CSE508 Winter 2024 A1 Report

Rahul Ajith — 2021083

09-02-2024

Overview

This report details the development and outcomes of three interconnected systems designed to process and query a dataset of text files. The project is structured into three main tasks: preprocessing text files, creating an inverted index, and implementing a positional index for phrase queries. Each task employs specific methodologies, makes certain assumptions, and achieves distinct results, which are elaborated below.

0.1 Preprocessing Text Files

Approach and Methodology

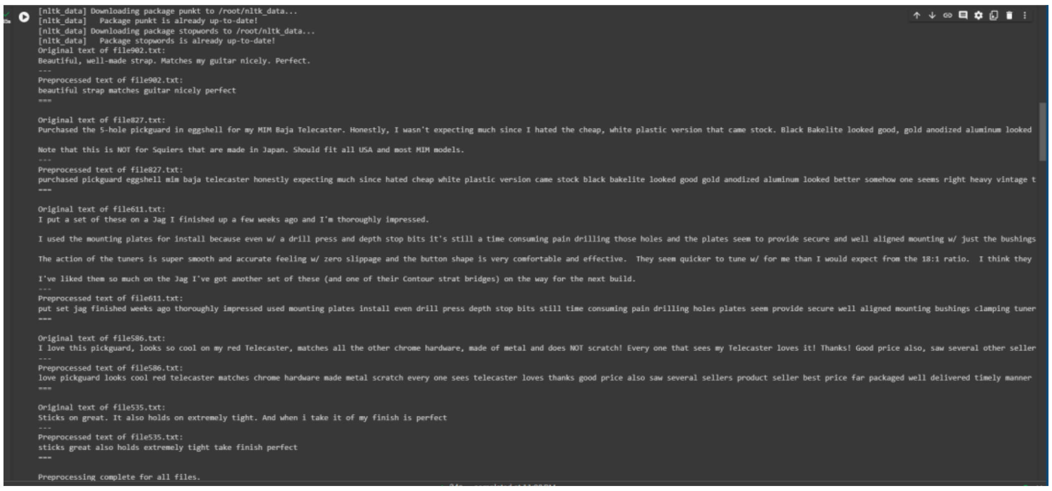
The preprocessing step involves reading text files from a dataset, converting the text to lowercase, tokenizing the text into words, removing stopwords and punctuation, and finally, saving the preprocessed text into new files. The process utilizes the NLTK library for tokenization and stopwords removal.

Assumptions

- Text content is primarily in English.
- Punctuation and stopwords (commonly used words that do not contribute significantly to the semantic meaning) can be safely removed without losing critical information.
- Lowercasing all text standardizes the dataset, making future processing and querying more consistent.

Results

Preprocessing was successfully applied to all files in the dataset. Each file's content was cleaned and stored in a new directory, ensuring that the original dataset remained unaltered. The system randomly selected five preprocessed files to display their content as a verification step, confirming the preprocessing effectiveness.



```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
Original text of file902.txt:
Beautiful, well-made strap. Matches my guitar nicely. Perfect.
---
Preprocessed text of file902.txt:
beautiful strap matches guitar nicely perfect
===

Original text of file827.txt:
Purchased the 5-hole pickguard in eggshell for my MIM Baja Telecaster. Honestly, I wasn't expecting much since I hated the cheap, white plastic version that came stock. Black Bakelite looked good, gold anodized aluminum looked

Note that this is NOT for Squiers that are made in Japan. Should fit all USA and most MIM models.
---
Preprocessed text of file827.txt:
purchased pickguard eggshell mim baja telecaster honestly expecting much since hated cheap white plastic version came stock black bakelite looked good gold anodized aluminum looked better somehow one seems right heavy vintage t
===

Original text of file611.txt:
I put a set of these on a Jag I finished up a few weeks ago and I'm thoroughly impressed.

I used the mounting plates for install because even w/ a drill press and depth stop bits it's still a time consuming pain drilling those holes and the plates seem to provide secure and well aligned mounting w/ just the bushings

The action of the tuners is super smooth and accurate feeling w/ zero slippage and the button shape is very comfortable and effective.  They seem quicker to tune w/ for me than I would expect from the 18:1 ratio.  I think they

I've liked them so much on the Jag I've got another set of these (and one of their Contour strat bridges) on the way for the next build.
---
Preprocessed text of file611.txt:
put set jag finished weeks ago thoroughly impressed used mounting plates install even drill press depth stop bits still time consuming pain drilling holes plates seem provide secure well aligned mounting bushings clamping tuner
===

Original text of file586.txt:
I love this pickguard, looks so cool on my red Telecaster, matches all the other chrome hardware, made of metal and does NOT scratch! Every one that sees my Telecaster loves it! Thanks! Good price also, saw several other seller
---
Preprocessed text of file586.txt:
love pickguard looks cool red telecaster matches chrome hardware made metal scratch every one sees telecaster loves thanks good price also saw several sellers product seller best price far packaged well delivered timely manner
===

Original text of file535.txt:
Sticks on great. It also holds on extremely tight. And when i take it of my finish is perfect
---
Preprocessed text of file535.txt:
sticks great also holds extremely tight take finish perfect
===

Preprocessing complete for all files.
```

0.2 Creating an Inverted Index

Approach and Methodology

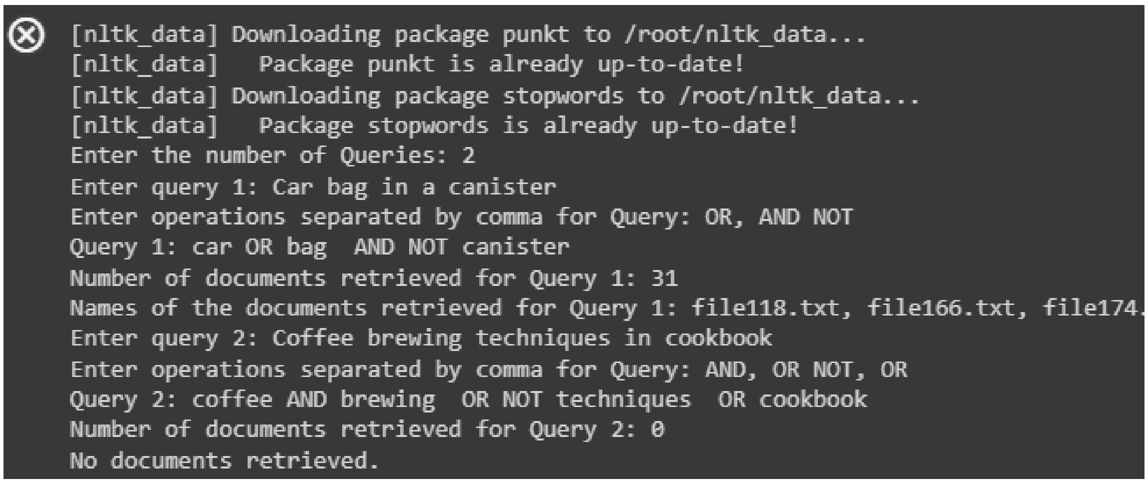
An inverted index was built to facilitate efficient document retrieval based on individual terms. The index maps each unique word to a set of document names containing that word. The process involves reading preprocessed files, extracting words, and updating the index accordingly.

Assumptions

- Each word's occurrence is significant for document retrieval, disregarding its specific position within the document.
- The dataset has been preprocessed, meaning it's in a consistent format without punctuation or stopwords, and all text is lowercase.

Results

The inverted index was successfully created and serialized using Python's pickle module for persistent storage. The system supports basic boolean query operations (”AND”, ”OR”, ”AND NOT”, ”OR NOT”), allowing for flexible document retrieval based on user input. The query system demonstrated the ability to accurately return documents that match the specified boolean criteria.



```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
Enter the number of Queries: 2
Enter query 1: Car bag in a canister
Enter operations separated by comma for Query: OR, AND NOT
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.
Enter query 2: Coffee brewing techniques in cookbook
Enter operations separated by comma for Query: AND, OR NOT, OR
Query 2: coffee AND brewing  OR NOT techniques  OR cookbook
Number of documents retrieved for Query 2: 0
No documents retrieved.
```

0.3 Implementing a Positional Index for Phrase Queries

Approach and Methodology

The positional index extends the inverted index by additionally storing the position(s) of each word within documents. This structure enables the processing of phrase queries, where the order of words matters. The system preprocesses phrase queries, searches for documents where the phrase occurs in the specified order, and reports the findings.

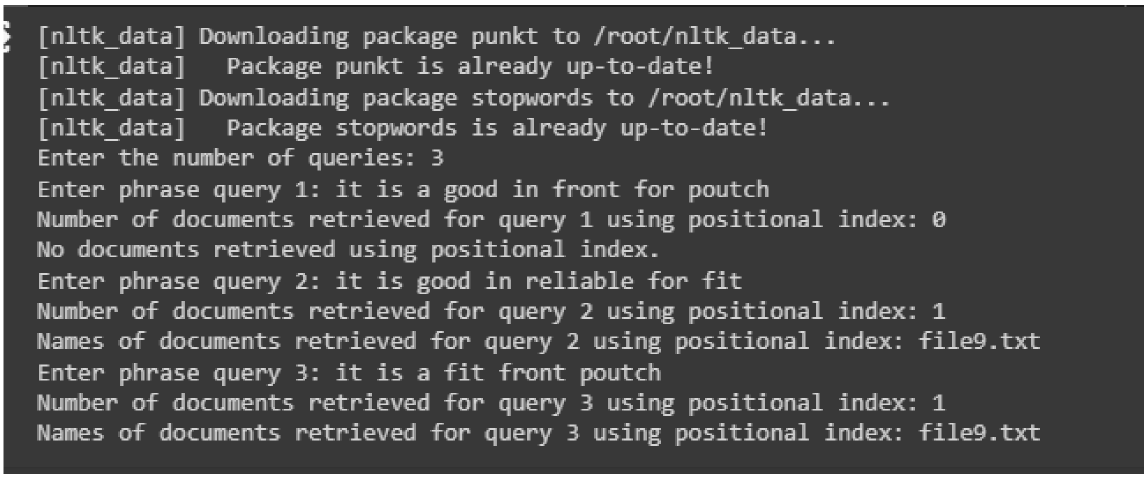
Assumptions

- The relevance of a document to a phrase query depends on the occurrence of all query terms in the given order within the document.

- Preprocessed documents are used, ensuring a uniform analysis base.
- Phrase queries are limited to a reasonable length (5 words) for performance considerations.

Results

The positional index system successfully processes phrase queries, identifying documents where the specified phrases occur exactly. The system's effectiveness was validated by running user-inputted queries and accurately reporting the number of matching documents and their names.



```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
Enter the number of queries: 3
Enter phrase query 1: it is a good in front for poutch
Number of documents retrieved for query 1 using positional index: 0
No documents retrieved using positional index.
Enter phrase query 2: it is good in reliable for fit
Number of documents retrieved for query 2 using positional index: 1
Names of documents retrieved for query 2 using positional index: file9.txt
Enter phrase query 3: it is a fit front poutch
Number of documents retrieved for query 3 using positional index: 1
Names of documents retrieved for query 3 using positional index: file9.txt
```

Conclusion

The project successfully implemented a comprehensive text processing and query system across three major tasks. Each component demonstrated the application of natural language processing techniques and data structures for efficient information retrieval. Through careful preprocessing, strategic index building, and the implementation of robust query functionalities, the system provides a solid foundation for more advanced text analysis and search applications in larger datasets.