CSE508 Information Retrieval

Winter 2024
Assignment-1

Vasanth Kumar 2020386

Github link : **https://github.com/vasanth20386/CSE508_Winter2024_A1_2020386**

Q1. Data Preprocessing

Preprocessing Steps

The function preprocess_files takes input file path, output file path, printing performed preprocessing tasks for the first five files after each operation.

- **Reading File Content**: Open and read the content of the input file.
- **Lowercasing**: Convert all text to lowercase to normalize the case.
- **Tokenization**: Split the text into individual words or tokens using NLTK's word_tokenize.
- **Removing Stopwords**: Filter out English stopwords using NLTK's stopwords list.
- **Removing Punctuation**: Remove punctuation from tokens to clean the text further.
- **Removing Blank Spaces**: Eliminate tokens that are just blank spaces.
- **Saving Preprocessed Text**: Write the cleaned and preprocessed text to the output file.

Output of 5 sample files before and after performing each operation: (Here i present 3 sample files output)

Original text from text_files/file502.txt:

Kit is awesome. I play in my garage just for personal enjoyment not for performances or anything. Once you take the time to break down all the settings, your able to dial in pretty much any kit and sound. With the expansion options and the relatively inexpensive parts expanding is easy and fun. After a few weeks of daily use for at least an hour a day it still looks and plays beautifully. Overall one of the best purchases I could have made.

After lowercase: kit is awesome. i play in my garage just for personal enjoyment not for performances or anything. once you take the time to break down all the settings, your able to dial in pretty much any kit and sound. with the expansion options and the relatively inexpensive parts expanding is easy and fun. after a few weeks of daily use for at least an hour a day it still looks and plays beautifully. overall one of the best purchases i could have made.

After tokenization: ['kit', 'is', 'awesome', '.', 'i', 'play', 'in', 'my', 'garage', 'just', 'for', 'personal', 'enjoyment', 'not', 'for', 'performances', 'or', 'anything', '.', 'once', 'you', 'take', 'the', 'time', 'to', 'break', 'down', 'all', 'the', 'settings', ',', 'your', 'able', 'to', 'dial', 'in', 'pretty', 'much', 'any', 'kit', 'and', 'sound', '.', 'with', 'the', 'expansion', 'options', 'and', 'the', 'relatively']

After removing stopwords: ['kit', 'awesome', '.', 'play', 'garage', 'personal', 'enjoyment', 'performances', 'anything', '.', 'take', 'time', 'break', 'settings', ',', 'able', 'dial', 'pretty', 'much', 'kit', 'sound', '.', 'expansion', 'options', 'relatively', 'inexpensive', 'parts', 'expanding', 'easy', 'fun', '.', 'weeks', 'daily', 'use', 'least', 'hour', 'day', 'still', 'looks', 'plays', 'beautifully', '.', 'overall', 'one', 'best', 'purchases', 'could', 'made', '.']

After removing punctuation: ['kit', 'awesome', 'play', 'garage', 'personal', 'enjoyment', 'performances', 'anything', 'take', 'time', 'break', 'settings', 'able', 'dial', 'pretty', 'much', 'kit', 'sound', 'expansion', 'options', 'relatively', 'inexpensive', 'parts', 'expanding', 'easy', 'fun', 'weeks', 'daily', 'use', 'least', 'hour', 'day', 'still', 'looks', 'plays', 'beautifully', 'overall', 'one', 'best', 'purchases', 'could', 'made']

After removing blank spaces: ['kit', 'awesome', 'play', 'garage', 'personal', 'enjoyment', 'performances', 'anything', 'take', 'time', 'break', 'settings', 'able', 'dial', 'pretty', 'much', 'kit', 'sound', 'expansion', 'options', 'relatively', 'inexpensive', 'parts', 'expanding', 'easy', 'fun', 'weeks', 'daily', 'use', 'least', 'hour', 'day', 'still', 'looks', 'plays', 'beautifully', 'overall', 'one', 'best', 'purchases', 'could', 'made']

--

Original text from text_files/file264.txt: I just tested this fog fluid with a 1byone 400W fogger. Two 30 second bursts were sufficient to create enough fog layers for a moody atmosphere in a 2 car garage. This being a hot space I was pleasantly surprised by how long the fog would linger. It would quickly rise to eye level and then just hang there. Another nice surprise was the odor- there is not much of it, but if you step in the middle of a thick pocket it smells like lavender (?) soap. Only downside is that the fog is not very dense.

After lowercase: i just tested this fog fluid with a 1byone 400w fogger. two 30 second bursts were sufficient to create enough fog layers for a moody atmosphere in a 2 car garage. this being a hot space i was pleasantly surprised by how long the fog would linger. it would quickly rise to eye level and then just hang there. another nice surprise was the odor- there is not much of it, but if you step in the middle of a thick pocket it smells like lavender (?) soap. only downside is that the fog is not very dense.

After tokenization: ['i', 'just', 'tested', 'this', 'fog', 'fluid', 'with', 'a', '1byone', '400w', 'fogger', '.', 'two', '30', 'second', 'bursts', 'were', 'sufficient', 'to', 'create', 'enough', 'fog', 'layers', 'for', 'a', 'moody', 'atmosphere', 'in', 'a', '2', 'car', 'garage', '.', 'this', 'being', 'a', 'hot', 'space', 'i', 'was', 'pleasantly', 'surprised', 'by', 'how', 'long', 'the', 'fog', 'would', 'linger', '.']

After removing stopwords: ['tested', 'fog', 'fluid', '1byone', '400w', 'fogger', '.', 'two', '30', 'second', 'bursts', 'sufficient', 'create', 'enough', 'fog', 'layers', 'moody', 'atmosphere', '2', 'car', 'garage', '.', 'hot', 'space', 'pleasantly', 'surprised', 'long', 'fog', 'would', 'linger', '.', 'would', 'quickly', 'rise', 'eye', 'level', 'hang', '.', 'another', 'nice', 'surprise', 'odor-', 'much', ',', 'step', 'middle', 'thick', 'pocket', 'smells', 'like']

After removing punctuation: ['tested', 'fog', 'fluid', '1byone', '400w', 'fogger', 'two', '30', 'second', 'bursts', 'sufficient', 'create', 'enough', 'fog', 'layers', 'moody', 'atmosphere', '2', 'car', 'garage', 'hot', 'space', 'pleasantly', 'surprised', 'long', 'fog', 'would', 'linger', 'would', 'quickly', 'rise', 'eye', 'level', 'hang', 'another', 'nice', 'surprise', 'odor', 'much', 'step', 'middle', 'thick', 'pocket', 'smells', 'like', 'lavender', 'soap', 'downside', 'fog', 'dense']

After removing blank spaces: ['tested', 'fog', 'fluid', '1byone', '400w', 'fogger', 'two', '30', 'second', 'bursts', 'sufficient', 'create', 'enough', 'fog', 'layers', 'moody', 'atmosphere', '2', 'car', 'garage', 'hot', 'space', 'pleasantly', 'surprised', 'long', 'fog', 'would', 'linger', 'would', 'quickly', 'rise', 'eye', 'level', 'hang', 'another', 'nice', 'surprise', 'odor', 'much', 'step', 'middle', 'thick', 'pocket', 'smells', 'like', 'lavender', 'soap', 'downside', 'fog', 'dense']

Original text from text_files/file270.txt: Do not let the low price fool you! This is an incredible device with the free mixing software. It has been years since i tracked anything. Back then everything was Solid State and Analogue. With this Scarlett Solo, a moderate computer, and a large monitor (i stress large because there is too much control with the software), you can really lay down some serious tracks. So if you are new or getting back to it, GET IT! Pro's Customer Support You Tube videos Con's TOO many features with the Scarl

After lowercase: do not let the low price fool you! this is an incredible device with the free mixing software. it has been years since i tracked anything. back then everything was solid state and analogue. with this scarlett solo, a moderate computer, and a large monitor (i stress large because there is too much control with the software), you can really lay down some serious tracks. so if you are new or getting back to it, get it! pro's customer support you tube videos con's too many features with the scarl

After tokenization: ['do', 'not', 'let', 'the', 'low', 'price', 'fool', 'you', '!', 'this', 'is', 'an', 'incredible', 'device', 'with', 'the', 'free', 'mixing', 'software', '.', 'it', 'has', 'been', 'years', 'since', 'i', 'tracked', 'anything', '.', 'back', 'then', 'everything', 'was', 'solid', 'state', 'and', 'analogue', '.', 'with', 'this', 'scarlett', 'solo', ',', 'a', 'moderate', 'computer', ',', 'and', 'a', 'large']

After removing stopwords: ['let', 'low', 'price', 'fool', '!', 'incredible', 'device', 'free', 'mixing', 'software', '.', 'years', 'since', 'tracked', 'anything', '.', 'back', 'everything', 'solid', 'state', 'analogue', '.', 'scarlett', 'solo', ',', 'moderate', 'computer', ',', 'large', 'monitor', '(', 'stress', 'large', 'much', 'control', 'software', ')', ',', 'really', 'lay', 'serious', 'tracks', '.', 'new', 'getting', 'back', ',', 'get', '!', "pro's"]

After removing punctuation: ['let', 'low', 'price', 'fool', 'incredible', 'device', 'free', 'mixing', 'software', 'years', 'since', 'tracked', 'anything', 'back', 'everything', 'solid', 'state', 'analogue', 'scarlett', 'solo', 'moderate', 'computer', 'large', 'monitor', 'stress', 'large', 'much', 'control', 'software', 'really', 'lay', 'serious', 'tracks', 'new', 'getting', 'back', 'get', 'pros', 'customer', 'support', 'tube', 'videos', 'cons', 'many', 'features', 'scarlet', 'pro', 'software', 'instructional', 'downloadable']

After removing blank spaces: ['let', 'low', 'price', 'fool', 'incredible', 'device', 'free', 'mixing', 'software', 'years', 'since', 'tracked', 'anything', 'back', 'everything', 'solid', 'state', 'analogue', 'scarlett', 'solo', 'moderate', 'computer', 'large', 'monitor', 'stress', 'large', 'much', 'control', 'software', 'really', 'lay', 'serious', 'tracks', 'new', 'getting', 'back', 'get', 'pros', 'customer', 'support', 'tube', 'videos', 'cons', 'many', 'features', 'scarlet', 'pro', 'software', 'instructional', 'downloadable']

Q2. Unigram Inverted Index and Boolean Queries

a)

```
def create_unigram_inverted_index(directory):
```

The function create_unigram_inverted_index reads each file in a directory, splits it into words, and updates an inverted index dictionary. The dictionary uses words as keys and lists of filenames as values. If a word is encountered for the first time, it is added to the dictionary. If the word already exists, it is checked to avoid duplicates.

b) Using Python's pickle module to save and load the unigram inverted index.

```
def save_inverted_index(index, save_path):
```

```
def load_inverted_index(load_path):
```

c) Created functions to support for the following operations:
a. T1 AND T2
b. T1 OR T2
c. T1 AND NOT T2
d. T1 OR NOT T2

```
def and_operation(set1, set2):
def or_operation(set1, set2):
def and_not_operation(set1, set2):
def or_not_operation(set1, set2, all_documents):
```

Other sections

```
def preprocess_text(text):
```

This function preprocesses the query like in Q1

```
def execute_query(query_tokens, operations, inverted_index, all_documents):
```

The execute_query function processes a list of query tokens and a series of operations. It converts each token into a set of documents (using an inverted index) that contain the token. It then sequentially applies the specified logical operations to these sets, computing the final set of documents that satisfy the entire query.

```
def reconstruct_query_with_operations(original_query, operations):
```

reconstruct_query_with_operations, aims to rebuild the query string from preprocessed terms and specified operations. It checks if the number of operations is appropriate for the number of terms; if not, it reports an issue.

```
def p_execute_queries(queries, inverted_index, all_documents):
```

p_and_execute_queries arranges the process by taking a list of queries and their associated logical operations, preprocesses the queries, executes them using the inverted index, and gathers the results. If the number of operations doesn't match the number of terms correctly, it reports an error.

Output

```
Query 1: car OR bag AND NOT canister Number of documents retrieved for query 1: 31
Names of the documents retrieved for query 1: file118.txt, file166.txt,
file174.txt, file264.txt, file3.txt, file313.txt, file363.txt, file404.txt,
file459.txt, file466.txt, file542.txt, file573.txt, file665.txt, file682.txt,
file686.txt, file698.txt, file699.txt, file73.txt, file738.txt, file746.txt,
file780.txt, file797.txt, file860.txt, file863.txt, file864.txt, file886.txt,
file892.txt, file930.txt, file942.txt, file956.txt, file981.txt
```

Q3. Positional Index and Phrase Queries

```
def create_positional_index(directory):
```

a)

Positional Index Creation:

- The create_positional_index function creates a positional index, an inverted index mapping words to documents and recording their positions within the documents.
- The defaultdict from the collections module is used to create a nested dictionary structure where each word points to another dictionary, mapping filenames to lists of positions where that word occurs.
- We iterate over each file in the specified directory (preprocessed_directory), reading and splitting the text into words. For each word, it updates the positional index with the word's position in that document.

b)

```
def save_positional_index(index, save_path):
def load_positional_index(load_path):
```

Save_positional_index function serializes positional index to a file (positional_index.pkl) for disk storage and retrieval. Load_positional_index function deserializes index, restoring it to memory for search operations.

C) Phrase Query Execution

```
def exe_phrase_query(query_tokens, positional_index):
```

- The exe_phrase_query function takes a list of query tokens and a positional index as input to execute phrase queries. A phrase query searches for documents where the query tokens appear in the exact sequence specified.
- The function first checks if the query tokens list is empty and returns an empty list if true.
- If the query consists of a single token, it directly returns the list of documents containing that token.
- For multi-token queries, the function iterates through the candidate documents of the first token and checks for the presence and correct sequence of subsequent tokens within each document. It does this by comparing the positions of the first token with the expected positions of the following tokens.
- Documents where the tokens appear in the exact sequence are added to the list of valid documents, which is then returned.

Input

1
Sounds good

Output

Number of documents retrieved for query 1 using positional index: 4 Names of documents retrieved for query 1 using positional index: file160.txt, file907.txt, file16.txt, file526.txt