Project Report

CS2001 Data Structures

Semester Project

Name

Syed Ahmed Mahmood 20K-0153

Muhammad Qasim Fuzail 20K-0157

Section D

Submitted to:

Ms Mubashra Fayyaz

Department of Computer Science BS(CS)

FAST-NUCES Karachi

Rule Based Sentiment Analysis

Description

In this project, we used the Amazon review dataset for sentiment analysis to determine for each product in the Kindle Store category, the number of total reviews, the number of positive reviews and the number of negative reviews. We also found the top 10 most used positive and top 10 most used negative words in the reviews of each product and assigned each product a star rating(out of 5) on the basis of positive reviews and total reviews.

Methodology

(Merger.cpp):

Merging Two JSON Files and Cleaning Data :

We had two JSON files of the Kindle category Amazon review data set, the *metadata("meta_Kindle_Store.json")* and the *review data("Kindle_Store_5.json")*, the field that was common in the two files was the unique product ID called "*asin*". From the *metadata* file we required the "*title*" field and from *review data* we required the "*reviewText*" field. Therefore we created a program called "*merger.cpp*" to merge these two JSON files into one as well as clean the data. We dropped any records that had any missing values from the fields that we required and also dropped useless fields such as reviewTime, vote etc. The program took the two json files, then read the files and made JSON objects using nlohmann's JSON for Modern C++ API. First it read the *metadata* file and saved it in a STL map with *asin*(unique id) as key and *title* as it's value. It then proceeds to read the *review file* and make JSON objects from it and matches *asin* from the map we previously made. Once *asin* matches it creates a smaller JSON object with only the fields that we require (*asin, reviewText, title*) and appends it to an object from nlohmann that represents a JSON array after which it is stored to the output file clean_Kindle_Data.json. Rest of the work is done in Source.cpp.

(Source.cpp):

Storing Lexicons into Data Structures:

We read and stored three text files that contained lists of words. Two of these files contained a list of positive and negative words that was stored into a STL Map each(the word was the key and the value was the count of the word for later use that was initially 0). The third file contained a list of stopwords that were read and stored into a STL Set.

Parsing the JSON File:

Our first biggest challenge was to parse the JSON files that C++ has no native support for. At first we tried using a library made using C++ (nlohmann's JSON for Modern C++ API) but we found out it took nearly an hour for it to just read the 1.63GB JSON file and make objects from it and it also took up a large amount of RAM for doing that which was more than double the size of the file.

So we spent the majority of our time trying to figure out how to read this file efficiently. We wrote some code to read the file in such a way that it only read one JSON object at a time, which was particularly difficult since we could not divide the objects by the newline character since the reviews in the file also used that character in multiple places and we could not divide them by a comma or a closing curly bracket since they were also present in the review. So after a lot of testing we devised a method to check if a JSON object had been completely read that would not cause exceptions or errors. Upon debugging it was observed that our method of parsing the JSON file was more efficient than directly parsing it using the nlohmann library since our method took only 6 minutes and 1.7 GB to read and parse this massive file into a STL Map.

Creating the STL Map:

We stored each object one-at-a-time as a string and passed it to a function(*fillmap*) along with the reference to a STL Map(that stored the product title as the key and a STL List of reviews(string) of that product as the value). What the *fillmap* function does is that it uses a single function from the previously mentioned nlohmann library to parse the string we passed to it and create a JSON object in C++ with it, we then took the reviews from this object and stored it into our Map. A product title has multiple reviews so this function used a static variable to keep track of the title whose reviews are currently being filled into the Map. Only those titles and reviews were used that were non-empty strings.

Analysis of the Reviews:

After our Map was completely made, it was time to tokenise each review and extract useful words from it. We made a function called *analyseReviews*, into which we passed each key value pair from the Map one at a time, along with this we also passed the the Map of positive words and the Map of negative words along with the Set of stopwords. This function iterated through the STL List of reviews(the value from the key value pair), and passed each string to another function called *tokenise* that also took the 2 Maps and the Set stated previously as arguments.

This function kept track of the polarity of the review by keeping a sum of the polarity of the words. First of all it scanned the review string for any symbols and replaced them with spaces, then it converted the string into a sstream object and took each space separated word one by one and compared it with the Set and 2 Maps. If the word was found in the stopwords it was ignored, if it was found in the positive words then a +1 was added to the polarity sum called *score* and the count(the value in the Map) of the word(the key in the Map) was incremented to indicate that this word was present in the review, if it was found in the negative words then a -1 was added to the polarity sum and the count of the word in the Map was incremented, else the word was ignored. Then *tokenise* returned the polarity of the review to the calling function *analyseReviews*.

This function judged if the review was positive or negative and incremented the appropriate counter variable, if the polarity of the review was 0 it would be considered neutral. 2 STL Multimaps were also created from the positive and negative words Maps, these Multimaps were iterated to get the first 10 most occurring positive and 10 most occurring negative words in the review.

Then the star rating is calculated by the formula:

$$StarRating(Roundedto2decimalplaces) = \frac{No.ofPositiveReviews}{TotalReviews} \times 5$$

Making The Output File

To output and present our finding we created a .csv file that has 95716 entries(The products) and 7 fields.

The 7 fields(table header is the first row in the .csv file) are:
Title, **Number of Total Reviews**, **Number of Positive Reviews**, **Number of Negative Reviews**, **Popular Positive Words(Atmost 10 words), Popular Negative Words(Atmost 10 words), Star Rating(Out of 5)**
Note: Due to neutral reviews being ignored column3 and column4 will not always add up to column2

Here is an excerpt from "Sentimental Analysis Output Data.csv" which would be available in the project folder:

	A	B	C	D	E	F	G
86266	The Sentinel (Wolfe Series Bool	11	10	0	['great','hot','love','good','e	['issues','frustrating','loos	4.5
86267	The Sentinel - Kindle edition	55	43	7	['love','great','good','enjoye	['plot','fiction','conflict','c	3.9
86268	The Separation - Kindle edition	11	4	6	['love','captivating','recomr	['loss','emergency','lies','p	1.8
86269	The September Letters (Kindle	11	7	1	['excellent','interesting','lov	['drastically','complaint','	3.1
86270	The Sequel (Bibliomysteries Bo	5	3	1	['cool','gold','greatest','pop	['mysterious','mystery','st	3
86271	The Serafina Sin City Series Coll	30	29	1	['love','angel','great','good',	['surrender','sin','danger',	4.8
86272	The Serano Brothers - Kindle ec	37	29	7	['love','great','good','enjoye	['killer','bad','death','disa	3.9
86273	The Seren Trilogy - Kindle editic	6	6	0	['love','magical','wonderful	['errors','strange','bland','	5
86274	The Sergeant (Cuffs, Collars, an	105	77	22	['love','loved','good','great',	['submissive','issues','plot	3.6

Results and Findings

Our project read 2,225,967 reviews of 491,670 products from the Kindle Books category in the Amazon store and cleaned this data to get 2,207,798 reviews of 95,716 products. It then analysed the number of positive and negative reviews of each product to predict a star rating for each product, using the formula stated in the previous section. Along with this it also analysed the top 10 most popular positive and negative words used in all the reviews of each product.

We created a file from our results of predicting the star ratings of each product called "Title-Total Reviews-Stars .xlsx", which will be available in the project folder, in which we used star icons to represent the star ratings for easy viewing.
Here is an excerpt from that file:

Title	Number of Total Reviews	Star Rating(Out of 5)	Stars
Grey: Fifty Shades of Grey as told	2217	3.3	★★★☆☆
Claimed	2014	3.7	★★★½☆
Death	1468	2.3	★★☆☆☆
The Atlantis Gene: A Thriller (The Origi	1461	2.8	★★☆☆☆
Fueled (The Driven Series Book 2) - Kin	1327	2.9	★★☆☆☆
Sense And Sensibility (Annotated	1249	3.8	★★★½☆
Taken	1199	3.6	★★★½☆
Bound	843	3.5	★★★½☆
The Girl in the Box Series, Books 1-3: A	808	3.3	★★★☆☆
Tempted	784	3.6	★★★½☆
Tears of Tess (Monsters in the Dark Bo	772	1.8	★½☆☆☆
Takedown Twenty: A laugh-out-loud c	762	2.9	★★☆☆☆

In the image above you can see that the graph is sorted by the field “Number of Total Reviews” (in descending order). We found that the greater the number of reviews of a product there are the better we can predict it’s star rating.

You can observe in the image of the table given below, that if it is sorted by the field “Star Rating”(in descending order) and then the field “Number of Total Reviews”(in descending order) then these records give us an idea about the highest rated products, which is more accurate as compared to just sorting the table by “Star Rating”(in descending order). For example there was a record with a 5 star rating but it had only 1 review which was positive so this entry does not tell us much about the sentiment regarding the product.

Title	Number of Total Reviews	Star Rating(Out of 5)	Stars
Super Cool Wildlife	76	5	★★★★★
ebook,Baby Professor,Sea Turtles: Fun	66	5	★★★★★
ebook,Baby Professor,Reptiles of the V	64	5	★★★★★
ebook,Baby Professor,Elephants of the	60	5	★★★★★
The Marriage Match (Suddenly Smitte	59	5	★★★★★
Beauty Recipes for Anti Aging (Boxed S	55	5	★★★★★
Why Does It Happen?	54	5	★★★★★
ebook,Baby Professor,Coral Reefs and	49	5	★★★★★
Around The Globe - Must See Places ir	46	5	★★★★★

Similarly in the image of the table given below, it is sorted by the field “Star Rating”(in ascending order) and then the field “Number of Total Reviews”(in descending order) and these records give us an idea about the lowest rated products.

Title	Number of Total Reviews	Star Rating(Out o	Stars
Para Elisa (Libro 1) Biloga (Spanish Edit	25	0	☆☆☆☆☆
Nada prohibido (Spanish Edition) - Kin	23	0	☆☆☆☆☆
El acontecimiento (Spanish Edition) - K	23	0.2	☆☆☆☆☆
CONFLICTO	22	0.2	☆☆☆☆☆
Dark Hearts eBook	22	0.2	☆☆☆☆☆
Serendipia: El amor puede ser un halla	22	0.2	☆☆☆☆☆
Heridas del pasado (Spanish Edition) e	20	0.2	☆☆☆☆☆
Irene 2 (Spanish Edition) eBook	20	0.2	☆☆☆☆☆
Luna Azul (Spanish Edition) eBook	20	0.2	☆☆☆☆☆
Doble o Nada (Spanish Edition) - Kindl	20	0.2	☆☆☆☆☆
Soy Tuya: Ella mirar en su corazn y solo	20	0.2	☆☆☆☆☆
Perdoname (ngel Prohibido n 6) (Span	32	0.3	☆☆☆☆☆
Regresar a ti (HQ) (Spanish Edition) - Ki	28	0.3	☆☆☆☆☆
El Enviado (La Flor de Jade nº 1	27	0.3	☆☆☆☆☆
Scars: Book One - Kindle edition	27	0.3	☆☆☆☆☆
Vendetta (Spanish Edition) - Kindle ed	26	0.3	☆☆☆☆☆

As stated previously our project also found the 10 most popular positive and negative words for each title and in the following wordclouds we have shown a small subset from the overall most popular positive words and negative words.

Top 279 overall most popular positive words from the reviews:

Top 300 overall most popular negative words from the reviews:

Tools and Dataset used

We used the 5-core subset(1.63GB) and metadata subset(474MB) for the category "Kindle Store" from Amazon Review Data (2018).
Along with this we also used a lexicon of positive words and a lexicon of negative words from http://www.cs.uic.edu/~liub/FBS/opinion-lexicon-English.rar.
We used nlohmann's JSON for Modern C++ API to convert JSON objects in string form to proper JSON objects that are useable in C++.
We used several data structure classes such as Maps and Sets and Lists from the C++ STL
We used Visual Studio 2022 as our IDE to develop this project and Github to collaborate with our team.

Conclusion

In summary, we used rule based sentiment analysis implemented in C++ on a large dataset of Amazon reviews to calculate the sentiments about products in the Kindle category of Amazon.com and by calculating the polarity of reviews and the number of reviews of each product, we predicted a star rating of each product. Along with this we also found the most used positive and negative words in all the reviews of each product and displayed the top 250 to 300 positive and negative words using a word map. Companies can use these results to predict whether their Products would make a profit in the Kindle category of Amazon.com and what reviews and ratings can be expected for their product by looking at similar products.

Citations and References:

C++ API:

nlohmann/json: JSON for Modern C++

JSON for Modern C++ API by Niels Lohmann

Datasets:

Amazon review data (nijianmo.github.io)
Please cite the following paper if you use the data in any way:
Justifying recommendations using distantly-labeled reviews and fined-grained aspects
Jianmo Ni, Jiacheng Li, Julian McAuley
Empirical Methods in Natural Language Processing (EMNLP), 2019 pdf

Opinion Lexicon

Available as 'A list of positive and negative opinion words or
sentiment words for English' from
http://www.cs.uic.edu/~liub/FBS/sentiment-analysis.html#datasets

Source:
http://www.cs.uic.edu/~liub/FBS/opinion-lexicon-English.rar

Attribution:
"Opinion Lexicon by Bing Liu [http://www.cs.uic.edu/~liub] is licensed under
CC BY 4.0 International [http://creativecommons.org/licenses/by/4.0/]"