Q1. Data Preprocessing

Following is the content of 5 sample files before and after performing each preprocessing operation:

Sample 1:

Original content of file903.txt:
This guitar is perfect in every way! It's so easy to play and sounds fantastic! Perfectly balanced with zero neck dive. The bomb inlays just look cool! I definitely plan on buying more Hardluck Kings guitars!

Performing operation a. Lowercase the text
this guitar is perfect in every way! it's so easy to play and sounds fantastic! perfectly balanced with zero neck dive. the bomb inlays just look cool! i definitely plan on buying more hardluck kings guitars!

Performing operation b. Perform tokenization
['this', 'guitar', 'is', 'perfect', 'in', 'every', 'way', '!', 'it', "'s", 'so', 'easy', 'to', 'play', 'and', 'sounds', 'fantastic', '!', 'perfectly', 'balanced', 'with', 'zero', 'neck', 'dive', '.', 'the', 'bomb', 'inlays', 'just', 'look', 'cool', '!', 'i', 'definitely', 'plan', 'on', 'buying', 'more', 'hardluck', 'kings', 'guitars', '!']

Performing operation c. Remove stopwords
['guitar', 'perfect', 'every', 'way', '!', "'s", 'easy', 'play', 'sounds', 'fantastic', '!', 'perfectly', 'balanced', 'zero', 'neck', 'dive', '.', 'bomb', 'inlays', 'look', 'cool', '!', 'definitely', 'plan', 'buying', 'hardluck', 'kings', 'guitars', '!']

Performing operation d. Remove punctuations
['guitar', 'perfect', 'every', 'way', "'s", 'easy', 'play', 'sounds', 'fantastic', 'perfectly', 'balanced', 'zero', 'neck', 'dive', 'bomb', 'inlays', 'look', 'cool', 'definitely', 'plan', 'buying', 'hardluck', 'kings', 'guitars']

Performing operation e. Remove blank space tokens
['guitar', 'perfect', 'every', 'way', "'s", 'easy', 'play', 'sounds', 'fantastic', 'perfectly', 'balanced', 'zero', 'neck', 'dive', 'bomb', 'inlays', 'look', 'cool', 'definitely', 'plan', 'buying', 'hardluck', 'kings', 'guitars']

Preprocessed content of file903.txt:
guitar perfect every way 's easy play sounds fantastic perfectly balanced zero neck dive bomb inlays look cool definitely plan buying hardluck kings guitars

Sample 2:

Original content of file902.txt:
Beautiful, well-made strap. Matches my guitar nicely. Perfect.

Performing operation a. Lowercase the text
beautiful, well-made strap. matches my guitar nicely. perfect.

Performing operation b. Perform tokenization
['beautiful', ',', 'well-made', 'strap', '.', 'matches', 'my', 'guitar', 'nicely', '.', 'perfect', '.']

Performing operation c. Remove stopwords
['beautiful', ',', 'well-made', 'strap', '.', 'matches', 'guitar', 'nicely', '.', 'perfect', '.']

Performing operation d. Remove punctuations
['beautiful', 'well-made', 'strap', 'matches', 'guitar', 'nicely', 'perfect']

Performing operation e. Remove blank space tokens
['beautiful', 'well-made', 'strap', 'matches', 'guitar', 'nicely', 'perfect']

Preprocessed content of file902.txt:
beautiful well-made strap matches guitar nicely perfect

Sample 3:

Original content of file199.txt:
Amazing and highly different Wah from ibanez,This is the WAH YA WANT!!,too many reasons!,Just check it out!!!

Performing operation a. Lowercase the text
amazing and highly different wah from ibanez,this is the wah ya want!!,too many reasons!,just check it out!!!

Performing operation b. Perform tokenization
['amazing', 'and', 'highly', 'different', 'wah', 'from', 'ibanez', ',', 'this', 'is', 'the', 'wah', 'ya', 'want', '!', '!', ',', 'too', 'many', 'reasons', '!', ',', 'just', 'check', 'it', 'out', '!', '!', '!']

Performing operation c. Remove stopwords
['amazing', 'highly', 'different', 'wah', 'ibanez', ',', 'wah', 'ya', 'want', '!', '!', ',', 'many', 'reasons', '!', ',', 'check', '!', '!', '!']

Performing operation d. Remove punctuations
['amazing', 'highly', 'different', 'wah', 'ibanez', 'wah', 'ya', 'want', 'many', 'reasons', 'check']

Performing operation e. Remove blank space tokens
['amazing', 'highly', 'different', 'wah', 'ibanez', 'wah', 'ya', 'want', 'many', 'reasons', 'check']

Preprocessed content of file199.txt:
amazing highly different wah ibanez wah ya want many reasons check

Sample 4:

Original content of file297.txt:
The quality is excellent

Performing operation a. Lowercase the text
the quality is excellent

Performing operation b. Perform tokenization
['the', 'quality', 'is', 'excellent']

Performing operation c. Remove stopwords
['quality', 'excellent']

Performing operation d. Remove punctuations
['quality', 'excellent']

Performing operation e. Remove blank space tokens
['quality', 'excellent']

Preprocessed content of file297.txt:
quality excellent

Sample 5:

Original content of file321.txt:
Good quality :) perfect for my office/studio setup!

Performing operation a. Lowercase the text
good quality :) perfect for my office/studio setup!

Performing operation b. Perform tokenization
['good', 'quality', ':', ')', 'perfect', 'for', 'my', 'office/studio', 'setup', '!']

Performing operation c. Remove stopwords
['good', 'quality', ':', ')', 'perfect', 'office/studio', 'setup', '!']

Performing operation d. Remove punctuations
['good', 'quality', 'perfect', 'office/studio', 'setup']

Performing operation e. Remove blank space tokens
['good', 'quality', 'perfect', 'office/studio', 'setup']

Preprocessed content of file321.txt:
good quality perfect office/studio setup

Q2. Unigram Inverted Index and Boolean Queries

Construction of Unigram Inverted Index

The data structure used to create Unigram Inverted Index is a dictionary (‘inverted_index’).

The keys represent terms found in documents, and the values represent lists of filenames in which those terms appear.

Assumption

It has been assumed that it is not mandatory to store the frequency of documents as well.

Unigram refers to continuous sequence of 1 token in the document.

Results

```
2
Car bag in a canister
OR, AND NOT

Query: car OR bag AND NOT canister
Number of documents retrieved: 31
Names of the documents retrieved ['file780.txt', 'file404.txt', 'file942.txt', 'file864.txt', 'file264.txt', 'file459.txt', 'fi

Coffee brewing techniques in cookbook
AND, OR NOT, OR

Query: coffee AND brewing OR NOT techniques OR cookbook
Number of documents retrieved: 999
Names of the documents retrieved ['file362.txt', 'file71.txt', 'file376.txt', 'file700.txt', 'file226.txt', 'file827.txt', 'fil
```

Q3. Positional Index and Phrase Queries

Construction of Positional Index

The data structure used to create Positional Index is a dictionary (‘positional_index’) with the following structure:

Key: Each unique term found in the documents.
Value: A nested dictionary where:
    Key: Filename
    Value: List of positions where the term appears in the document.

Assumption

Length of input sequence <= 5.

Results

```
4

Good

Number of documents retrieved for the phrase 'good' using positional index: 204
Names of documents retrieved for the phrase 'good' using positional index: ['file1.txt', 'file103.txt', 'file106.txt', 'file110.

Good Quality

Number of documents retrieved for the phrase 'good quality' using positional index: 17
Names of documents retrieved for the phrase 'good quality' using positional index: ['file118.txt', 'file154.txt', 'file179.txt',

kinda flimsy oit Job

Number of documents retrieved for the phrase 'kinda flimsy oit job' using positional index: 1
Names of documents retrieved for the phrase 'kinda flimsy oit job' using positional index: ['file15.txt']
```