Unveiling User Sentiments: A Comparative Analysis of Facebook App Review Sentiments from Play Store and App Store & Identify Commonly Discussed Features

Mehrab Mustafy Rahman
UIN: 661630274

1 Introduction

We use our phones every day. To stay connected we use social media through our phones. Alongside connectivity, social media has become a hub for getting important news, updates and so much more. Since social media has become a part and parcel of our everyday lives, it is important to get the best possible experience from social media. Thus understanding the sentiments associated with using these apps is crucial. Sentiment analysis is a valuable tool for businesses and organizations to gain insights into customer opinions, market trends, and public perception. It is widely used in various industries, including marketing, customer service, brand management, and social media monitoring, to make informed decisions and respond effectively to public sentiment. Machine learning algorithms, deep learning models, and rule-based systems are often employed to automate the sentiment analysis process and enhance its accuracy.

When an app becomes popular and has millions of downloads and users, sometimes it becomes difficult for the developers to track the progress, the bugs, and the inefficiencies of that app. That is because reviews from so many people are not feasible to be read by the developers of the app. I feel that there is a disconnection between the customers and the development team. As a result, many of the customer demands and requirements are not getting reflected or fixed on the newer versions/ releases of the app itself. For example, there are more than 78,000 reviews of the Facebook app in the Apple app store and there are more than 137 million ratings in the Google Play store. One more thing to consider is whether can we trust ratings. For example, there are reviews like, "This is such a bad app", but the rating is 5. It could be that the user was either not paying attention while giving a rating or thought 5 to be bad and 1 to be good. Thus, trusting ratings to understand the sentiment is not a very good approach. We need to go through the reviews left by the users to get a better grasp of the sentiment. Since this process is very lengthy (given the huge number of reviews), we need to automate the process and the way to do it by using some Machine Learning model to perform the sentiment analysis with a reasonably high accuracy. Hence the first part of this project deals with **Sentiment Analysis of Reviews.**

Alongside the sentiments of the reviews, it is important to find the features that are most talked about by the users, whether those features need to be updated, or are the users satisfied with those features. If this can be ensured the users will no doubt have a very pleasant experience

	Number of Reviews	Starting Year	Ending Year	Positive Reviews	Negative Reviews	Neutral Reviews
Playstore	10000	2023	2023	778	8506	716
AppStore	6020	2017	2023	350	5471	199

Table 1: Data Insight

while using that app. To deal with this, we need to **find the Keywords/Topics of the Reviews of the users.** In this study, I analyze the Facebook App reviews from 2 popular stores - Playstore (for Androids) and Appstore (for iPhones). I find the features that are talked about in the Android version of the app, in the iOS version, and then the common ones for both versions. Although this study has been done for the Facebook app, it can easily be done with any similar app, i.e. the pipeline and workflow remain the same.

The finding suggests that these are the common features of the app that both types of users are talking about: *"photos, video, marketplace, notification, reels, friend, groups, posts, comments, pages, ad"*. Unfortunately, all of the talked-about features are negative, which is consistent with the ratings of the app(further details will be provided in the Result discussion and Appendix section of this report[1]). Sentiment Analysis has also been performed for the review data. A high accuracy of **95.43% was achieved by an optimized RoBERTa-Large**. The findings are described in the Result Analysis and Appendix sections.

2 Methods/Case Study

2.1 Previous Study

A study similar to the one in this report has been performed before. [2] has performed topic analysis and sentiment scoring on 7 apps from the Play Store and App Store. For finding the keywords they used Latent Dirichlet Allocation (LDA) to find out the ground-level keywords and use those keywords to get high-level features. Similar techniques have been applied in this study as well. To generate the sentiment score they have used SentiStrength [4]. They report a precision of 0.59 and a recall of 0.51. For this study, we report accuracy and loss values.

2.2 Data Collection

The data was collected by scraping the playstore and appstore for the facebook app, hence the data is new and fresh. The appstore reviews range from the year 2017 to 2023, and the playstore reviews are all from the year 2023. There are 6020 reviews from the appstore and there are 10000 reviews from the Play Store. A quick look at the data statistics from the Table 1 can help. Initially, no positive, negative or neutral labels were present.

2.3 Data Preprocessing

Since these are review data, people are not writing sentences properly and to express sentiments they may use local slang or emojis. These are hard for the model to interpret hence we need

[1] All of the code, data, findings, graphs for this study has been made public and can be found at this GitHub link. Any feedback will be appreciated.

proper processing of the data to feed it to the model.

2.3.1 Dropping Columns & Cleaning

The scraped data contained many different fields that were unnecessary for our target. Hence we need to clean and drop those columns, the appstore data contained 7 columns including ID, Time, Rating, etc. I dropped all columns except the Review. Similarly, for the Playstore data, there were 10 columns I dropped all columns except the Review Column. Then I do some text cleaning by removing the stopwords, punctuations, and emojis. Since the work has 2 directions **1.** Sentiment Analysis and **2.** Topic Identification. We split our preprocessing in two directions.

After that, I performed some POS tagging. For the sentiment classification, we need nouns, verbs, adjectives, and adverbs. But for the Topic Identification, we will need only the nouns because the features of the app are only noun words. Again for the sentiment analysis, we need to perform lemmatization but it is not necessary for the topic modeling. So at this point, I have 2 different files with only Reviews.

2.3.2 Generating Sentiments

Since the data is unlabeled, there are no sentiment labels present in the text. It is a tedious as well as expensive process to label the reviews. One would need lots of time as well as human annotators to do the task. An alternate approach was taken in this study. Since there are strong pre-trained models available to us, I have used a pre-trained Bert model to generate the sentiments for the reviews. It can be easily done by using the "transformers" library and calling the pre-trained Bert model. For Sentiment Classification, the playstore and appstore data were combined together, but for the topic modeling, they were kept separate.

As per the findings of [3], it was evident that the neutral class is twofold and is not easily interpretable by the models. Hence it is better to not have a neutral class in our data. It also makes sense, because the users will either like the features and provide a positive or dislike the feature and leave a bad review. Hence, the reviews that were from the neutral class were dropped. Leaving with only positive and negative reviews. The issue here is that we have an imbalanced dataset, where there are around 13000 negative samples but only around 1000 positive samples. We discuss how to deal with this in our methodology. At this point for sentiment classification, we are dealing with a binary classification problem.

2.4 Methodology

A very simplistic approach to our methodology can be viewed in this figure:

2.4.1 For Sentiment Classification:

I have formulated the task to make it a binary classification. So the target now is to perform this binary classification, I report the loss and the accuracy for the classification models, and the goal is to maximize the accuracy and minimize the loss. For this project, I am experimenting with using Support Vector Machines (SVM), Logistic Regression, and RoBERTa. Before passing the review data, they were converted to vectors using TF-IDF vectorizer. The experiments for

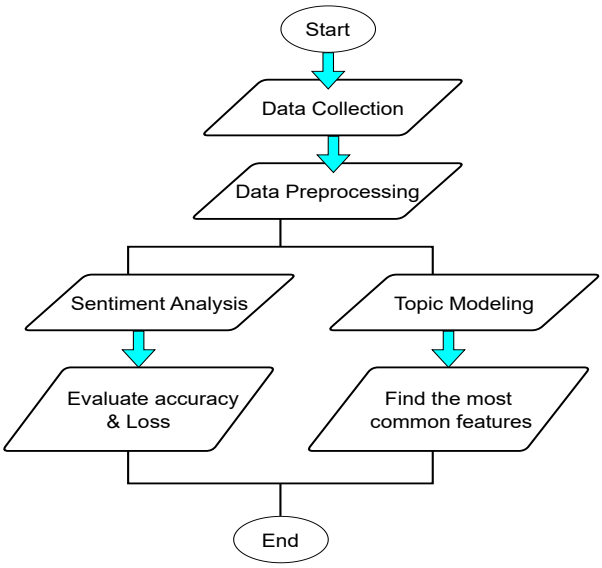

Figure 1: Workflow

accuracy were performed for the imbalanced dataset and an improved sampled dataset (more on this will be discussed in the Novelty section). Again the experiments for the loss calculation were performed on the imbalanced dataset. Experiments were performed for different preset loss functions and then for a customized loss function (more on this will be discussed in the Novelty section). After that, I performed GridSearch on different Hyperparameters(more on this will be discussed in the Results and Discussion section) to find the best-performing model. The total data was partitioned into train, test, and validation sets, and their curves are also reported in the Results and Discussion Section.

2.4.2 For Keyword Identification (Topic Modeling):

Following the strategy of [2], the key features or topics of the reviews were identified using Latent Dirichlet Allocation (LDA). The only difference between this study and the aforementioned research is that they have used the keywords to find higher-order features/topics from the keywords but I have just captured the topics with their relevance score for that topic and stored it in a dictionary as the keyword to be the key and the relevance score to value pair. I then found out the sentiment of that keyword from the dataset by summing the sentiment instances of that key over the whole dataset and then appended the score with the value with the relevance value of that particular key. Thus one entry would look something like this: *{'notification': [0.071, 0]} i.e. {'key': ['relevance score', 'sentiment of that key']}*. This was done for both the playstore data and the appstore data. After that, I compiled a list of features of the Facebook app (The whole list can be found in the Appendix section). This is not an exhaustive list. Since it was manually made by myself by observing the app, further study can be done to ensure the quality of this list. After creating the list I matched the words from the list with the ones from the dictionary and cross-checked for both the playstore and appstore keywords. As a result, this

would leave us with the common features that are talked about in both the apps, features that are talked about in the Android version and not the iOS version and vice-versa, and finally, some keywords that are not features of the app but are talked about regardless. All of the major findings will be discussed in the Results Section and non-major ones will be in the Appendix.

2.5 Novelty

To cross the baseline attempts were made to maximize the accuracy and minimize the loss.

1. To minimize the loss I propose a customized loss function as a weighted combination of 'Binary Cross Entropy' (BCE) loss and 'Mean Squared Error' (MSE) loss. Specifically, I calculated, $Loss = 0.5 * BCE + 0.5 * MSE$.

2. Since the dataset is highly imbalanced, i.e. there are 10 times more negative classes in comparison to the positive classes, the model does not perform as expected. To solve this issue I have made attempts to oversample the minority class. Specifically, I have used '**SMOTE**: synthetic minority over-sampling technique' [1].

Results are reported in the following section having used the general strategies and the strategies previously mentioned.

3 Results and Discussion

3.1 SVM:

For the experiments, SVM with linear kernel was chosen. Testing the data for SVM initially provided an accuracy of 94.88%. To calculate loss for SVM generally, hinge loss is used. After getting the results for SVM, to get the best model for this data a hyperparameter search was performed. The train-validation-test split ratio was **70:15:15**

Hyperparameter: includes the different values of the Regularization Parameter 'C'. Using the values of C = [0.5, 1, 5, 10, 20, 50, 100, 1000] to find the values of train and validation loss and test accuracy were calculated. The detailed results can be found here in this Table of the Appendix. The best model was acquired for C = 1. The train validation loss curve is reported in this figure:

3.2 Logistic Regression:

Detailed experimentation was performed for Logistic Regression. Initial implementation of Logistic Regression was performed using the Sklearn library on the imbalanced dataset. Then a PyTorch implementation was made for the imbalanced dataset. After that, using SMOTE, to create a balanced dataset, it was tested again. The findings suggest that, by using the imbalanced dataset we get: **92.45% accuracy,** whereas using SMOTE gives an accuracy of **95.09%**, clearly showing the impact of the oversampling technique of minority class. The train-validation-test split ratio was **70:15:15**

Experiments were also performed for the loss functions. It included experiments for the BCE loss, Mean Absolute Error (MAE) loss, and finally the customized loss. The train and validation

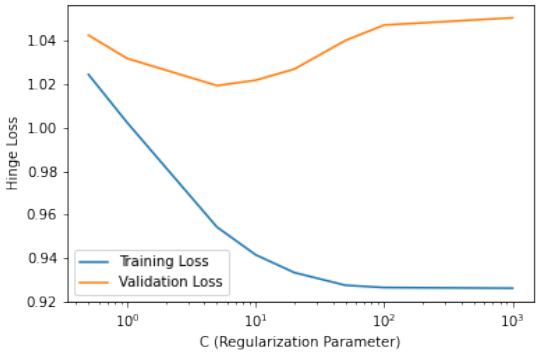

Figure 2: SVM-loss vs C for train and validation

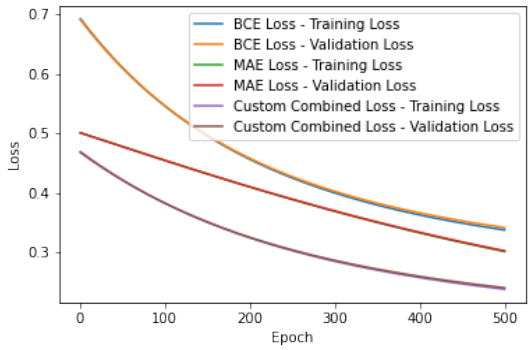

Figure 3: LogReg -loss vs epoch for train and validation on CEloss, MAEloss, Customloss

loss for the same model using the 3 different losses can be found here. The customized loss performs better on the train and validation set.

Further tests are mentioned in the appendix.

Hyperparameter: For the optimal hyperparameters extensive search was done on the following components, which are mentioned in this table. The best accuracy model was obtained when trained on the following hyperparameters: {'C': 100, 'class_weight': None, 'max_iter': 500, 'penalty': 'l2', 'solver': 'lbfgs', 'sampling_strategy': '0.2'}

3.3 RoBERTa:

Similar to the experiments of Logistic Regression, the dataset was tested using RoBERTa-Large model also. Both the imbalanced and the minority class oversampled dataset was used for testing. The findings suggest that by using the imbalanced dataset we get: **92.49% accuracy,** whereas using SMOTE gives an accuracy of **95.43%**, clearly showing the impact of the oversampling technique of minority class. The train-validation-test split ratio was **70:10:20**

Experiments were also performed for the loss functions. It included experiments for the

Hyperparameter Search for Logistic Regression						
C	0.001	0.01	0.1	1	10	100
max_iter	100	500	1000			
solver	lbfgs	liblinear				
class_weight	None	Balanced				
penalty	L1	L2				
sampling_strategy(only for SMOTE)	0.2	0.5	0.8			

Table 2: Hyperparameter Search of Logistic Regression

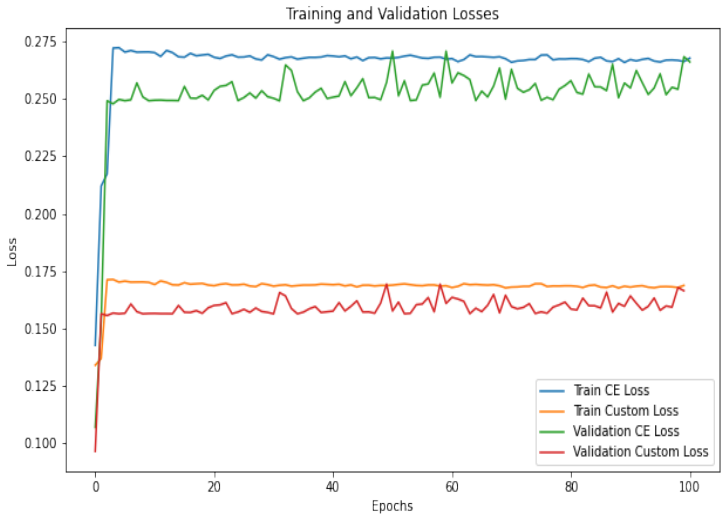

Figure 4: RoBERTa-Large -loss vs epoch for train and validation on CEloss, Customloss

CE loss and the customized loss. The train and validation loss for the same model using the 2 different losses over 100 epochs can be found here. The performance is not as good in this case, leading to believe that there needs to be more experiments on this and more complex loss functions may be required for LLMs.

Further tests with accuracy and individual curves of the losses are mentioned in the appendix.

Hyperparameter: For the optimal hyperparameters extensive search was done on the following components, which are mentioned in this table. The best accuracy model was obtained when trained on the following hyperparameters: {'batch_size': 16, 'learning rate': 1e-05, 'model_name': 'roberta-large', 'num_epochs': 3, 'sampling_strategy': 0.2}

3.4 LDA:

The experiments with LDA provided the following topics and keywords for the Android version, that match the features set of Facebook: *'messenger', 'photos', 'song', 'video', 'button', 'mar-*

Hyperparameter Search for RoBERTa			
Learning Rate	1e-5	2e-5	5e-5
Batch Size	4	8	16
Epochs	1	2	3
Sampling Strategy (only for SMOTE)	auto	0.2	0.5
Model Variant	RoBERTa-Large	RoBERTa-Base	

Table 3: Hyperparameter Search of RoBERTa

ketplace', 'notification', 'reels', 'story', 'profile', 'friend', 'event', 'timeline', 'groups', 'posts', 'comments', 'share', 'privacy', 'pages', 'ad', 'memory', 'message'

The experiments with LDA provided the following topics and keywords for the iOS version, that match the features set of Facebook: *'photos', 'video', 'marketplace', 'notification', 'reels', 'friend', 'groups', 'status', 'posts', 'comments', 'pages', 'ad', 'dating', 'reacts', 'privacy', 'album', 'login'*

Thus the common features that are talked about are: *"photos, video, marketplace, notification, reels, friend, groups, posts, comments, pages, ad"*

Since the sentiments associated with these features are negative, we can conclude that people are facing issues with these features. For example, it could be that for the 'video' feature either the volume or the duration scroller may not be working properly, etc.

The other statistics and total feature set will be provided in the appendix section.

Hyperparameter: The hyperparameter for this task was selected to be the topic number and keywords for each topic. The topic number was searched for 5, 10, 20 topics, and 5 keywords for each. Through manual observation, it was found that 10 topics and 5 keywords for each yielded the best results giving us the important features and some small amount of irrelevant information. The 5 topics had lots of key features missing and the 20 topics had lots of irrelevant words like- nothing, standards, etc.

4 Conclusion

The experiment and the final outputs show interesting results. The features that have been talked about in both the apps, need to be improved and be more tailored to user needs. The sentiment accuracy and loss function could be improved further through more experimentation, which will be a future work for this project. One interesting find is that, for Logistic Regression, although the customized loss and the MAE loss start approximately at the same point in the graph after 500 epochs the difference between them is evident. This shows the effectiveness of the loss function which could be a new avenue of study. The SMOTE technique for minority classes has shown better performance and better accuracy.

References

[1] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

[2] Emitza Guzman and Walid Maalej. How do users like this feature? a fine grained sentiment analysis of app reviews. In *2014 IEEE 22nd International Requirements Engineering Conference (RE)*, pages 153–162, 2014.

[3] Gabriel Roccabruna, Steve Azzolin, and Giuseppe Riccardi. Multi-source multi-domain sentiment analysis with bert-based models. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pages 581–589, 2022.

[4] Mike Thelwall, Kevan Buckley, Georgios Paltoglou, Di Cai, and Arvid Kappas. Sentiment strength detection in short informal text. *Journal of the American society for information science and technology*, 61(12):2544–2558, 2010.

A Appendix

A.1 SVM experiments

	Training Loss	Validation Loss	Test Accuracy
C = 0.5	1.0246146922645785,	1.0427163875823457,	0.941747572815534
1	1.0024401438419603,	1.0320195456398344,	**0.9488084730803178**
5	0.9542532474671601,	1.0194839114277818,	0.9373345101500441
10	0.9414695904894332,	1.0219436637521384,	0.9324801412180053
20	0.9332727182104119,	1.0270904301715917,	0.9232127096204766
50	0.9274112335224871,	1.040285997254177,	0.910414827890556
100	0.926364359789048,	1.047406166955802,	0.9029126213592233
1000	0.9260446943310189,	1.0507430678440173,	0.8984995586937334

Table 4: Train, Validation Loss & Test Accuracy for different values of C for SVM

A.2 Logistic Regression Experiments

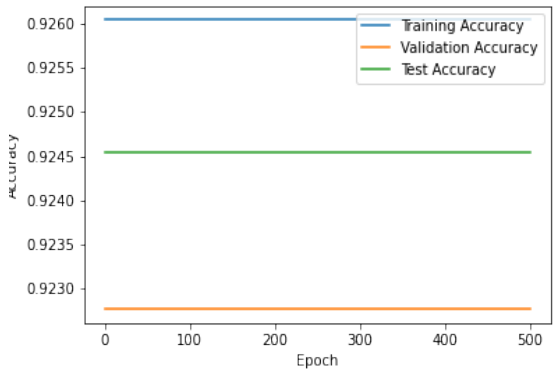

Figure 5: Logistic Regression(Sklearn implementation) curve: epoch vs accuracy

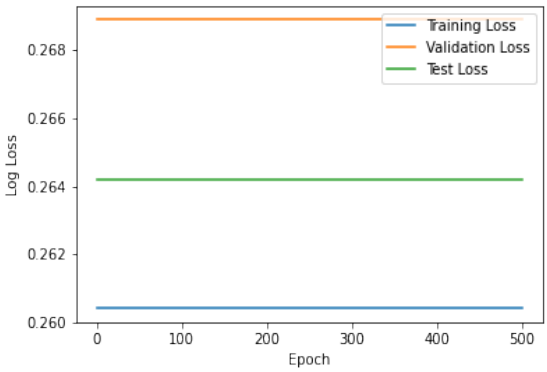

Figure 6: Logistic Regression(Sklearn implementation) curve: epoch vs loss

A.3 RoBERTa Experiments

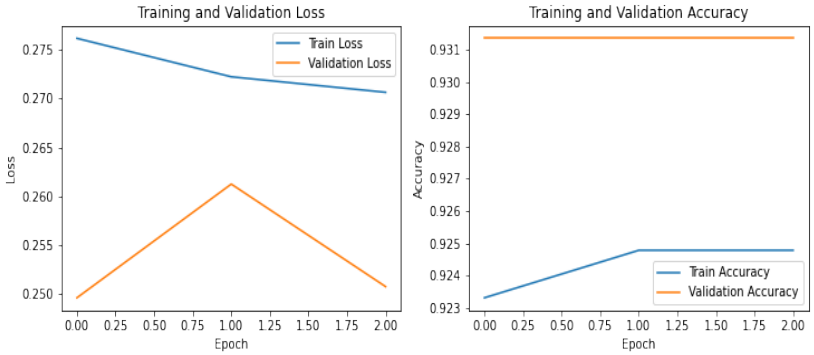

Figure 7: RoBERTa(3 epochs) curve: epoch vs accuracy

A.4 Facebook Features List:

This is the Facebook feature list used in this study, obtained manually from regular users: *'messenger', 'newsfeed', 'photos', 'song', 'video', 'back', 'button', 'gaming', 'marketplace', 'notification', 'menu', 'reels', 'story', 'cover', 'cover photo', 'profile', 'friend', 'event', 'timeline', 'groups', 'status', 'save', 'posts', 'like', 'comments', 'share', 'reacts', 'privacy', 'album', 'login', 'signup', 'register', 'pages', 'ad', 'advertisement', 'dating', 'memory', 'message'.*

A.5 Features of the Android app version that are less talked about:

This is the Facebook feature list used in this study, obtained manually from regular users: *'newsfeed', 'back', 'gaming', 'menu', 'cover', 'cover photo', 'status', 'save', 'like', 'reacts', 'album', 'login', 'signup', 'register', 'advertisement', 'dating'*

A.6 Features of the iOS app version that are less talked about:

This is the Facebook feature list used in this study, obtained manually from regular users: *'messenger', 'newsfeed', 'song', 'back', 'button', 'gaming', 'menu', 'story', 'cover', 'cover photo', 'profile', 'event', 'timeline', 'save', 'like', 'share', 'signup', 'register', 'advertisement', 'memory', 'message'*

A.7 Some common talked about topics in both versions that are not the features of the app:

Facebook, update, phone, people, time, updates, speech, standards, data, place, community, option, accounts, apps, money, meta, account, fb