Group 39

IR Assignment 1

Krishnasai Addala,

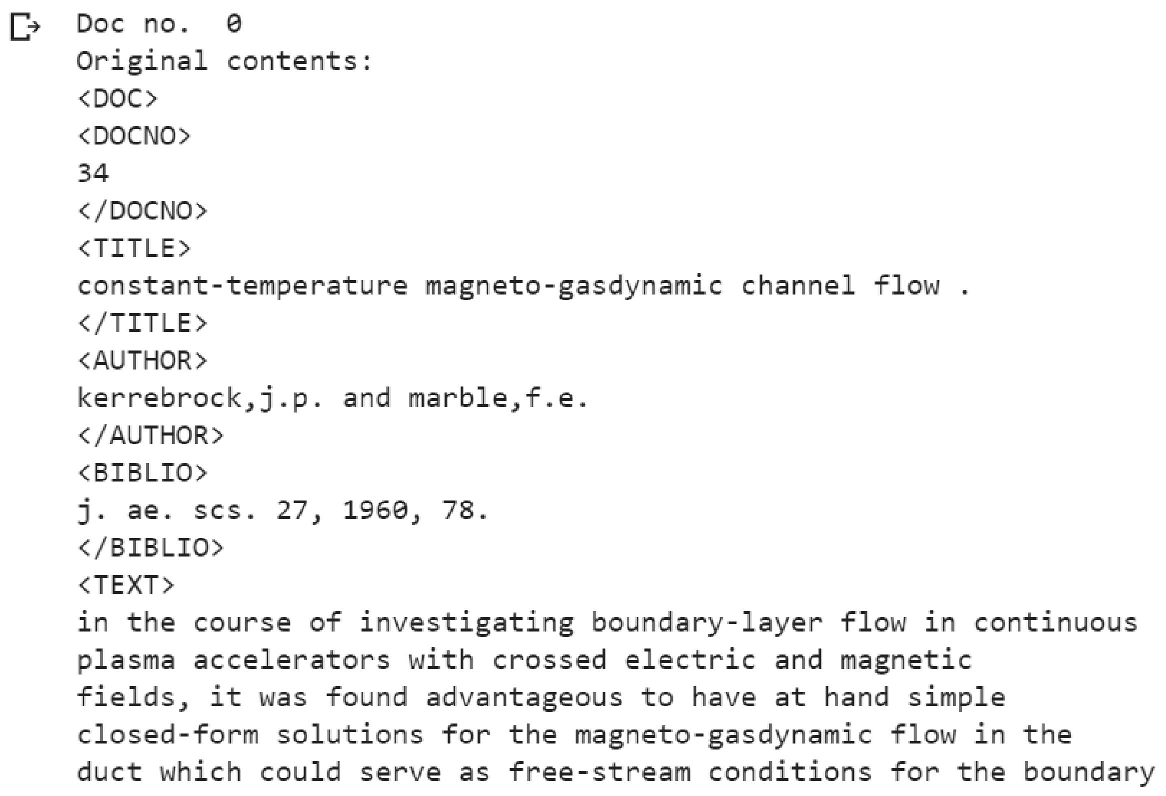
Sahil Deshpande,

Kabir Dev Paul Baghel

The assignment was done using Google Colab and Google Drive. The dataset was hosted on Google Drive, and the code was hosted on Google Colab. The pickle module was used to save and load the indices created.

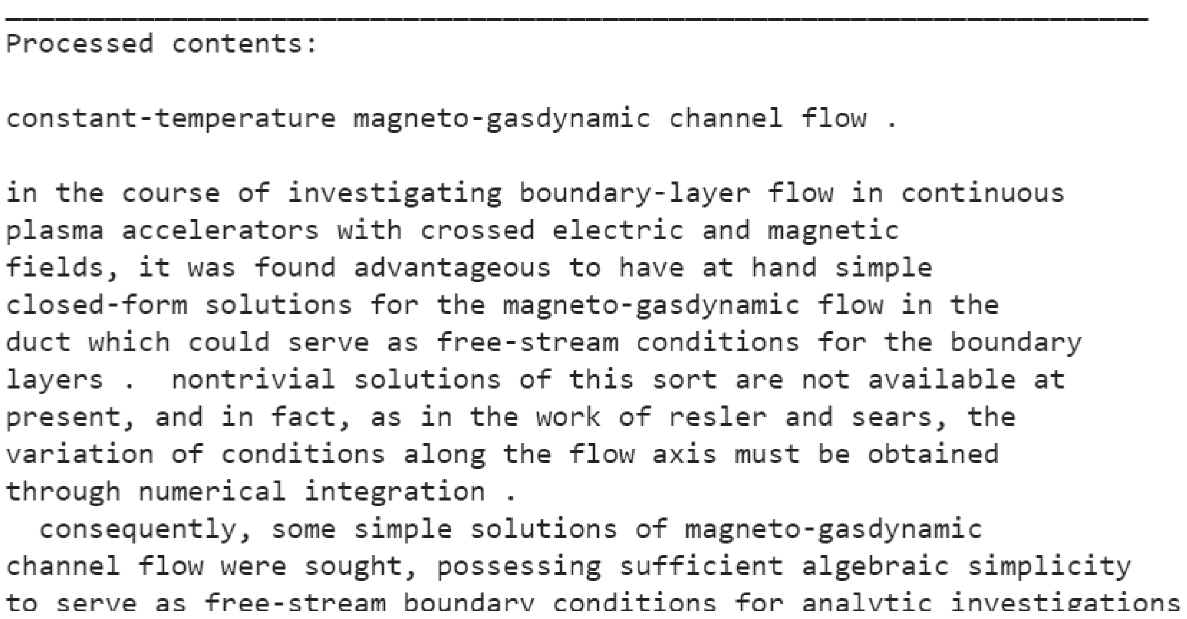
Link to the GitHub repository-:https://github.com/krishnasai20442/CSE508_Winter2023_A1_39
Link to the Google Colab notebook-:https://colab.research.google.com/drive/173NCL7z-crKP4O-NmAL_ao1BYMdtTz9k?authuser=1#scrollTo=aTSPGeuOwiZs

Q1-:
For this question, we assumed all the data in the given files was only strings, i.e, no tables or images.
The library NLTK was used to remove stopwords.
To extract the relevant text, we used the built-in string.find() function in python
Preprocessing was also done using inbuilt python functions such as string.lower()



```
Doc no.  0
Original contents:
<DOC>
<DOCNO>
34
</DOCNO>
<TITLE>
constant-temperature magneto-gasdynamic channel flow .
</TITLE>
<AUTHOR>
kerrebrock,j.p. and marble,f.e.
</AUTHOR>
<BIBLIO>
j. ae. scs. 27, 1960, 78.
</BIBLIO>
<TEXT>
in the course of investigating boundary-layer flow in continuous
plasma accelerators with crossed electric and magnetic
fields, it was found advantageous to have at hand simple
closed-form solutions for the magneto-gasdynamic flow in the
duct which could serve as free-stream conditions for the boundary
```

Group 39



```
________________________________________________________________
Processed contents:

constant-temperature magneto-gasdynamic channel flow .

in the course of investigating boundary-layer flow in continuous
plasma accelerators with crossed electric and magnetic
fields, it was found advantageous to have at hand simple
closed-form solutions for the magneto-gasdynamic flow in the
duct which could serve as free-stream conditions for the boundary
layers .  nontrivial solutions of this sort are not available at
present, and in fact, as in the work of resler and sears, the
variation of conditions along the flow axis must be obtained
through numerical integration .
  consequently, some simple solutions of magneto-gasdynamic
channel flow were sought, possessing sufficient algebraic simplicity
to serve as free-stream boundary conditions for analytic investigations
```

Above are some screenshots of the original contents and processed contents of some of the files of the dataset

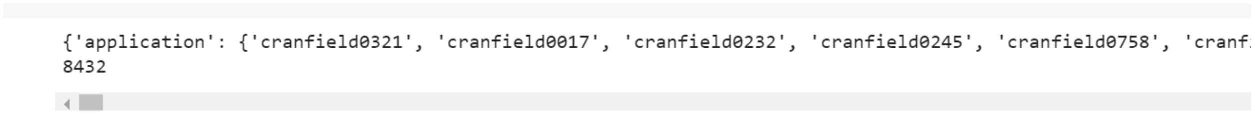
Group 39

Q2-:
Methodology:
Preprocessing the Documents:
The first step was to preprocess the documents. The preprocessing involved converting all text to lowercase, removing punctuation and non-alphabetic characters.

Creating the Inverted Index:
We created an inverted index from the preprocessed documents. The inverted index was a dictionary that maps each term to a list of document ids where the term occurs. The inverted index was created by looping over all the preprocessed documents and adding each term to the inverted index with its corresponding document id.

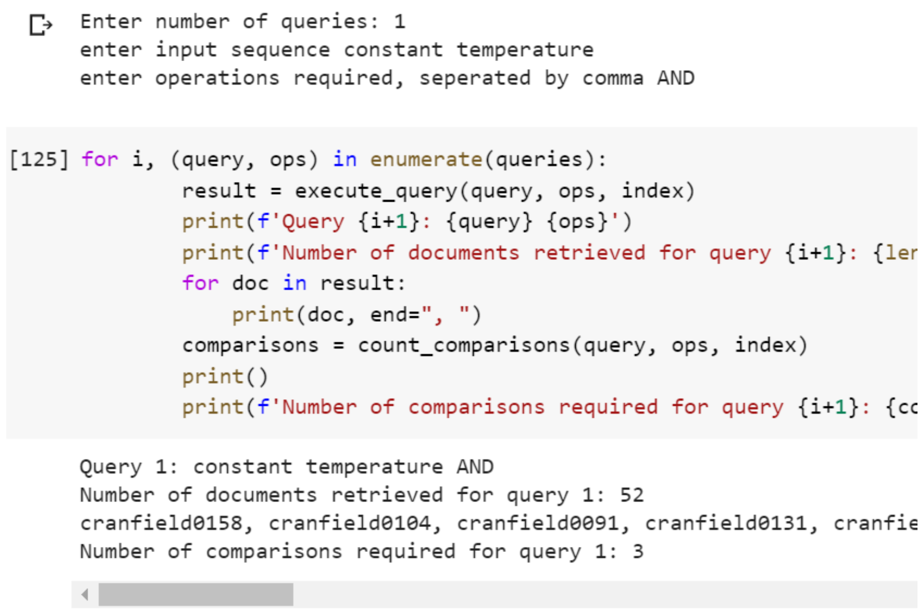
Querying the Documents:
We implemented support for the following Boolean operations: AND, OR, NOT. Queries were parsed and executed on the inverted index. To optimize query execution, we used a merge algorithm that retrieves the document ids that satisfy the query. We also implemented a function to count the number of comparisons required to execute the query.

Saving the Inverted Index:
We used Python's pickle module to save and load the inverted index.



```
{'application': {'cranfield0321', 'cranfield0017', 'cranfield0232', 'cranfield0245', 'cranfield0758', 'cranf
8432
```

A screenshot of the index created, as well as its length



```
Enter number of queries: 1
enter input sequence constant temperature
enter operations required, seperated by comma AND
```

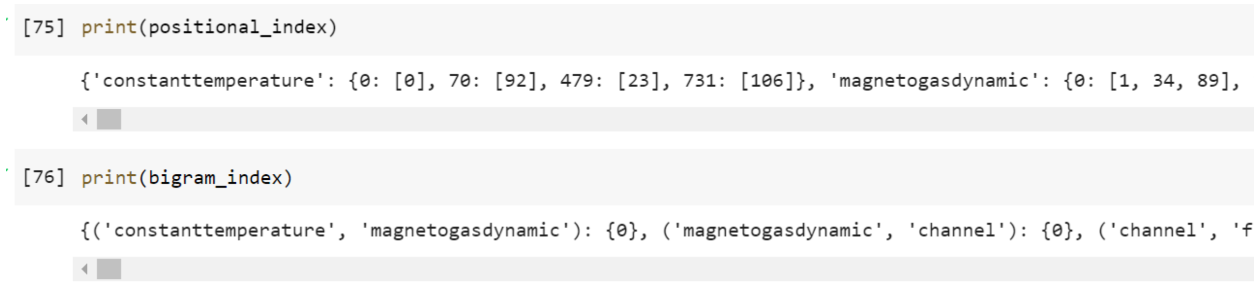
```
[125] for i, (query, ops) in enumerate(queries):
          result = execute_query(query, ops, index)
          print(f'Query {i+1}: {query} {ops}')
          print(f'Number of documents retrieved for query {i+1}: {ler
          for doc in result:
              print(doc, end=", ")
          comparisons = count_comparisons(query, ops, index)
          print()
          print(f'Number of comparisons required for query {i+1}: {cc
```

```
Query 1: constant temperature AND
Number of documents retrieved for query 1: 52
cranfield0158, cranfield0104, cranfield0091, cranfield0131, cranfie
Number of comparisons required for query 1: 3
```

A screenshot of an example execution of a simple query.

Group 39

Q3-:
To prepare data for indexing and subsequent searching, we first populated a list with the text of each document after preprocessing.
Then we defined functions to create the bigram and positional indices from scratch, as well as to search the indices.
The indices were initialized and then stored using the pickle module.



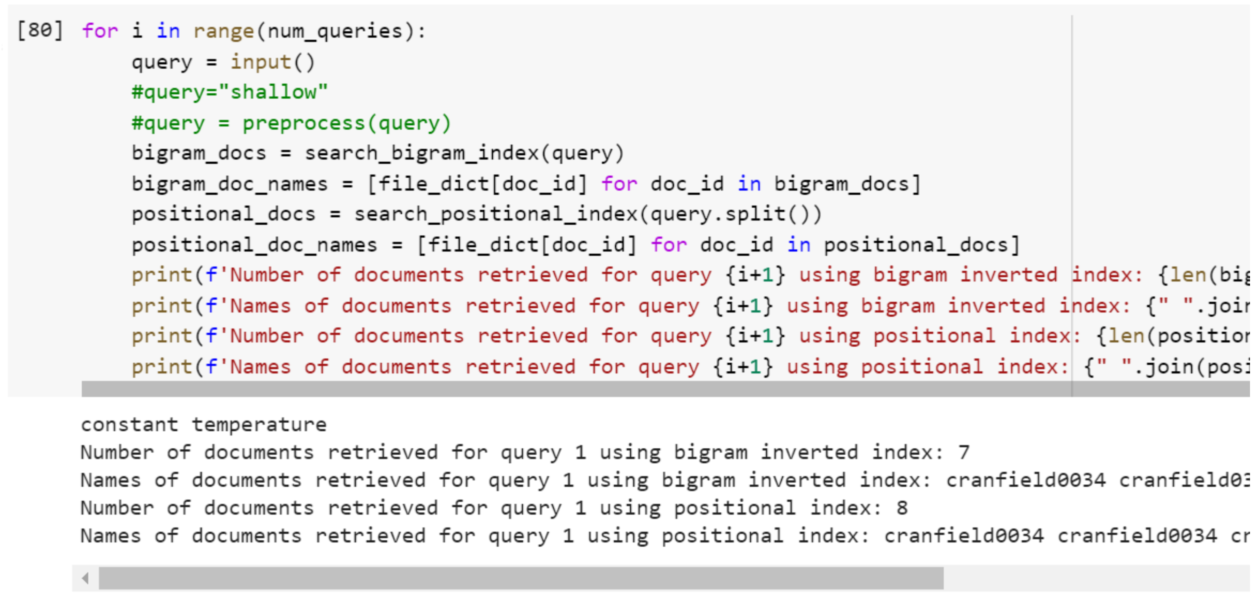
```
[75] print(positional_index)

{'constanttemperature': {0: [0], 70: [92], 479: [23], 731: [106]}, 'magnetogasdynamic': {0: [1, 34, 89],

[76] print(bigram_index)

{('constanttemperature', 'magnetogasdynamic'): {0}, ('magnetogasdynamic', 'channel'): {0}, ('channel', 'f
```

A screenshot showing the contents of the constructed indices



```
[80] for i in range(num_queries):
         query = input()
         #query="shallow"
         #query = preprocess(query)
         bigram_docs = search_bigram_index(query)
         bigram_doc_names = [file_dict[doc_id] for doc_id in bigram_docs]
         positional_docs = search_positional_index(query.split())
         positional_doc_names = [file_dict[doc_id] for doc_id in positional_docs]
         print(f'Number of documents retrieved for query {i+1} using bigram inverted index: {len(big
         print(f'Names of documents retrieved for query {i+1} using bigram inverted index: {" ".joir
         print(f'Number of documents retrieved for query {i+1} using positional index: {len(positior
         print(f'Names of documents retrieved for query {i+1} using positional index: {" ".join(posi

constant temperature
Number of documents retrieved for query 1 using bigram inverted index: 7
Names of documents retrieved for query 1 using bigram inverted index: cranfield0034 cranfield03
Number of documents retrieved for query 1 using positional index: 8
Names of documents retrieved for query 1 using positional index: cranfield0034 cranfield0034 cr
```

A screenshot showing an example execution of the index search for phrase queries.
The bigram inverted index was faster to create and required less storage than the positional index.
The positional index, on the other hand, was slower to create and required more storage.
The positional index is less likely to give false positives as it is more accurate.