IR Assignment Report 1

Q1

Approach-

Folder Structure:

Organize text files in a dedicated folder named "text_files."

Preprocessing Functions:

1. Implement a set of functions to perform specific preprocessing tasks:
2. convert_to_lowercase: Converts text to lowercase.
3. tokenize_words: Tokenizes text into words using NLTK's word_tokenize.
4. filter_stopwords_from_tokens: Removes common stopwords from the tokenized list.
5. remove_punctuation_from_tokens: Eliminates punctuation from the list of tokens.
6. filter_non_blank_tokens: Filters out non-blank tokens.
7. preprocess_text: Integrates the above functions to preprocess a given text.

File Mapping:

1. Create a file mapping to associate each file index with its corresponding file path.
2. Text Files Preprocessing:
3. Use preprocess_text_files_folder function to preprocess all text files in the "text_files" folder.
4. Generate a list of preprocessed tokens and a file mapping for future reference.

Methodology:

1. Collect text data and organize it in the "text_files" folder.
2. Function Implementation:
3. Implement text preprocessing functions for converting to lowercase, tokenization, stopwords removal, punctuation removal, and filtering non-blank tokens.
4. File Mapping Creation:
5. Create a mapping between file indices and file paths using create_file_mapping function.
6. Text Files Processing:
7. Utilize preprocess_text_files_folder to preprocess all text files in the folder.
8. Obtain preprocessed tokens and a file mapping.

Results Display:

```
File: text_files/file1.txt
Content before preprocessing:
Loving these vintage springs on my vintage strat. They have a good tension and great stability. If you are floating your bridge
and want the most out of your springs than these are the way to go.

Content after preprocessing:
['bridge', 'floating', 'go', 'good', 'great', 'loving', 'springs', 'stability', 'strat', 'tension', 'vintage', 'want', 'way']

--------------------------------------------------

File: text_files/file10.txt
Content before preprocessing:
Awesome stand!

Tip: The bottom part that supports the guitar had a weird angle when arrived, making the guitar slide back, becoming almost 10
0% on a vertical.
To solve this, I assembled the product and the put a some pressure on the support frame, making it bend a little. Now my guitar
sits perfectly. Check photos!

Content after preprocessing:
['100', 'almost', 'angle', 'arrived', 'assembled', 'awesome', 'back', 'becoming', 'bend', 'bottom', 'check', 'frame', 'guitar',
'little', 'making', 'part', 'perfectly', 'photos', 'pressure', 'product', 'put', 'sits', 'slide', 'solve', 'stand', 'support',
'supports', 'tip', 'vertical', 'weird']

--------------------------------------------------

File: text_files/file100.txt
Content before preprocessing:
This amp is the real deal.  Great crunch and gain tones and with some tweaking, not half bad clean"ish" tones.  I've played thi
s through the two 8" Orange cabs (had to get those too as they were just TOO cool ((and cute)) and not crazy money) and the sou
nd is very pleasing and revealing for a practice amp.  I primarily play it through my Blackstar stack that I've fitted with Cel
estion V30s... Wow...there it is~!!!  You would never know this thing was such a tone monster... Even with just a few knobs i
t's easy to get lost for hours playing this thing.  My favorite match is with my Chapman ML-1 Hotrod...which only has a volume
"tone" control (EVH fans get this).  Not a lot of mucking around with too many knobs or too many options on either the guitar o
r this amp... Just tone up and go.  I see the Micro Dark just came out...that's probably next~!  Higher gain, buffered effects
loop and speaker emu at the headphone out for recording direct (if that's your thing).

Content after preprocessing:
['8', 'amp', 'around', 'bad', 'blackstar', 'buffered', 'cabs', 'came', 'celestion', 'chapman', 'clean', 'control', 'cool', 'cra
zy', 'crunch', 'cute', 'dark', 'deal', 'direct', 'easy', 'effects', 'either', 'emu', 'even', 'evh', 'fans', 'favorite', 'fitte
d', 'gain', 'get', 'go', 'great', 'guitar', 'half', 'headphone', 'higher', 'hotrod', 'hours', 'is', 'ish', 'knobs', 'know', 'lo
op', 'lost', 'lot', 'many', 'match', 'micro', 'ml1', 'money', 'monster', 'mucking', 'never', 'next', 'options', 'orange', 'pla
y', 'played', 'playing', 'pleasing', 'practice', 'primarily', 'probably', 'real', 'recording', 'revealing', 's', 'see', 'soun
d', 'speaker', 'stack', 'thing', 'tone', 'tones', 'tweaking', 'two', 'v30s', 've', 'volume', 'would', 'wow']

--------------------------------------------------

File: text_files/file101.txt
Content before preprocessing:
You can do a lot with this mixer. its great for podcasting. has 4 outputs that can be used to monitor, record, cue audio...The
mute to 3/4 figure on every channel is fantastic and the three source switch to headphone/control room is a must for podcastin
g. Also has aux return inputs that can be used as extra stereo inputs and be volumed by the aux return knobs.

Only thing I didn't like about this mixer is the XLR outputs in back that require adaptors to use with RCA or 1/4 plugs. get th
e adaptors with it

Content after preprocessing:
['14', '34', '4', 'adaptors', 'also', 'audio', 'aux', 'back', 'channel', 'cue', 'every', 'extra', 'fantastic', 'figure', 'get',
'great', 'headphonecontrol', 'inputs', 'knobs', 'like', 'lot', 'mixer', 'monitor', 'must', 'mute', 'nt', 'outputs', 'plugs', 'p
odcasting', 'rca', 'record', 'require', 'return', 'room', 'source', 'stereo', 'switch', 'thing', 'three', 'use', 'used', 'volum
ed', 'xlr']

--------------------------------------------------

File: text_files/file102.txt
Content before preprocessing:
<div id="video-block-R2VOQ5CBZHFCKL" class="a-section a-spacing-small a-spacing-top-mini video-block"></div><input type="hidde
n" name="" value="https://images-na.ssl-images-amazon.com/images/I/E1%2B6MhK2MfS.mp4" class="video-url"><input type="hidden" na
me="" value="https://images-na.ssl-images-amazon.com/images/I/21-Jk5lxqsS.png" class="video-slate-img-url"> This mic is a
BOSS and a lot better than just about any other mic I've seen or used out-of-the-box for voice over. It sounds great even befor
e processing, and with some compression and EQ, it sounds fantastic. It rejects a ton of background noise and sounds amazing.

It runs very HOT! So you'll want clean pre-amping as to get a clean signal, but this is an amazing mic for the price.

Content after preprocessing:
['2b6mhk2mfsmp4', 'amazing', 'asection', 'aspacingsmall', 'aspacingtopmini', 'background', 'better', 'boss', 'class', 'clean',
'compression', 'div', 'eq', 'even', 'fantastic', 'get', 'great', 'hidden', 'hot', 'https', 'id', 'imagesnasslimagesamazoncomima
gesi21jk5lxqsspng', 'imagesnasslimagesamazoncomimagesie1', 'input', 'll', 'lot', 'mic', 'name', 'nbsp', 'noise', 'outofthebox',
'preamping', 'price', 'processing', 'rejects', 'runs', 'seen', 'signal', 'sounds', 'ton', 'type', 'used', 'value', 've', 'video
block', 'videoblockr2voq5cbzhfckl', 'videoslateimgurl', 'videourl', 'voice', 'want']

--------------------------------------------------
```

Q2

Approach:

In the initial preprocessing phase (Q1), we covered essential steps like lowercase conversion and tokenization. Creating a Unigram Inverted Index involved mapping terms efficiently, iterating over documents, and saving the index. The subsequent search engine implementation included functions for AND, OR, AND NOT, and OR NOT operations. Users input queries, and the system executed the selected operation, returning relevant documents.

Methodology:

a. Create a unigram inverted index to efficiently map terms to their positions in the text documents.
b. Iterate over each document in the collection, tokenize the document, and record term frequencies and positions.
c. Use data structures like dictionaries to build the inverted index.
d. Implement a search engine that utilizes the unigram inverted index to perform search operations.
e. Provide a user interface for users to input search queries and specify search operations.
f. Implement functions for different search operations such as AND, OR, AND NOT, and OR NOT.
g. Process user queries, execute the appropriate search operation using the inverted index, and return the matching documents

Results:

```
Enter the query sentence: Coffee brewing techniques in cookbook
Enter the operation sequence: and, or not,or
['brewing', 'coffee', 'cookbook', 'techniques']
['and', 'or not', 'or']
Number of documents matched: 999
Documents:

file1.txt
file10.txt
file100.txt
file101.txt
file102.txt
file103.txt
file104.txt
file105.txt
file106.txt
file107.txt
file108.txt
file109.txt
```

```
Enter the number of queries: 2
Enter the query sentence: Car bag in a canister
Enter the operation sequence: or, and not
['bag', 'canister', 'car']
['or', 'and not']
Number of documents matched: 25
Documents:

file118.txt
file3.txt
file313.txt
file363.txt
file404.txt
file459.txt
file466.txt
file573.txt
file665.txt
file682.txt
file686.txt
```

Q3

Approach:

The text documents undergo preprocessing, including lowercase conversion, tokenization, stopwords removal, and filtering non-blank tokens. These preprocessed tokens are organized into a list of lists representing document tokens. A positional index is created by mapping tokens to positions within documents, and the resulting index is efficiently saved to a pickle file. The system enables users to input and search phrase queries, returning a set of documents that precisely match the query.

Assumptions:

1. Assumption is made that the length of the input phrase query should not exceed five words.
2. Queries longer than five words may lead to decreased efficiency or relevance in the current implementation.
3. Assumption is made that users provide phrase queries in a standard format, with each word separated by a space.

Methodology:

1. Implement text preprocessing functions to clean and prepare the text data for indexing and searching.

2. Convert text to lowercase, tokenize into words, remove stopwords, and filter non-blank tokens.
3. Develop a function to create a positional index from the preprocessed tokens list.
4. Record the positions of each token in each document for efficient retrieval.
5. Serialize the positional index to a pickle file for efficient storage and reloading.
6. Implement a function to execute phrase queries using the positional index.
7. Allow users to input phrase queries, preprocess the queries, and execute the search using the positional index.
8. Return the set of documents that match the phrase query.

Results:

```
Enter the number of queries: 2
Enter the phrase query: Car bag in a canister
Number of documents retrieved for query 1 using positional index: 0
Names of documents retrieved for query 1 using positional index: []
Enter the phrase query: Coffee brewing techniques in cookbook
Number of documents retrieved for query 2 using positional index: 297
Names of documents retrieved for query 2 using positional index: ['File 3.txt', 'File 517.txt', 'File 520.txt', 'File 9.txt',
'File 11.txt', 'File 525.txt', 'File 527.txt', 'File 16.txt', 'File 529.txt', 'File 18.txt', 'File 19.txt', 'File 530.txt', 'Fi
le 532.txt', 'File 535.txt', 'File 25.txt', 'File 27.txt', 'File 539.txt', 'File 31.txt', 'File 543.txt', 'File 34.txt', 'File
38.txt', 'File 551.txt', 'File 555.txt', 'File 44.txt', 'File 556.txt', 'File 557.txt', 'File 47.txt', 'File 48.txt', 'File 49.
txt', 'File 561.txt', 'File 51.txt', 'File 52.txt', 'File 53.txt', 'File 566.txt', 'File 55.txt', 'File 567.txt', 'File 58.tx
t', 'File 570.txt', 'File 60.txt', 'File 572.txt', 'File 63.txt', 'File 64.txt', 'File 65.txt', 'File 576.txt', 'File 67.txt',
'File 577.txt', 'File 580.txt', 'File 581.txt', 'File 586.txt', 'File 75.txt', 'File 588.txt', 'File 77.txt', 'File 589.txt',
'File 590.txt', 'File 594.txt', 'File 595.txt', 'File 84.txt', 'File 85.txt', 'File 598.txt', 'File 89.txt', 'File 601.txt', 'F
ile 602.txt', 'File 93.txt', 'File 94.txt', 'File 609.txt', 'File 100.txt', 'File 612.txt', 'File 613.txt', 'File 103.txt', 'Fi
le 104.txt', 'File 105.txt', 'File 106.txt', 'File 107.txt', 'File 614.txt', 'File 617.txt', 'File 624.txt', 'File 114.txt', 'F
```