CSE508 Information Retrieval

Winter 2024
Assignment-1

Name: Aniketh 9th February Roll No: 2020360

Q1. Data Preprocessing

Overview:

In this task, we preprocess text data from a dataset by applying several common preprocessing steps. These steps include converting text to lowercase, tokenization, and removing stopwords, punctuation, and blank space tokens. The goal is to clean the text data and prepare it for further analysis or modelling tasks.

Methodologies:

For all the tasks, I used the nltk library in Python.

Lowercasing: Convert all text to lowercase to ensure consistency and reduce the complexity of the data.

Tokenization: Split the text into individual tokens or words. This step helps in further analysis by breaking down the text into meaningful units.

Stopword Removal: Remove common stopwords such as 'the', 'is', 'and', etc., as they do not contribute much to the meaning of the text.

Punctuation Removal: Remove punctuation marks from the text as they do not carry significant meaning in most analysis tasks.

Blank Space Token Removal: Remove any tokens that consist only of blank spaces to clean up the data further.

For each file in the directory, I use a preprocess file to read the contents of the file, preprocess that text using a function, and then save that to another file in another directory that keeps all the preprocessed files to be used in further questions.

Assumptions:

I assume that the dataset consists of plain text files with no specific formatting requirements. This leaves my processing sometimes with quotation marks or special symbols.

Results:

Apart from saving all the files, I print the contents of 5 files.

Below are the contents of 5 sample files before and after performing each operation:

Processing: file1.txt

Contents 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.

Contents after preprocessing:

loving vintage springs vintage strat good tension great stability floating bridge want springs way go

Processing: file10.txt

Contents 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 100% 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!

Contents after preprocessing:

awesome stand tip bottom part supports guitar weird angle arrived making guitar slide back becoming almost 100 vertical solve assembled product put pressure support frame making bend little guitar sits perfectly check photos

Processing: file100.txt

Contents 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 this 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 sound is very pleasing and revealing for a practice amp. I primarily play it through my Blackstar stack that I've fitted with Celestion V30s... Wow...there it is~!!! You would never know this thing was such a tone monster... Even with just a few knobs it'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 or 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).

Contents after preprocessing:

amp real deal great crunch gain tones tweaking half bad clean " ish " tones 've played two 8 " orange cabs get cool cute crazy money sound pleasing revealing practice amp primarily play blackstar stack 've fitted celestion v30s ... wow ... is~ would never know thing tone monster ... even knobs 's easy get lost hours playing thing favorite match chapman ml-1 hotrod ... volume `` tone " control evh fans get lot mucking around many knobs many options either guitar amp ... tone go see micro dark came ... 's probably next~ higher gain buffered effects loop speaker emu headphone recording direct 's thing

Processing: file101.txt

Contents 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 podcasting. 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 the adaptors with it

Contents after preprocessing:

lot mixer great podcasting 4 outputs used monitor record cue audio ... mute 3/4 figure every channel fantastic three source switch headphone/control room must podcasting also aux return inputs used extra stereo inputs volumed aux return knobs thing n't like mixer xlr outputs back require adaptors use rca 1/4 plugs get adaptors

Processing: file102.txt

Contents before preprocessing:

<input type="hidden" name="" value="https://images-na.ssl-images-amazon.com/images/I/E1%2B6MhK2MfS.mp4" class="video-url"><input type="hidden" name="" 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 before 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.

Contents after preprocessing:

div id= " video-block-r2voq5cbzhfckl " class= " a-section a-spacing-small a-spacing-top-mini video-block " /div input type= " hidden " name= " " value= " https //images-na.ssl-images-amazon.com/images/i/e1 2b6mhk2mfs.mp4 " class= " video-url " input type= " hidden " name= " " value= " https //images-na.ssl-images-amazon.com/images/i/21-jk5lxqss.png " class= " video-slate-img-url " nbsp mic boss lot better mic 've seen used out-of-the-box voice sounds great even processing compression eq sounds fantastic rejects ton background noise sounds amazing runs hot 'll want clean pre-amping get clean signal amazing mic price

Q2. Unigram Inverted Index and Boolean Queries

Methodology:

To create the Unigram Inverted Index, I followed these steps:

- Iterate through each preprocessed document in the dataset obtained from Q1.
- Tokenize the text and create a mapping of each unique token (unigram) to the set of document IDs where it appears.
- If a token already exists in the inverted index, append the current document ID to its set of document IDs. If not, create a new entry for the token with the current document ID.

Supported Operations:

- Implement functions for each boolean operation: AND, OR, AND NOT, OR NOT.
- Parse the input query string to identify the tokens and the boolean operations specified.
- Retrieve the document IDs associated with each token from the inverted index.
- Perform the boolean operation according to the query and return the resulting set of document IDs.

Input Preprocessing:

- Apply the same preprocessing steps as in Q1 to the input sequence before executing the queries.
- Preprocessing includes lowercasing, tokenization, stopword removal, punctuation removal, and blank space token removal.

Approach:

Creating Unigram Inverted Index:

I create a dictionary called inverted_index. Now iterating through all the .txt files, I preprocess and extract tokens from that. Further use these tokens to create a postings list if it doens't already exist. If already exists, then append the current doc/file to it.

After successfully creating this I save this dictionary in a pickle file.

Now for boolean operations I create different functions for 1) AND 2) OR 3) AND NOT 4) OR NOT

Using these functions I creates sets of documents which contain the two terms passed into the function and the inverted_index dictionary, to perform basic boolean operations and return a list of resulting document_ids/files.

As example, I perform the operations given in the assignment on terms ‘great’ and ‘stability’.

Now to further process generic and modular operations, I create different functions for same 4 boolean operations, where 1 term and result from previous operations is passed. Performing similar boolean operations on sets to obtain consecutive results.

Now, since I am using jupyter notebook, I prompt the user to input the text and operations which allow for input/output in the required format mentioned in the assignment.

Results:

```
Query 1: car OR bag AND NOT cannister
Number of documents retrieved for query 1: 31
Names of the documents retrieved for query 1: file864.txt, file264.txt, file797.txt, file573.txt, file459.txt, file892.txt, file863.txt, fil
```

Q3. Positional Index and Phrase Queries

Methodology:

Creating Positional Index:

- Iterate through each preprocessed document in the dataset obtained from Q1.
- Tokenize the text and create a mapping of each unique token (unigram) to the list of positions where it appears in the document.
- If a token already exists in the positional index, append the current position to its list of positions. If not, create a new entry for the token with the current position.

Supported Operations:

- Implement a function to execute phrase queries.
- Parse the input phrase query to identify the tokens and their positions.
- Retrieve the document IDs associated with each token from the positional index.
- Check if the positions of tokens in the query match the positions in the documents.
- Return the set of document IDs where the phrase query is found.

Input Preprocessing:

- Apply the same preprocessing steps as in Q1 to the input sequence before executing the queries.

- Preprocessing includes lowercasing, tokenization, stopword removal, punctuation removal, and blank space token removal.

Approach:

Creating Positional Index:

Similar to the method followed in Q2, except along with the documents/files a list of positions for each file is also saved which records the position of the token in that file.

Executing phrase queries:

- Now to process phrase queries, I similarly take input from the user, which is first preprocessed.
- For each token in the processed query, I check whether it exists in the positional index created in first part, if not query is exited and error message is printed.
- Else if it's the the first term ie result is not initialized the current list of docs for the current word is set to result
- Creating a set of documents and using intersection operation to find common documents that contain the tokens from the input
- Further we keep a variable count which notes the relative position of the terms from the input query in a sequential manner
- This count variable allows to extract phrases from the documents
- Now for each common document we evaluate the set intersection of current documents positions and the augmented positions with adding count
- From this intersection we get our result

Results:

```
['ring', 'finger']
Number of documents retrieved for query 1 using positional index: 2
Names of documents retrieved for query 1 using positional index: file994.txt, file70.txt
```