json: 100% 1.21k/1.21k [00:00<00:00, 58.9kB/s]
spiece.model: 100% 792k/792k [00:00<00:00, 4.34MB/s]
```

```
[38] output = model.rsv("analysis and design", result_merged["processed_text"].values)

     data =[(list(obj.document.items.values())[0].text,obj.score) for obj in output]
     reviews_result_v2 = pd.DataFrame(data, columns=['document', "score"]).sort_values(by="score", ascending=False)
```

```
reviews_result_v2.sort_values(by="score", ascending=False)
```

	document	score
3	measur dielectr constant liquid use microwav t...	-20.131430
6	measur dielectr constant liquid use microwav t...	-23.040516
5	measur dielectr constant liquid use microwav t...	-23.622366
1	measur dielectr constant liquid use microwav t...	-24.971731
4	measur dielectr constant liquid use microwav t...	-25.007284

- **ELMo:**

```
[35] import tensorflow as tf
     import tensorflow_hub as hub
     import numpy as np
     #load the ELMo model
     elmo = hub.load("https://tfhub.dev/google/elmo/3")
```

```
[36] embeddings = elmo.signatures["default"](tf.constant(result_merged["processed_text"]))["elmo"]

     embedding_D0 = embeddings.numpy()[0]
     embedding_D1 = embeddings.numpy()[1]
     embedding_D2 = embeddings.numpy()[2]
     embedding_D3 = embeddings.numpy()[3]

     #print the embeddings
     print("Embedding vector for D0:", embedding_D0,"\n",20* "------")
     print("Embedding vector for D1:", embedding_D1,"\n",20* "------")
     print("Embedding vector for D2:", embedding_D2,"\n",20* "------")
     print("Embedding vector for D3:", embedding_D3,"\n",20* "------")
```

```
Embedding vector for D0: [[ 0.03040026  0.41542646  0.47917175 ...  0.08776546  0.09890337
  -0.18801557]
 [-0.31924325  0.20698862  0.3663548  ...  0.22917204  0.3439814
  -0.1792855 ]
```

```
# get Size of (vectors) sentences S1 & S2
Size_V_of_D0 = len(embedding_D0)
Size_V_of_D1 = len(embedding_D1)
Size_V_of_D2 = len(embedding_D2)
Size_V_of_D3 = len(embedding_D3)

print(np.vstack(embedding_D0).sum(axis=0),"\n",10* "------")
print(np.vstack(embedding_D1).sum(axis=0),"\n",10* "------")
print(np.vstack(embedding_D2).sum(axis=0),"\n",10* "------")
print(np.vstack(embedding_D3).sum(axis=0),"\n",10* "------")
```

```
[1.4954203  2.373102   1.0622026  ... 0.23686907 0.10839848 1.1826057 ]
 ------------------------------------------------------------
[1.4954203  2.373102   1.0622026  ... 0.23686907 0.10839848 1.1826057 ]
 ------------------------------------------------------------
[1.4954203  2.373102   1.0622026  ... 0.23686907 0.10839848 1.1826057 ]
 ------------------------------------------------------------
[1.4954203  2.373102   1.0622026  ... 0.23686907 0.10839848 1.1826057 ]
 ------------------------------------------------------------
```

```
Sum_Ve_D0 = np.vstack(embedding_D0).sum(axis=0)
Sum_Ve_D1 = np.vstack(embedding_D1).sum(axis=0)
Sum_Ve_D2 = np.vstack(embedding_D2).sum(axis=0)
Sum_Ve_D3 = np.vstack(embedding_D3).sum(axis=0)

Centorid_D0 = Sum_Ve_D0/Size_V_of_D0
Centorid_D1 = Sum_Ve_D1/Size_V_of_D1
Centorid_D2 = Sum_Ve_D2/Size_V_of_D2
Centorid_D3 = Sum_Ve_D3/Size_V_of_D3

print(Centorid_D0,"\n", 10* "------")
print(Centorid_D1,"\n", 10* "------")
print(Centorid_D2,"\n", 10* "------")
print(Centorid_D3,"\n", 10* "------")
```

```
[0.21363148 0.33901456 0.15174322 ... 0.03383844 0.0154855  0.16894367]
 ------------------------------------------------------------
[0.21363148 0.33901456 0.15174322 ... 0.03383844 0.0154855  0.16894367]
 ------------------------------------------------------------
[0.21363148 0.33901456 0.15174322 ... 0.03383844 0.0154855  0.16894367]
 ------------------------------------------------------------
[0.21363148 0.33901456 0.15174322 ... 0.03383844 0.0154855  0.16894367]
```

calculate cosine similarity between the embeddings

+ Code + Text

```
#calculate cosine similarity between the embeddings
def cosine_similarity(v1, v2):
    dot_product = np.dot(v1, v2)
    norm_v1 = np.linalg.norm(v1)
    norm_v2 = np.linalg.norm(v2)
    return dot_product / (norm_v1 * norm_v2)
similarity_score = cosine_similarity(Centorid_D0, Centorid_D1)
print("Cosine similarity between D0 & D1:", similarity_score)
```

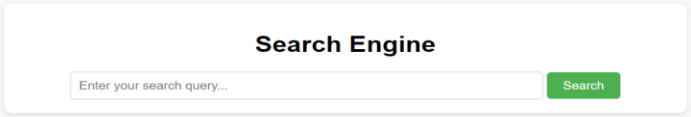
```
Cosine similarity between D0 & D1: 0.99999994
```

Code :

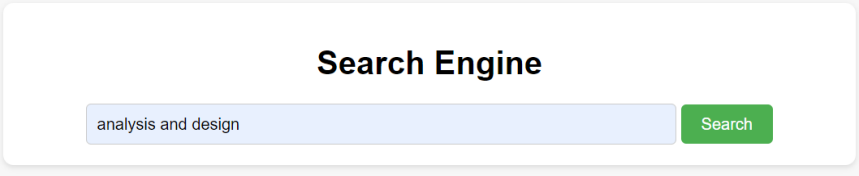
```
def cosine_similarity(v1, v2):
    dot_product = np.dot(v1, v2)
```

```
    norm_v1 = np.linalg.norm(v1)
    norm_v2 = np.linalg.norm(v2)
    return dot_product / (norm_v1 * norm_v2)
similarity_score = cosine_similarity(Centorid_D0, Centorid_D1)
print("Cosine similarity between D0 & D1:", similarity_score)
```

UI :



The result of Search:



Search Results for "analysi design"

qid	docid	docno	rank	score	query
1	1	2	0	4.261485822207533	analysi design
1	18	19	1	2.667820074855919	analysi design
1	81	82	2	2.667820074855919	analysi design
1	58	59	3	2.3737186059518156	analysi design
1	46	47	4	2.1216597719519696	analysi design
1	48	49	5	2.1216597719519696	analysi design
1	59	60	6	1.8877672162557169	analysi design
1	12	13	7	1.789148993811201	analysi design
1	77	78	8	1.789148993811201	analysi design
1	2	3	9	1.700322983500982	analysi design
1	50	51	10	1.6198997049527046	analysi design

Search Engine

digital computers

Search

Search Results for "digit comput"

qid	docid	docno	rank	score	query
1	43	44	0	4.81961736273758	digit comput
1	42	43	1	4.342149250033817	digit comput
1	58	59	2	4.022642769531516	digit comput
1	2	3	3	3.6232178928368275	digit comput
1	45	46	4	3.4518439452608547	digit comput
1	4	5	5	2.3186274113699987	digit comput
1	40	41	6	2.3186274113699987	digit comput
1	41	42	7	2.3186274113699987	digit comput
1	35	36	8	2.1655036033128194	digit comput