CmpE 492 Project: Automated User Feedback Classification

Project Advisor: Fatma Başak Aydemir
Project Members: Olcayto Türker, Berkay Alkan, Sertay Akpınar

1. Introduction and Motivation

In app stores, there are thousands of user comments given for apps. Analysis of these user feedbacks has a huge importance. However, it is hard to analyze large amounts of user comments manually. In this project with natural language processing and machine learning techniques, it is aimed to classify the user feedback with their sentiments.

By clustering user feedback from app stores, app companies could plan their future releases. User level of satisfaction from the app could be measured. For instance if users are not happy with UI, this can be observed from the reviews and changes could be implemented.

Furthermore, bugs that are not known could be fixed by taking account of the user comments. Some users specify why they are happy or not when they are using the app. The ones that are querulous may stop using the app. This could be prevented by improving the common complaints.

What's more, showing user feedback is considered is very important to users. They feel precious and recommend the app more to their friends and families. In this way, app companies could increase their profits by both keeping their current users and reaching out potential users.

2. State of the Art

Since measuring customer experience became very important for utilizing user reviews on app stores take an important place for app development companies. That increases the number of scientific research papers about this topic.

We analyzed four different research papers with four different perspectives. First one was for giving insights about their apps to producer companies. It suggests producers change their release routine and start releasing new versions more often. Researchers claim that with more frequent releases leads to more data about customer feedback. By doing so, they can answer customer needs fastly. Second paper's approach was about fake review detection. It aims to generate more reliable data for machine learning models that review analysts work with. Third one analyses sentimental structure of user reviews and tries to understand their emotional viewpoint. They used different sentiment analysis tools in order to achieve that and gave great results. Last one was about classifying user reviews in order to orient producers better with their versions. They seperated user reviews into 4 different labels and trained models categorizing user reviews under four labels.

We decided to start our study with the last approach. With the base of this approach, we are planning to create a classifier that also analyses reviews sentimentally and get better evaluation results.

However, on the commercial side, there are still too many places in the market. Not many companies are providing a service that extracts user feedback from app reviews.

Both scientific researchers and developers in companies make use of NLP methods for classifying user reviews. Commercial products do not explicitly share their technology but researchers generally use pretrained NLP models. In order to increase their model's success, they apply different solutions for the preprocessing phase. They also work on predicting classes with sentimental analysis of reviews.

3. Methods

Google Play Store reviews of Trendyol app are used in this project. In order to get the data needed, Google Play Developer API is used. API access on the Google Play Console is taken. Trendyol service account is used for setting up API access clients. A JSON file that consists of the necessary information for the service account and a private key file for authorization of the service account are taken from Trendyol. An example project on Github is refactored for this project's purpose. API response is a list that consists of user comments. One list element contains various elements like author name, reply of the review, app version that the user has, review language, etc. The list of comments is first filtered by review language. Only the Turkish comments are taken. Then, actual text and the star rating fields of the comments are extracted. These comments are put on a list and a JSON file is generated. Java programming language is used for extracting the user reviews.

After fetching the reviews, data is labeled manually in order to create a train set. For the labeling process a Python project is created. In this project, user reviews are taken and grouped according to their star ratings initially. Then this data structure is saved in Pickle format.

We defined four labels which are "bug report", "feature request", "user experience" and "rating". Bug report class includes the comments where the user specifies a bug in the app. Feature request class contains the comments which the user makes a wish from the developers. User experience class covers the comments that the user specifies what has happened when using the app. The comments that indicate the level of satisfaction of users from the app belong to the rating class.

Some examples for this process:

- Bug report: "Hesabımda ve bütün adreslerimde kendi telefon numaram olmasına rağmen daha önce aldığım bir ürün için farklı numaraya onay SMS gidiyor."
- Bug report: "Facebookla bağlanamıyorum"
- Feature request: "Kapıda ödeme seçeneği gelsin ya hemen şimdi gelse alışveriş yapıcam"

- User experience: “Aldığım eşya kırık çıktı”
- Rating: “Urun sipariş verdik kargo teslimatı yok.memnun degilim.”

When creating the train set, we pay attention to train data to have an approximate number of comments from all classes. In order to achieve this, we examined an equal number of comments from all star ratings. We needed to do this, because the rating of Trendyol app on Google Play Store is near five stars and most of the five star comments were in the rating class.

We aimed to have at least 600 comments for the training set. Since labeling this many labels as a group would take too much time, we decided to split. However, in order to train data to be reliable, we labeled the first 150 comments together. When labeling the comments, mostly, all of us agreed when giving labels. But, we may not be sure about some comments. At those moments we discussed which part of the comment made us not so sure. We understand each other's mind. We saved the dictionary positions of the comments, then created a JSON file manually using those indexes. Then we divided and manually labeled the comments. We ended up with 800 comments ready to be trained. After that, we created a database and saved all comments with the “review”, “rating” and “category” fields. At the end, we calculated our Fleiss’ Kappa score by labeling 100 comments individually and giving them as input. For this purpose we implemented a Python code.

After creating our manually labeled dataset, the next part of the project was creating the most promising pretrained NLP model for classifying user feedback. We decided to choose a model among various NLP models such as OpenAI’s GPT, ELMo and Facebook’s RoBERTa and Google’s BERT. Since we are working with a Turkish company’s user reviews, we were limited to pick a model that can work with Turkish language. Therefore, we decided on using Google’s BERT model due to its Turkish language support. Following this decision, we found numerous pretrained BERT models in Turkish and proceeded on a Turkish BERT model developed by MDZ Digital Library team at the Bavarian State Library.

Preparing the data for developing our classification model based on the BERT model we used was the first step when creating our model. In order to prepare our dataset, we assigned each class to a number and changed our data frame accordingly.

Labels after refactoring is below:

- Bug Report: 0
- Feature Request: 1
- Ratings: 2
- User Experience: 3

After refactoring the data, model training operations have begun. With the purpose of a better model evaluation, we used K-Fold Cross Validation with K = 5. For each fold, we splitted 20% of the data from each label from our manually prepared dataset and formed our test data.

Remaining part of the data is used as the training data. For training our classification model, we used the ClassificationModel of SimpleTransformers.

Arguments for the model are:

- Use early stopping: In order to stop training when early_stopping_metric doesn't improve (based on early_stopping_patience and early_stopping_delta)
- Early stopping delta: The improvement over best_eval_loss necessary to count as a better checkpoint.
- Early stopping metric: The metric that should be used with early stopping.
- Early stopping patience: For terminating training after this many evaluations without an improvement in the evaluation metrics greater than early_stopping_delta.
- Evaluate during training steps: Performs evaluation at every specified number of steps. A checkpoint model and the evaluation results will be saved.
- Number of epochs: Number of epochs the model will be trained for.

After training the model explained above, we tested our model with the remaining 20% part of our dataset. We applied two different methods for evaluating our model. First, we tried using the eval_model function which is built in the ClassificationModel class. However, outputs of that function did not fully address our goal. Then, we implemented an evaluation function providing prediction results for each label. By using these outputs, we calculated accuracy, precision and recall of each category. Visualization of model test results with a Confusion Matrix based on these outputs as well.

4. Results

Assume X is a category,

- Recall = the number of correctly predicted X values / the number of actual X values.
- Precision = the number of correctly predicted X values / all predicted X values.

We applied 5-Fold cross validation to our dataset. In each run(5 runs in total), we calculated the predicted values of all categories. In the end, we summed up the values calculated from each run and created a heat map by using the seaborn python package.

Since we have more data in user experience and ratings categories, these categories have relatively bigger values in the heat map.
In the heat map in figure a),

- B represents "Bug Report"
- U represents "User Experience"
- F represents "Feature Request"
- R represents "Ratings"

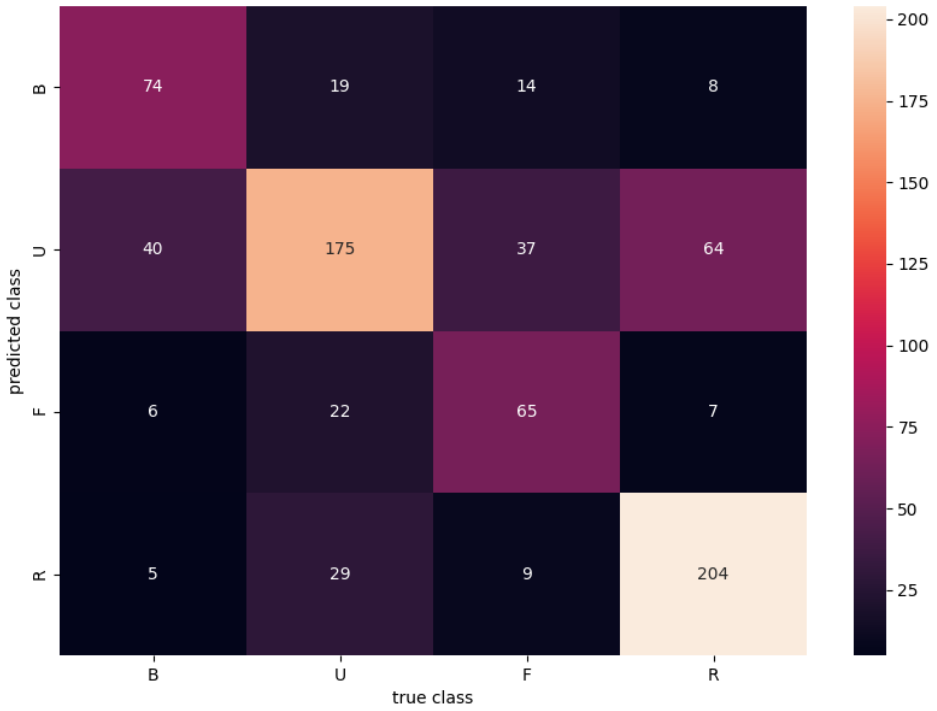

Figure a) Heat Map

- Bug Report
  - Precision: 74 / 115 = 0.643
  - Recall: 74 / 125 = 0.592
- User Experience
  - Precision: 175 / 316 = 0.554
  - Recall: 175 / 245 = 0.714
- Feature Request
  - Precision: 65 / 100 = 0.650
  - Recall: 65 / 125 = 0.520
- Ratings
  - Precision: 204 / 247 = 0.826
  - Recall: 204 / 283 = 0.720

Total Accuracy: 518 / 778 = 0.666

Also we calculated our Fleiss' kappa score as 0.66. This score is not perfect. It represents substantial agreement.

5. Conclusion and Discussion

As in paper “Interrater reliability: the kappa statistic” by McHugh, M.L. in many texts, the minimum percentage for an acceptable interrater agreement is 80%. Thus, our Fleiss’ kappa score needs to be improved. We should find a middle ground when classifying the reviews manually. In this way, we can increase the Fleiss’ kappa score.

Currently, we have 796 rows of data in our database, and the data is not equally distributed between the categories due to the lack of ‘feature request’ and ‘bug report’ reviews. That’s why we need more reviews, especially in ‘bug report’ and ‘feature request’ categories, to obtain a better classification model.

Up to now, we have worked with the Trendyol app’s reviews which is not enough for creating a generic model. That’s why we are planning to extract more data from various apps in the app store or google play store to increase reliability of our model. After we train the model with a bigger database, we believe our model will be more generic.

Our project has a potential impact especially on the mobile app industry. Since we are aiming to classify the user reviews of the mobile apps and automate this classification process, this project can even be converted to a product in the future if we improve the accuracy and genericity of our model.

6. Future Work

For the future progress, we are planning to implement some changes:

- We are going to change the model arguments like epoch numbers in the training process. We will select the best combinations by making k-fold cross validation.
- We used a pretrained model and specialized it in our needs by adding another layer in the training process. We will try bigger sized models for our project and compare the differences.
- Our Fleiss’ kappa score is not enough for a truly reliable agreement. We will set up a guideline for labeling the comment data. We will make tries until we get a high score.
- What’s more, we are going to train a model with a bigger train data set.
- Our test results are on a comment data that has not seen any preprocessing steps. We are planning to add some steps like stemming or lemmatization and observe if these changes improve our model. We are aiming to add sentimental analysis to our data and see if anything is changing. Also, there are too many emojis in the text data. We did not evaluate these emojis specially. We are going to implement a code using external libraries where emojis are taking into account more.
- We are planning to experiment if our model can be generalized or not. We will extract user reviews from various kinds of apps and test our model.
- Lastly we are aiming to finalize our project as a package. When labeled user feedback data is given to this package as input, it would create a resulting report that contains various metrics and classification results for the user.

7. References

- https://towardsdatascience.com/multi-class-metrics-made-simple-part-i-precision-and-recall-9250280bddc2
- https://en.wikipedia.org/wiki/Fleiss%27_kappa
- https://www.researchgate.net/publication/232646799_Interrater_reliability_The_kappa_statistic
- https://simpletransformers.ai/docs/usage/#configuring-a-simple-transformers-model
- https://github.com/stefan-it/turkish-bert
- https://github.com/savasy/Turkish-Bert-NLP-Pipeline
- https://mast.informatik.uni-hamburg.de/wp-content/uploads/2015/06/review_classification_preprint.pdf
- https://mast.informatik.uni-hamburg.de/wp-content/uploads/2014/06/FeatureSentiments.pdf
- https://arxiv.org/pdf/1906.06403
- https://link.springer.com/article/10.1007/s10664-019-09706-9