CS410 Text Information Systems - Fall 2022

Final Project Report

Team SALE

- Eunbi Go - eunbigo2 (Team leader)
- Akarsh Bhagavath - akarshb2
- Lingfei Yang - lyang26
- Sruthi Jayanti - sjayan3

Motivation

We have chosen the topic of intelligent browsing for this project. We plan to build a browser extension that takes in book title keywords as user input and outputs a sentiment score for reviews about that book after performing sentiment analysis on a book reviews dataset. The extension will also serve as a recommendation system by recommending books similar to the book title entered based on sentiment scores.

This project solves the problem of having to read through multiple book reviews to gauge how good a book is. This can be a tedious experience, and our proposed project aims to solve this problem intelligently. Similarly, a recommendation system for books pushes useful information to the user, which minimizes the need for users to go find similar books on their own.

This topic and project relate to the theme of the text information systems because the Chrome extension we propose involves sentiment analysis on a textual dataset, as well as pushes information to the user based on a recommendation system model. These are topics directly relevant to the course.

Implementation

Programming language(s)

- Backend: Python
- Chrome extension: HTML, CSS, JavaScript

Repository: https://github.com/eunbigo91/CS410_teamSALE

Extension/ ├── jquery/ ├── back-end/ │ ├── data/	jQuery library used for front-end Back-end related files

│ │ │── kindle_review.csv │ │── output/ │ │ │── Final_Asin_To_Review_Mappings~.csv │ │ │── Final_Mapping_with_recommendations.csv │ │ │── final_mapping.json │ │ │── final_mapping_recommendation.json │ │ │── sentiment_output.csv │ │ │── sentiment_output.json │ │ │── sentiment_output_original.csv │ │── FinalProjectRecommendations.ipynb │ │── dataprep.py │ │── requirements.txt │── front-end/ │ │── book.png │ │── index.css │ │── script.js │── index.html │── manifest.json	CSV and json files that contains the analysis result Front-end related files Extension icon Cosmetic features Search function and result display Front-end index html Chrome extension manifest

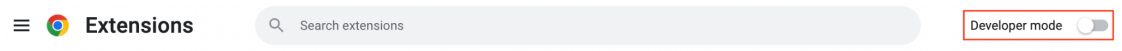
Installing the extension

Below are steps to install the Chrome extension and perform the execution. This project code has not been deployed, so Chrome extension developer mode should be enabled to execute the code.

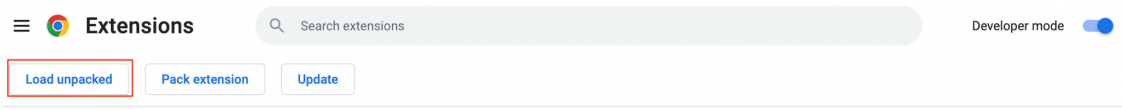
1. Download or clone the project repository. If you are using the command line, the following command will accomplish this:

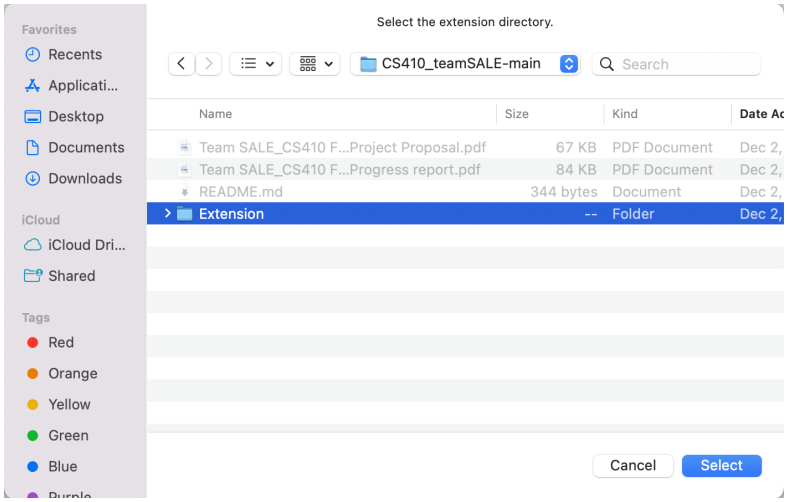
   ```
   git clone https://github.com/eunbigo91/CS410_teamSALE
   ```

2. Go to chrome://extensions/ and enable Chrome extension developer mode on the upper right corner.



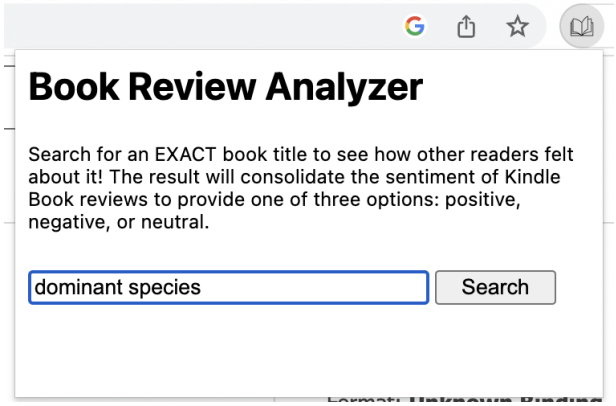
3. Click ‘Load unpacked’ and select the ‘Extension’ folder on the file explorer.

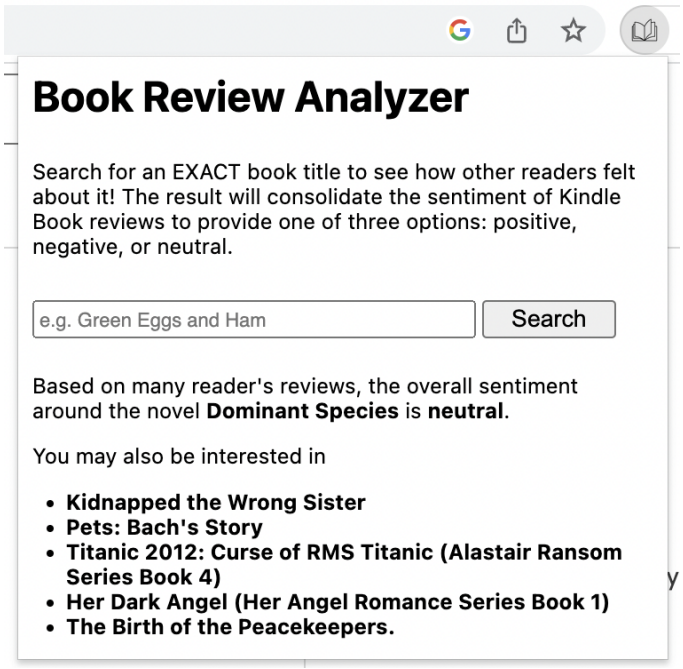




4. Once you have loaded the extension, you should be able to view it on the chrome://extensions page. Click on the puzzle icon on the top right corner of the page in order to see all extensions that have been loaded. Then, pin the extension called “Book Review Analyzer”. Now, you should be able to click on it easily in order to load the extension.

5. Enter in the title of a book (the current functionality is case-insensitive but only accommodates exact book titles). Once you search for a book title, you should be able to view the results of the sentiment of reviews about that book. You should also be able to view five titles of books that have a high similarity with the user-inputted book. An example is shown below.





Backend: Sentiment Analysis

The raw data used is saved under Extension/back-end/data/kindle_review.csv that contains 12,000 rows of book reviews from Amazon Kindle as described in the previous progress report. Each row represents one review for each book with a score given by the reviewer with review summary (title). The purpose of sentiment analysis is to provide an emotion (positive, neutral, or negative) for each row given the book review text and review summary text.

Methodology and Process Description:

First, we performed the data preparation where some of the steps were mentioned in the progress report already. For example, we detected non-English reviews, cleaned data by removing stopwords and tokenized the words. We also check short reviews and ensure that they are meaningful and should be included. After the progress report, we also checked if the data is balanced and observed that the data by rating scores (1-5) are quite evenly distributed. Hence, there is no need to perform imbalance treatment, and the results are considered robust. In addition, we confirmed that the data does not contain blank reviews. As a result, we prepared a clean version of the review text after the tokenization of stopwords removal.

Then, the model we used for text sentiment analysis is VEDAR[1] (Valence Aware Dictionary for Sentiment Reasoning) that is sensitive to both polarity (positive / negative) and intensity (strength) of emotion. It is available in the NLTK package and can be applied directly to unlabeled text data.

We noticed that one of the features of the VEDAR model is that it can even understand the emphasis of capitalization and punctuation, such as “ENJOY” in a more positive way. Thus, we tested both the data after tokenization without differentiation of uppercase or lowercase, and the raw review text data as VEDAR model may work better in the original review text without tokenization. Besides, we observed that the review title is short but conveys straightforward emotions. For instance, “I love it”, “Five stars”, etc. Hence, VEDAR model is also applied to the review title summary.

After analyzing the VEDAR scores for (1) raw review text, (2) clean version of review text and (3) summary text, we observed the following:

- The accuracy of summary text is the highest among all three types of texts. The possible reason is that summary text usually barely contains vague expressions and the tone and expressions are usually straightforward.
- The accuracy of raw review text is higher than the accuracy of the clean version of the review texts. To some extent, this proved our initial guess that the VEDAR model suits the raw text which can keep capitalization, punctuation, etc, which are part of VEDAR score calculation and can improve the accuracy.

Given the above observations, we set up the following cut-off rule of VEDAR scores to classify the sentiment to (1) neutral, (2) positive, and (3) negative:

- **neutral**: summary VEDAR score == 0 and reviewText VEDAR score [0, 0.8963]
- **positive**: summary VEDAR score > 0 or (summary VEDAR score == 0 and reviewText VEDAR score > 0.8963)
- **negative**: summary VEDAR score < 0 or (summary VEDAR score == 0 and reviewText VEDAR score < 0)

Summary VEDAR score == 0 can indicate neutral sentiment in most cases. VEDAR score on raw review text can assist to distinguish those false indications from summary VEDAR score == 0 cases. Here, threshold 0.8963 is from the mean of Vedar score statistics for raw review text without tokenization.

As a result, this setting can lead to the overall accuracy to reach around 73%.

How to implement it:

[1] Reference: https://towardsdatascience.com/sentimental-analysis-using-vader-a3415fef7664

The data preparation (mentioned in the previous progress report) and sentiment analysis codes are in python, which are saved under Extension/back-end/dataprep.py. Python 3.8 and version above works for the code. The user can run the following codes in the terminal to implement it. The requirements.txt includes modules / packages required for the run with version predefined.

pip install -r requirements.txt
python3 dataprep.py

The packages in the requirements.txt are below:

```
nltk==3.6.5
numpy==1.21.2
pandas==1.3.4
textblob==0.15.3
langdetect==1.0.9
```

The followings are the list of the functions, and the description of the functions for sentiment analysis, which includes initial data preparation:

- Function **remove_stopwords**: It is used for stopwords removal. The package NLTK iis used for word tokenization. Packages of nltk required for tokenization and stopwords are downloaded in advance.
- Function **detect_language**: It is used for english review detection. We use langdetect as a package to detect it. From observation, lower cases for the words have higher precision. Hence, we lowered the cases first, and also printed out those which are identified as non-english data for double confirmation.
- Function **categorize**: Apply the cut-off thresholds for Vedar scores based on review text and summary text. The function will generate the final classification of positive, negative, or neutral as labels in the output file.
- Function **main**: the flow of the whole process from end to end.
    - Data preparation:
        - Read raw data, and call function detect_language
        - Call remove_stopwords
        - Check short reviews with words less than 3 to make decision if short reviews are meaningful to be included in the scope
        - Check if data is balanced to ensure the robustness
        - Check if NA values are in review / summary texts
        - Generate a clean version of the review text after tokenization with lower cases
    - Sentiment Analysis

- Apply Vedar Models on (1) clean version generated from data preparation (2) original review text (3) review summary (title) text to generate Vedar scores for each of these three types of texts.
- Perform data analysis to determine the cut-off of sentiment scores
- Determine the cut-off and call function categorize to generate the final output of label (pos = positive, neg = negative, neu = neutral)

○ Save output files

The output file is sentiment_output_original.csv saved under the back-end/output folder. The file will be connected to the front-end.

The output produced by success script running is similar to the following:

```
################################################################
# Project: Book review sentiment analysis and recommendation     #
# Team: SALE (Eunbi, Sruthi, Lingfei, Akarsh)                    #
# Authors: Lingfei Yang                                          #
# Part I: data preparation and Sentiment analysis                #
################################################################


Attempting to download required package files...
Successfully downloaded required package files.
Reading raw data...
Raw data is read successfully.

Step 1: Data preparation

(1) detect non-English item... eye manual double confirmation. The following shows the rows package identified as non-English...

1542 :  did not finish it. deleted from kindle.word word word word word word word word word word word word
2411 :  very hot
2423 :  love it
2928 :  enjoyable
6159 :  i was disappointed i did not like it.  i did not like it.  again i did not like it. period.
8345 :  estoy conforme con &eacute;l, sin embargo es un poco pesado, casi tanto como el kindle dx. ser&iacute;a mejor que diera la misma protecci&oacute;n, pero con un peso menor.
8894 :  very great book

language detection is completed. After double confiramtion, only row 8345 is Spanish. All others are valid English reviews.

(2) tokenize and remove stop words ...
```

…

The end of the output of script running looks like the following:

```
Apply the cut-off based on the following rules:

 neutral: summary score == 0 and reviewText score [0, 0.8963];
 positive: summary score > 0 or (summary score == 0 and reviewText score > 0.8963;
 negative: summary score < 0 or (summary score == 0 and reviewText score < 0

Label of sentiment has been created successfully, output saving ...

Output sentiment_output_original.csv has been saved.
```

Further improvement for sentiment analysis:

Since the project's main purpose is to implement the sentiment analysis and make it useful and meaningful through extensions, we did not split the raw data into in-sample and out-of-sample to perform the testing to ensure the result is robust. However, since the data we used is balanced data, the performance is robust in general.

Besides, the pretrained data is on raw data to obtain the threshold setting for Vedar model during sentiment analysis. Ideally, the pretrained data should be another set of review data. Our current way may be overfitting the outputs. However, from the purpose of the project usage, we keep it simple in the project.

Finally, due to time constraints, we did not test different models for sentiment analysis performance. However, since the Vedar model is widely used and considered as a decent model for sentiment analysis, it is sufficient for the project purpose. We may explore other alternatives in the future.

Backend: Recommender System

In addition to sentiment analysis, the project's initial goal was also to create a recommender system. The way we did this was to run TF-IDF weighting on the book's product description, and score the top 5 books with the most similar descriptions based on cosine similarity. This was a rather simplistic approach, but effective. We followed this guide to implement the TF-IDF vectorizer: https://towardsdatascience.com/how-to-rank-text-content-by-semantic-similarity-4d2419a84c32

And ran the algorithm on the 12000 entries, retrieving the top 5 similar books for each entry.

Further improvement for the recommender system:

We need to exploit the semantic similarity in the product description to get a better sense of similar product reviews in future iterations.

Team Effort

- Sruthi Jayanti and Eunbi Go focused on building the frontend for this project.
    - Sruthi Jayanti worked on some of the cosmetic features of the extension interface. She also worked on writing the code for parsing the sentiment analysis output based on user inputted keywords in order to display the sentiment label on the extension interface.
    - Eunbi Go worked on functionality of the extension interface and some parts of cosmetic features as well. She further implemented the display of the 5 recommended books on the extension by applying the code created by Sruthi.
- Lingfei Yang and Akarsh Bhagavath focused on the backend work.
    - Lingfei worked on backend work of progress report, which focused on the initial data preparation. In the later stage, Lingfei performed the sentiment analysis, explained it to

the team and discussed with the whole team the connection between backend and frontend.

- Akarsh worked on mapping the data from asin to product title and product description using the ASIN Data API. Akarsh also used TF-IDF to create book recommendations based on the product description.