National Institute of Applied Science and Technology

Carthage University

End of Year Project Report

Mobile app Reviews Analysis : Topics categorization and Fake reviews detection

Authors
Meriem Ben Chaaben
Emna Ksontini
Fares El Kahla

Advisors
Dr. Meriem Chater
Dr. Marouane Kessentini
Mrs Chaima Abid

2019 - 2020

Contents

List of Figures

Chapter 1
General Context

This chapter introduces the project context, a brief description of the ISELAB, of their software refactoring tool and introduces both the problem statement and the research objectives.

1.1 Project Context

This project is elaborated at the National Institute of Applied Science and Technology (INSAT) in collaboration with the Intelligent Software engineering Research Lab (ISELab) at the UNiversity of Michigan. The project is done for the fourth year software engineering, End of Year academic project.

1.2 ISELAB Presentation

Founded by Dr. Marouene Kessentini, The ISELAB is known to be a world-leading research Lab in the areas of Intelligent Software Engineering (also called Search Based Software Engineering), Software Refactoring, Model-Driven Engineering, Software Quality, Machine Learning and Computational Intelligence. The Lab is based in Michigan,and includes students from Canada, France, Tunisia and Japan. It aims to investigate and create intelligent techniques to improve code quality.

1.3 Mobile Reviews:

Studies have shown that nowadays consumers are more vulnerable to crowd influence rather than advertisements and this is what makes testimonials so powerful.
Google Play much like Apple stores as app stores give their users the opportunity to evaluate an app once downloaded. The feedback is received in the form of a review comment and an associated star rating. This constitutes a useful communication channel in which application developers and users can productively exchange information about apps. It

also helps prospective installers decide which apps they would like to try (over 59% of all users check your app's rating before deciding on whether they'll install your app or not). Review scores and comments provided within these platforms are used to determine an overall score for the app.The higher the score an app gets, the more people liked it,at least in theory. In practice, a lot of reviews are less than useful for prospective installers and developers. In fact, an illegal market for fake app reviews has emerged, with the goal to offer services that help app vendors improve ranking in app stores or help other app owners post negative reviews against rivals to drive potential customers away.

1.4 Problem Statement and Project Objectives

In this section, we will introduce the problem statement, then we are going to introduce the project objectives.

Problem Statement
Developers working in the mobile applications industry face a wild rivalry in getting and holding clients. They have to quickly implement new features and fix bugs otherwise they risk losing their user base to the competition. To achieve this goal they must closely monitor and analyze the user feedback as they contain a wealth of information on the issues that users are experiencing.
For feedbacks to reflect genuine user experiences and opinions, developers need to first filter noisy and irrelevant reviews and then map each one to the appropriate topics. Currently these tasks require massive manual effort, sifting and understanding meanings and abbreviations.

Project Objectives :
The ISELAB is currently working on an app that performs quality assessment of mobile apps. It analyzes the code as well as reviews.

Regarding the code part, it computes quality and security metrics, finds the dependencies between packages and generates refactoring recommendations.

And when it comes to reviews, it provides features like number of reviews/reviewers, topics, frauds, languages, sentiment analysis etc.

Our mission was to design, implement and evaluate two AI models one to classify app reviews into categories relevant to software maintenance and evolution and one for detecting fake reviews. The first model is a multi-label classifier which aims to assign reviews to representative topics based on their content, while the second one tries to spot fake feedbacks on the basis of linguistic content and user-behavior using binary classification.

Chapter 2
Basic Knowledge

This chapter will go through all necessary concepts that were involved in this project.

2.1 Preprocessing

Real-world data is often incomplete, inconsistent, and/or lacking in certain behaviors or trends, and is likely to contain many errors. To resolve such issues we had to go through different proven methods of data preprocessing.

2.1.1 Data Collection

Each problem in machine learning has its own unique nuance and approach. And sometimes it can be quite hard to find the specific dataset for the variety of machine learning problems. An inordinate amount of time is usually spent on data collection, which largely consists of data acquisition, data labeling, and improvement of existing data or models.

2.1.2 Data Preparation

To ensure that the dataset used for modelling stage is clean and acceptable we have to think about Data preparation.This would help us avoid misleading predictions.
Data preparation is the process of cleaning and transforming raw data prior to processing and analysis. It is an important step prior to processing and often involves reformatting data, making corrections to data and the combining of data sets to enrich data.

An imbalanced data-set is a set of observations that contains unequal number of instances for different classes. Some methods that are often used to tackle this problem may be classes undersampling(adding more copies of the minority class), classes oversampling (replicating some points from the minority class in order to increase its cardinality) or synthetic data generation.

2.2 Data modeling

During this project, we built two different types of classification models : A binary classification model and a multi-label classification model.
Unlike the binary classifier where each observation is categorized into one of two buckets (0 or 1, True or False), the multi-label classifier assigns to each sample a set of well known labels.

The process of training a model involves providing an algorithm and a “training set” to learn from. The learning algorithm mission was to find patterns that map the input data attributes to the target,yet we had to provide this algorithm with “hyper parameters”. These parameters are values that must be specified outside of the training procedure and that can be used to control certain properties of the training process such as shuffle type, learning rate ...

We also had to think about the training metric and optimize the loss function.

2.3 Model evaluation

The idea of building models works on a constructive feedback principle.We build a model, get feedback from metrics, make improvements and continue until we achieve a desirable accuracy. Evaluation metrics explain the performance of a model. An important aspect of evaluation metrics is their capability to discriminate among model results.
We evaluate our models using precision, recall, and F-measure[1]. Precision is computed by dividing the number of true positives by the sum of true positives and false positives. Recall is computed by dividing the number of true positives by the sum of true positives and false negatives. We use the general form of the F-measure, which combines the precision and recall results, for its computation, $2*$ ((precision$*$ recall)/(precision+recall)).

Chapter 3
Literature review

The main goal of this section is to describe the state of the art in the development of mobile app review mining techniques and algorithms.

3.1 Mobile app review categorization

Paper name	Description	Extracted Topics	Threat
AR-miner: Mining informative reviews for developers from mobile app marketplace [2].	Extracts informative user reviews by filtering noisy and irrelevant ones, then grouping the informative reviews automatically using topic modeling based on content similarity. Algorithms : Naive Bayes, LDA.	-Informative -Non-informative -Other categories extracted automatically using unsupervised learning.	The authors are not professional app developers.The categories extracted by clustering method are not semantically comprehensible by developers eg “chinese, jelly bean” unlike supervised learning which uses a set of specific and well defined labels.
Ensemble Methods for App Review Classification: An Approach for Software Evolution[3].	Identifies 7 categories then using multiple machine learning techniques (Naive Bayes SVM Logistic Regression Neural Network) and stacking techniques.	-Bug report -Complaint -User request -Feature shortcoming -Feature strength -Noise -Praise -Usage scenario	The categories are generic and require developers to manually extract details resulting in a waste of time. Moreover the results achieved are not good.

Analyzing Reviews and Code of Mobile Apps for Better Release Planning[4].	Defines a high and low level taxonomy containing mobile specific categories (e.g. performance, resources, battery, memory, etc.) then splitting them to low level taxonomy - Splits review to sentence and transform the problem to multiclass.	-Bug report -Suggestion new features	Analyzing the text of 1566 user reviews. Sentences often contain a variety of intents: a major intent and some minor intents (i.e. How can I access my profile? Please fix the bug).
Analyzing and automatically labelling the types of user issues that are raised in mobile app reviews [5].	Analyses the issues in the mobile app reviews having 1-2 star review.using multi-labelling approaches Binary Relevance (BR), Classifier Chains (CC), and Pruned Sets with threshold extension (PSt) as well as several different classifiers e.g., support vector machines (SVM), decision tree (J48) and Naive Bayes (NB).	-Additional Cost -Functional Complaint -Compatibility Issue -Crashing -Feature Removal -Feature Request -Network Problem -Privacy and Ethical Issue -Resource Heavy -Response Time -Uninteresting Content -Update Issue -User Interface	Reviews with medium or higher ratings can present information useful for Application maintenance and development (exp : feature requests).

Threats to external validity concern the generalisation of their findings. A threat to the external validity could be represented by the particularity of the apps they selected to extract reviews and use them in the experimentation. Experimental results may be applicable only on the extracted reviews. To mitigate this threat we selected more mobile apps: 51 different apps with 156k reviews.

3.2 Fake reviews detection

Paper name	Description	Threat
What Yelp Fake Review Filter Might Be Doing?[10]	Studies both behavioural and linguistic features, but each for its own. The results allow us to postulate that Yelp's filtering algorithm seems to be correlated with abnormal spamming behaviors.	Assumed that the word distribution of fake reviews does not significantly differ from official reviews.
Detecting Product Review Spammers using Rating Behaviors[9].	Detects users generating spam reviews or review spammers. Identifies characteristic behaviors of review spammers and models them to detect the spammers. Proposes scoring methods to measure the degree of spam for each reviewer and apply them on an Amazon review dataset.	Focuses only on behavioural features and does not classify a review as fake or legit. Applied on amazon reviews not on mobile app reviewers.
Towards understanding and detecting fake reviews in app stores.[11]	Studies fake reviews, their providers, characteristics, and how well they can be automatically detected. Uses supervised machine learning algorithms to detect fake reviews. Random Forest classifier identifies fake reviews, given a proportional distribution of fake and regular reviews as reported in other domains, with a recall of 91 % and f1 score 96 %.	Nearly all reviews were extracted from a single provider. Assumed that the word distribution of fake reviews does not significantly differ from official reviews.

3.3 Conclusion

Taking into consideration all the approaches and the threats studied we noticed three things:
-The scope of the research is most of the time specific to certain domain and can't be generalized.
-The results are not satisfying enough especially in the categorisation problem.
-In neither problem was there an approach that uses deep learning or any form of neural network.

Chapter 4
Topic Identification for mobile apps reviews

4.1 Introduction

Classifying app reviews into categories relevant for software programming development, has been the subject of multiple research papers with different views.
After reading several documents, we have come up with our own implementation.
In this section, we are going to introduce the intuition behind our model, and explain in detail each step.

4.2 Approach and intuition

4.2.1 Data Collection and Preparation:

ISELAB provided us with a Multi-Label dataset that contains reviews posted on Play store.
Our dataset consists of 154497 english reviews. These Feedbacks were selected to cover 33 categories : “Complexity” , “Design UX” , “Use cases” , “Feature Requests” , “Frequency” , “Update” , “Camera & Photos” , “Video” , “Performance” , “Security Accounts” , “Streaming” , “Devices” , “Privacy” , “Connectivity” , “Notifications & Alerts” , “Audio” ,“Gaming” , “Customer Support” , “Location Services” , “Sign Up & Login” , “Advertising” , “Payment” , “Pricing” , “Social & Collaboration” ,“Battery” , “Internationalization” , “Operating System” ,” HDMI” ,“Import Export” ,” bug” ,”' satisfied user” ,” dissatisfied user” and ” Touch ID”.

Data Visualization: As shown in the figure 1.0 the dataset seemed to be imbalanced. we have more observations in some specific topics compared to the others (for example Design & UX was identified as a topic in 70000 observations while only 1 instance is spotted for the Touch ID Category) the problem is that models trained on unbalanced datasets often have poor results. In fact, The algorithm receives significantly more examples from one class, prompting it to be biased towards that particular class.

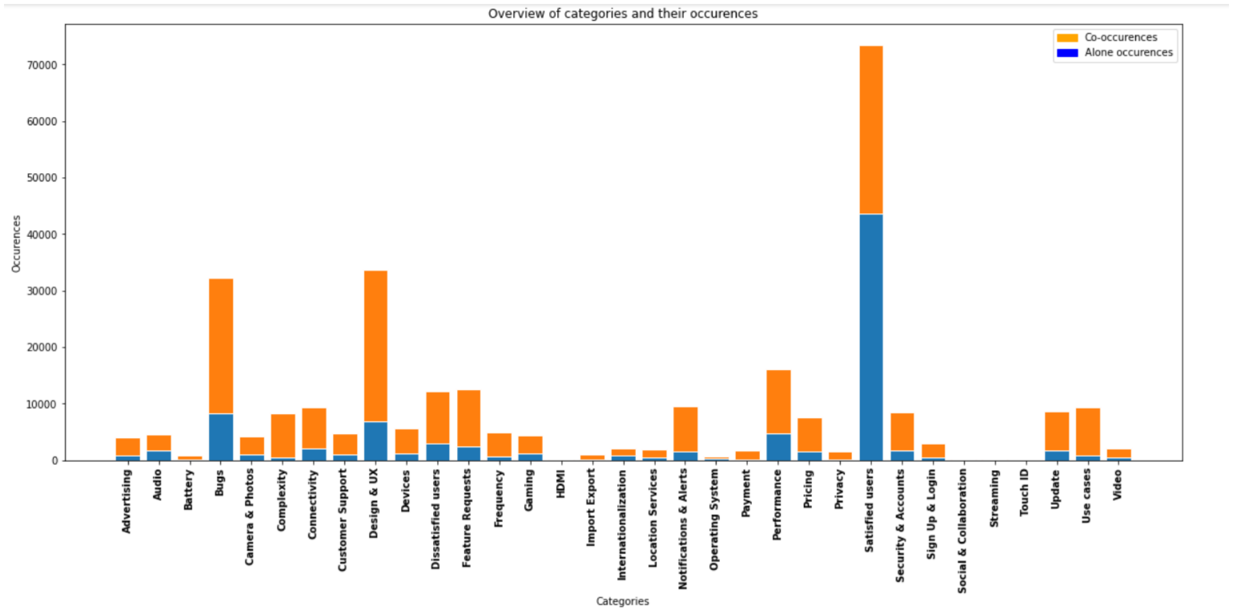


Figure 4.1: Overview of categories and their occurrences

Balancing the data: Based on the barplot it was necessary to think about rebalancing the dataset before fitting a classifier on it. So we decided to:

- Remove Categories with very low occurrences like "Touch ID".
- Delete insignificant topics like Bugs , Satisfied user and Dissatisfied user(Both Satisfied user and Dissatisfied user are not considered as categories but instead as sentimental labels. We only focus only on the reviews claiming about bugs or asking for new features).
- Merge close categories based on fields. For example we combined the following topics: "video" with "streaming" and "audio", "Pricing" with "Payment" and "connectivity" with "HDMI".
- Add synthetic samples manually to increase instances of important topics like "Operating System" (We tried to include different version of OS within the added observations).

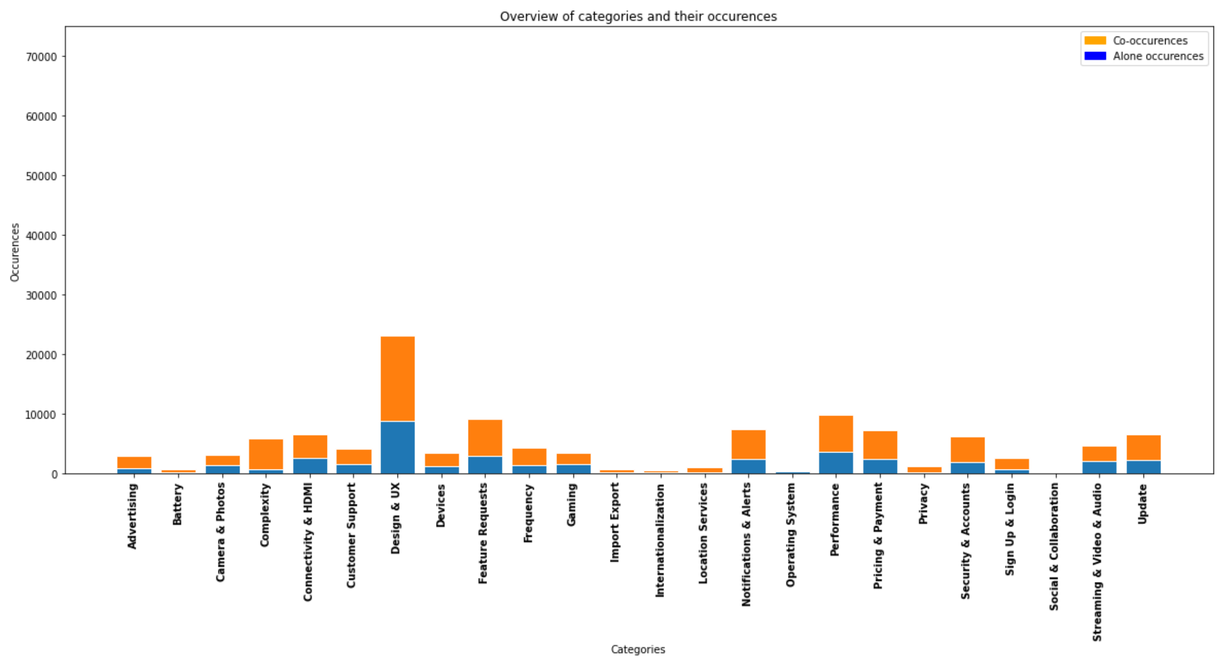


Figure 4.2: Overview of categories and their occurrences after balancing the dataset

N.B: Oversampling may seem a solution. However, in our case with such a huge dataset oversampling may lead to overfitting. The generated observations may not be a true representative of the actual minority class.

Text preprocessing:

The first step in text categorization is to transform documents, which typically are strings of characters, into a representation suitable for the learning algorithm and the classification task.
In this step we worked on preprocessing the user feedback extracted from the subject and the body of the review by implementing the following pipeline:

- **Tokenization** : user feedback is transformed into lists of words (i.e.,tokens) which will be the atomic part of the next steps, excluding numbers, punctuation and emojis that usually do not contain information.
- **Stopword removal** : the tokens are intersected with a list of common words that are frequent in written English (e.g.," the"," a",and "an") and only introduce noise to NLP activities.
- **Stemming** : the words are reduced to their stem form (e.g "liked"," liking"," likes" is replaced with the same stem of "like"). This step reduces the number of tokens and thus the complexity of the NLP work.
- **Count Vectorization** : Transforms the dataset to count matrix containing the number of occurrences of each token.

- **Term frequency - inverse document frequency (tf-idf [8])** : reduces the relevance of too generic tokens that are contained in several reviews. More formally, the tf-idf score is defined according to the formula

$$\textbf{tf-idf}_{t,d} = tf_{t,d} \times idf_t = tf_{t,d} \times log(\frac{N}{df_t})$$

  tft,d: is the raw frequency (number of occurrences) of word t in review d
  Dft: is the number of reviews containing the word t
  N : total number of reviews

For example, the word 'app' would be penalized because it occurs in many user reviews whereas an uncommon word like 'omission' would be given a higher weight.

As the final result, the process outputs a set of bag-of-words, fixed-length vectors for each feedback that will be the input of the subsequent Models.

4.2.2 Models implementation

We used the NLP features extracted in the previous phase to train Machine Learning techniques.
Traditional ML Algorithms : We started by implementing traditional techniques of text classification such as Naïve Bayes Classifier (NBC) which was very popular and computationally inexpensive In the earliest history of information retrieval, Support Vector Machine (SVM) which is another popular technique that researchers use as a baseline to compare against their own works to highlight novelty and contributions. Finally logistic regression (LR) which is considered as one of the simplest classification algorithms and it has been addressed in most text classification problems.

Nb: Since our problem is a multilabel classification the implemented algorithms that will be listed below are mostly conceived to be used in a binary classification, so tackling this issue we opted for using a One vs Rest [6] strategy which consists in fitting one classifier per class. For each classifier, the class is fitted against all the other classes.

We used this paper [7], that describes several machine learning classifiers and approaches to deal with multi-class/multi-label problems as a reference to implement and judge the efficiency of the traditional algorithms.

Multinomial Naive Bayes: Multinomial Naive Bayes, is a **generative** model that explicitly models the actual distribution of each class using Bernoulli distribution.
This algorithm predicts the tag of a text. It calculates the probability of each tag for a given review and then outputs the tag with the highest probability.
As shown in Table 4.1, we achieved 89% precision, 10% recall, 19% F-score.

linear Support Vector Machine (SVM) with Stochastic Gradient Descent (SGD) optimizer : An algorithm that determines the best decision boundary between vectors that belong to a given topic and vectors that do not belong to it. It's known that most text categorization problems are linearly separable, thus the idea of SVMs is to find such linear (or polynomial) separators.
SGD is an optimisation technique used to speed up the training time. To put it simple, instead of calculating the cost function of the whole dataset to update the model parameters, the sgd optimiser calculates that function for only a part of the data at each step.
As reported in table 4.1 we obtained additional improvement in terms of recall, precision and F-score up to 18%, 5% and 25%, respectively.

Result Analysis :
Taking into consideration only the precision would lead us to think that the model did perform well however the bad recall score means that the model is predicting most of the labels as false negative.

Logistic Regression : Logistic regression is a **discriminative** classifier that extracts real-valued features from the input, multiplies each by a weight, sums them, and passes the sum through a sigmoid function to generate a probability. A threshold is used to make a decision.The weights (vector w and bias b) are learned from a labeled training set via the loss function, the cross-entropy loss, that must be minimized.

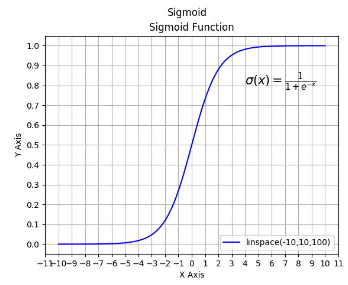
$$\textit{Cross Entropy} = -\overset{③}{\frac{1}{N}} \overset{②}{\sum_{j}} \overset{①}{y_j * \log(\hat{y}_j)}$$

① true label * log(predicted)

② sum over all sequences

③ divide by the number of samples

(a) cross entropy formula



(b) sigmoid function

Figure 4.3: loss function and logistic function

Given the evaluation results, the logistic model achieved a higher overall recall, 65%.

Interpretation :
The one vs rest approach did permit us to train a multi label classification and we achieve pretty satisfying results with logistic regression. However this approach intensified the data imbalance problems.
In fact, for each label its appropriate model had a bias toward classifying that label as negative (each model was trained with data where more than 90% of the instances were negatively labelled.)

Results		Recall	F1	Precision
ML Algorithms	multinomial Naive Byaes	0.1	0.19	0.89
	SVM boosted by SGD	0.28	0.44	0.94
	Logistic regression	0.65	0.75	0.9

Table 4.1: Results' comparative table

Conclusion :
The One Vs rest is fairly reasonable when the total number of classes is small, but becomes increasingly inefficient as the number of classes rises.
In fact, decomposing the multi label classification problem into multiple sets of classification problems – one for each label - has a significant disadvantage, it does not take into account dependencies between different categories. Hence a different approach needs to be employed.
After further research we have chosen to build a neural network model since the common hidden layers would treat the inter-dependency between different categories.

Deep Neural Network
Lately, deep learning approaches have achieved surpassing results in comparison to previous machine learning algorithms on tasks such as image classification, natural language processing, face recognition, etc. The success of these deep learning algorithms relies on their capacity to model complex and non-linear relationships within data as well as handling huge data.

After tuning, we settled up with a shallow neural network, containing one hidden layer with ReLU activation function.

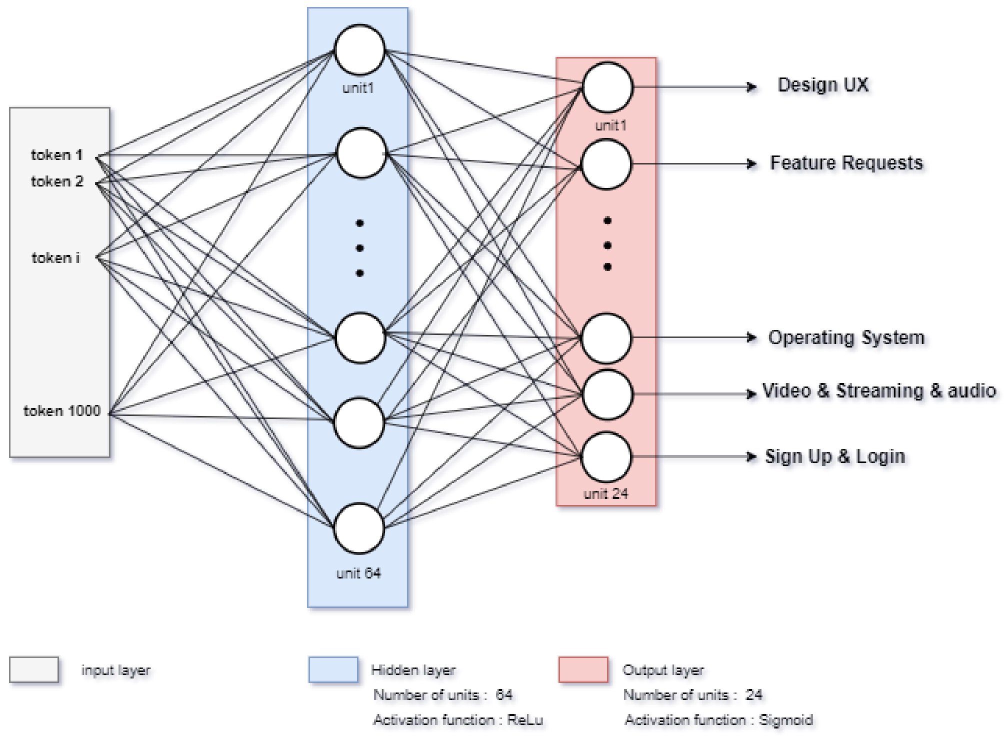
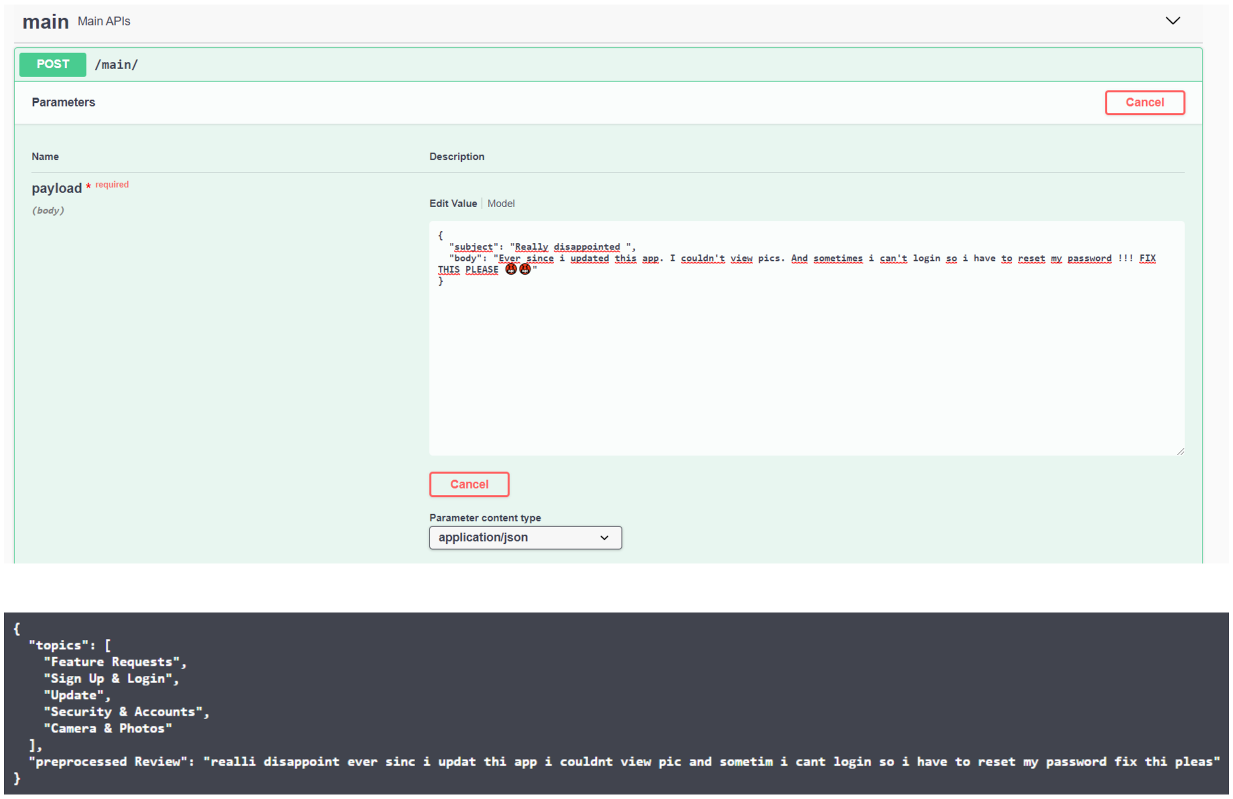


Figure 4.4: Overview of the shallow neural network for topics categorization

Using the sigmoid activation function permits us to extract the probability of each category being related to the review ranging from 0 to 1.
The threshold is the hyper-parameter we use to binarize continuous values, in our case we have a threshold used to consider if the category is present or not depending on the probability provided by the model, and the default value is 0.5 (50%) which means that if the model is more than 50 % certain that one category is present the prediction algorithm will consider it.
Since our model is a multi label classifier and the number of categories is quite big (24), it may occur that the model does not detect all the labels with a certainty of over 50 %, so we had the idea to lower the threshold and after tuning we ended with a threshold of 0.3 the f1 score increased to 81.16 %.
After analysing the reviews we discovered that they contain multiple sentences separated by punctuation. We exploited this information by making the model predict each sentence alone then combine the detected categories instead of predicting the whole review. This boosted the recall which resulted to a further increase of 2 % in f1 score to 83.34 %.

4.2.3 API /Integration

Building a model is generally not the end of a machine learning project. Even if the purpose of the model is to increase knowledge of the data, the knowledge gained will need to be organized and presented in a way that the customer can use it.
In our case we created an api that takes as input a review in form of head and body then returns all the detected categories referenced in this review.
For better transparency we return the review after preprocessing to give the API user insights of the process for example the words that were ignored.
We started by using the pickle operation to serialize our deep learning algorithm and save the serialized format to a file.
We then serialized the count vectorised and Tf Idf parameters.
We created an API using flask for better compatibility and to easily import the learnt model
For better visualization, we used Swagger, a set of open-source tools built around the OpenAPI Specification that helped us design, build, document and consume REST APIs.



```
{
  "subject": "Really disappointed ",
  "body": "Ever since i updated this app. I couldn't view pics. And sometimes i can't login so i have to reset my password !!! FIX THIS PLEASE 😡😡"
}
```

```
{
  "topics": [
    "Feature Requests",
    "Sign Up & Login",
    "Update",
    "Security & Accounts",
    "Camera & Photos"
  ],
  "preprocessed Review": "realli disappoint ever sinc i updat thi app i couldnt view pic and sometim i cant login so i have to reset my password fix thi pleas"
}
```

Figure 4.5: API Testing With Swagger

Chapter 5
Fake Reviews Detection

5.1 Introduction

In the previous section we suggested a classifier that derives actionable information for software teams from reviews. However, we haven't considered fake reviews and their implications. Negative fake reviews, e.g., by competitors reporting false issues, can lead to confusion and waste developers' time. Positive fake reviews might also lead to wrong insights about real users needs and requirements.
In this section, we tackle this problem by building a classifier that takes the review text and the basic information of its reviewer as input and outputs whether the review is reliable.

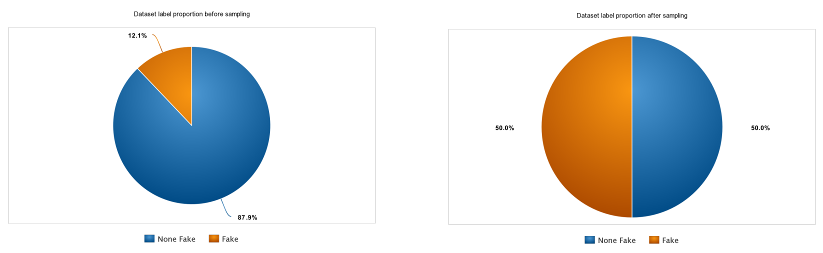
5.2 Approach and intuition

Unlike review categorisation, people can't really spot fake reviews reliably (especially those well written ones).
So following the traditional approach of manually labeling a dataset, then using it for training and testing is not possible in our case.

5.2.1 Yelp fake reviews dataset

After several readings about fake reviews detection in different domains, we found out that most review sites still do not publicly filter fake reviews. Yelp [x] is an exception which has been filtering reviews over the past few years. However, Yelp's algorithm is trade secret.
Yelp.com is a platform for user reviews,different reviews are posted about multiple domains like restaurants, hotels, barbers ...
It has for years implemented a fake reviews detection system that is considered very robust, Yelp has been progressively refining a fake review detection system which is currently considered a very robust one.
We had the intuition of using a dataset containing both reviews that are considered fake and legitimate by Yelp as our dataset then try to construct a generic model that uses features which can be exported to mobile apps reviews.

Data Visualization The dataset is collected from Yelp.com and firstly used by Rayana and Akoglu [7] and it includes product and user information, timestamp, ratings, and a plaintext review.



(a) dataset label proportion before sampling

(b) dataset label proportion after sampling

Figure 5.1: Yelp Fake Reviews Data Visualization

As shown in the pie chart the dataset seemed to be imbalanced. Only 12.1% of the reviews are considered fake, such disproportion may mislead the binary classification model.

Balancing the data:

Based on the pie chart it was necessary to think about rebalancing the dataset before fitting a classifier on it. We couldn't apply the solution of merging/removing labels that we used in the review categorisation since we only have two labels. We had a choice of using undersampling and oversampling. The main disadvantage with oversampling, from our perspective, is that by making exact copies of existing fake reviews (minority class), it makes overfitting likely.
Therefore, we opted for **undersampling** which is in our case a reasonable and valid strategy since the minority class has 8500 occurrences which is sufficient for a classification model, and we don't risk underfitting.

preprocessing

As mentioned in the topic categorization preprocessing section, textual preprocessing consists of the following steps :

1) **Tokenization**
2) **Stopword removal**
3) **Stemming**

4) **Count Vectorization**

5) **Term frequency - inverse document frequency (tf-idf)**

The preprocessing phase outputs a set of bag-of-words that will be the input of our review-centric textual Models.

Model Implementation & Evaluation :
After tuning, we settled up with a deep neural network, containing two hidden layer with ReLU activation function.

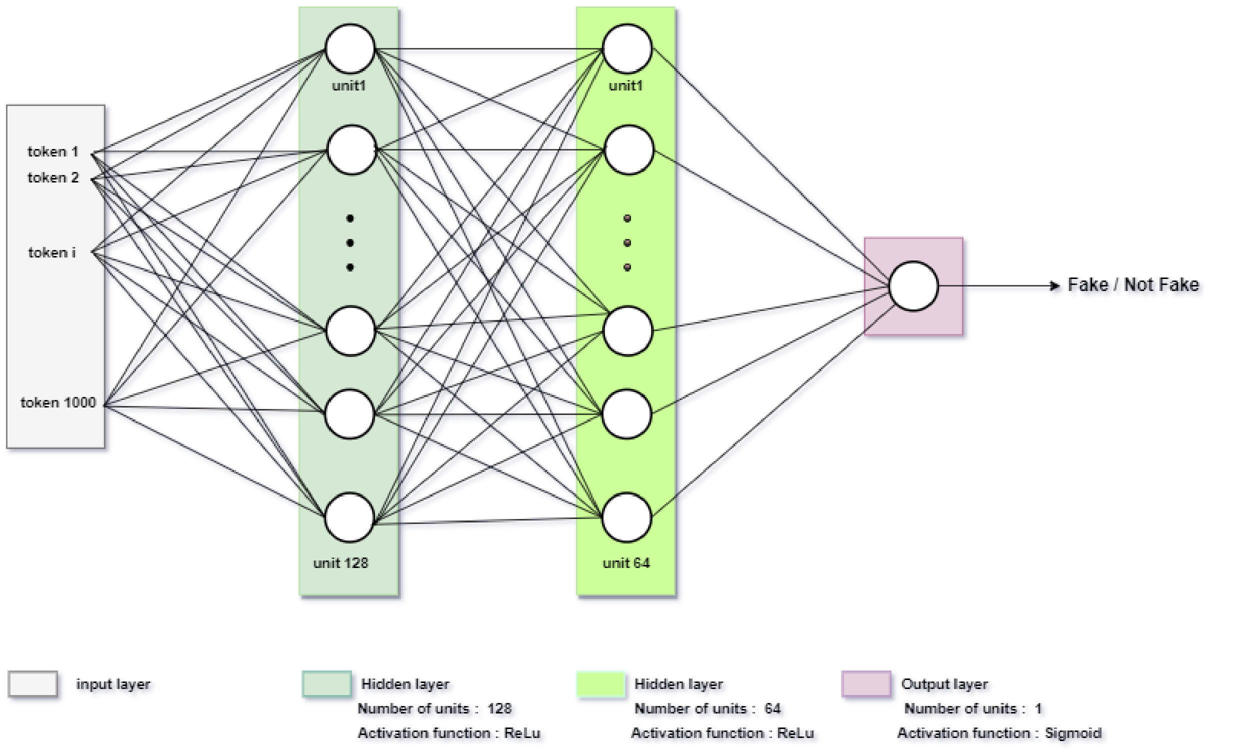


Figure 5.2: Overview of the Textual Model

As expected, the results were not satisfying, we only got an f1 score of 64.9 %, a recall of 67.44% and precision of 61.08 %.

5.2.2 Statistical Model :

In order to boost the results, we included a huge number of statistical features related to the text review such as number of each part-of-speech (POS) tag (adverbs, verbs, nouns..), the rate of misspelling, word count, some Psycho Linguistic variables...
preprocessing :
We had to go through a selection phase where we check the correlation between a numerical feature and whether a review is fake or not. Plotting The CDF (cumulative distribution function) helped us investigate the contribution of each feature.

Below are the empirical CDFs for 3 features : length in character for both fake and non fake reviews, number of money references(psycho linguistic variable) and misspelling rate. The 4th plot illustrates the percentages of pos tags.

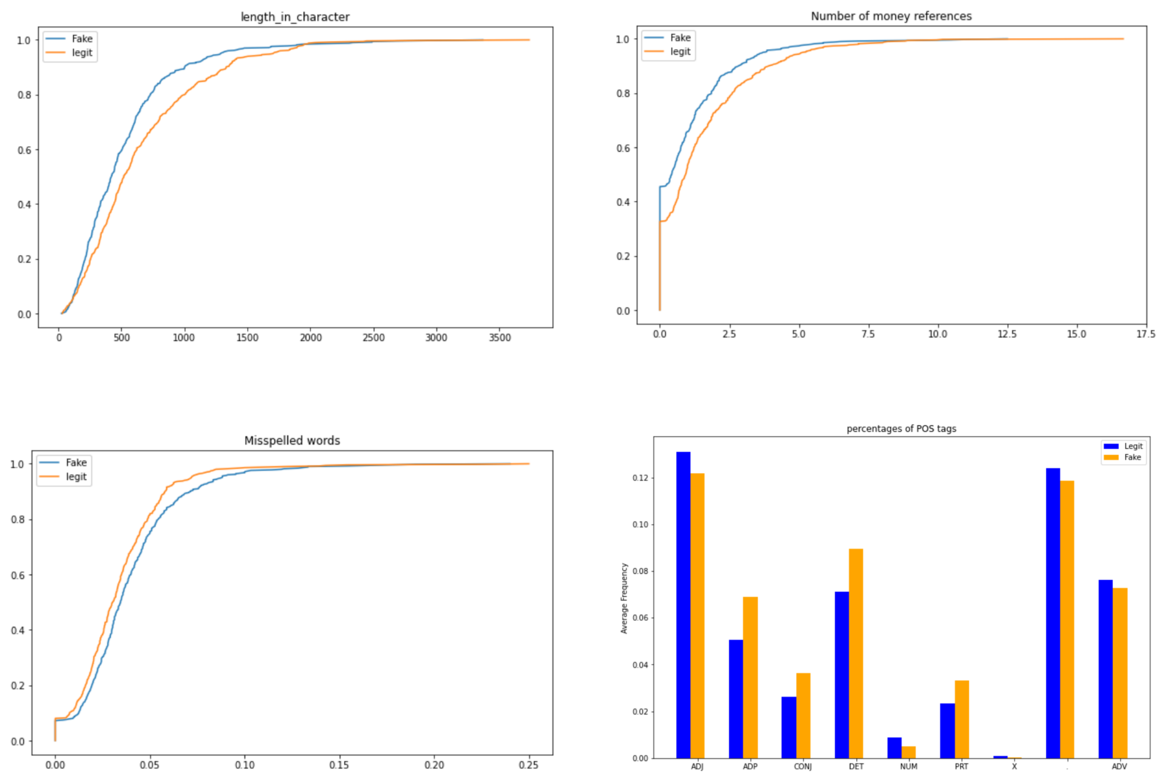


Figure 5.3: Statistical Features Analysis

We can observe that fake reviews tend to have shorter length, also spammers rarely write misspelled words. Nonetheless, there is no significant difference in the percentages of POS tags.

To face the large variance of data, all the features are normalized between zero and one.

Model Implementation & Evaluation :

After tuning, we settled up with a deep neural network, containing two hidden layers with ReLU activation function.

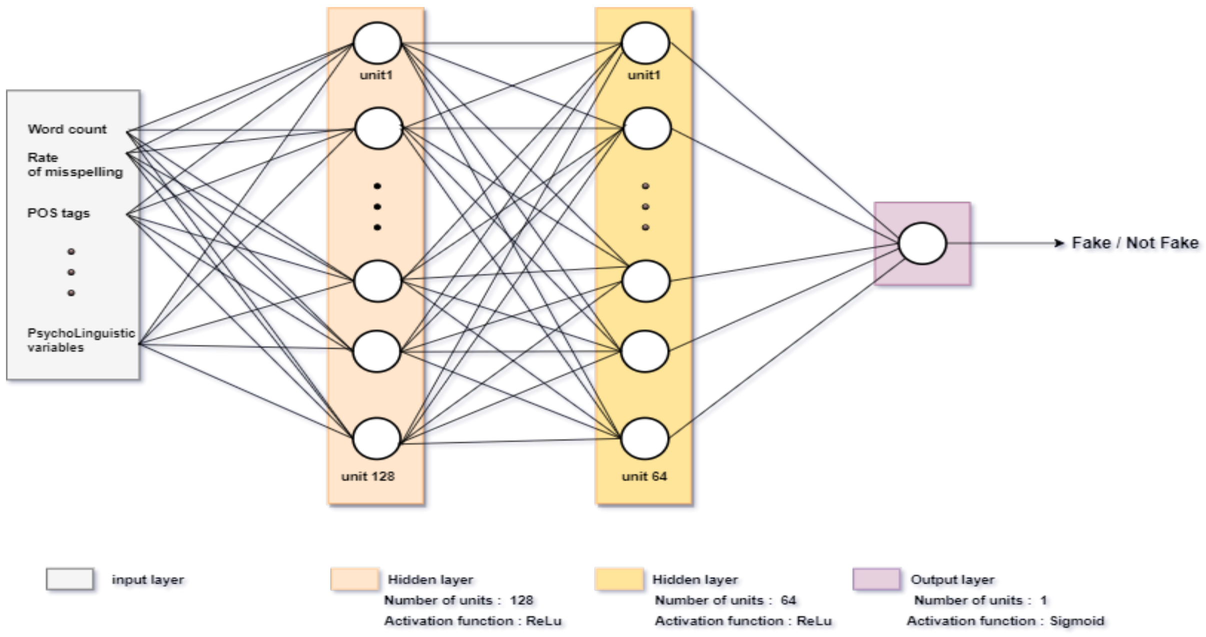


Figure 5.4: Overview of Statistical Model

The model gives a precision of 60.12%, and a recall of 54.88% which is close to the random guessing of 50%.

Review-centric model: statistical model and textual model combination

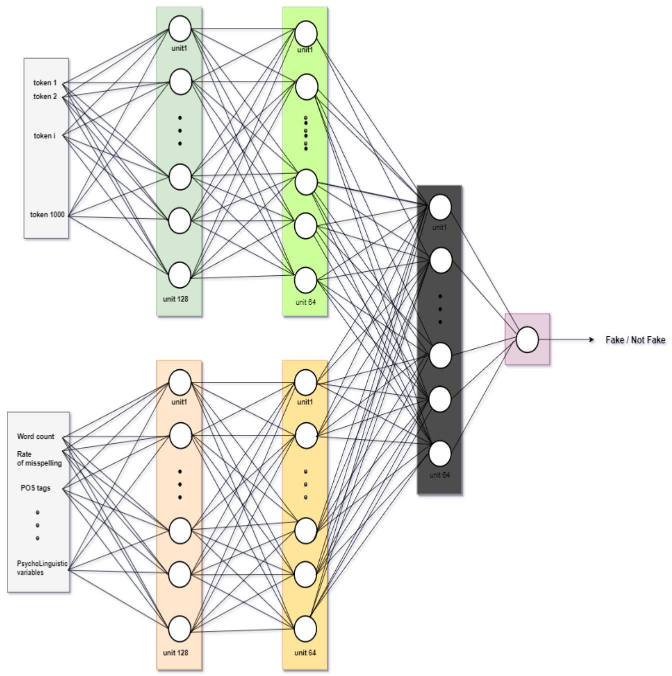


Figure 5.5: Overview of the Statistical-Textual model combination

Including numerical features to the previous model, boosts precision by about 10% and recall by around 8%, resulting in around 9.65% improvement in F1.

5.2.3 Reviewer-centric model : behavioural model :

In this section, we study detection performance using behavioral features.
preprocessing :
For each review we add the behavioral feature of its reviewer.
We went through the features selection process again and we separated reviewers in our data into two groups:

- **Spammers**: Authors of fake reviews.
- **Non-spammers**: Authors who didn't write fake reviews in our data.

We analyzed the reviewers' profile on the following behavioral dimensions:
Jacquard distance(uni-grams intersection in all reviews), Average rating, Semantic similarity, Rating deviation from average app rating, Max reviews per day... For each dimension we retrieved the std and the mean or the min and max of the values.

Below is the empirical CDF of the reviewer rating deviation from average rating it shows that spammers are likely to give a rate close to the app rate. To measure reviewer's deviation, we first compute the absolute rating deviation of a review from other reviews on the same business. Then, we compute the mean deviation of a reviewer over all his reviews.

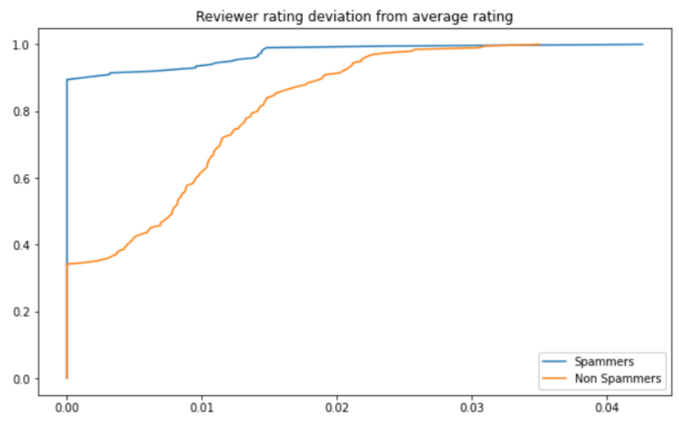


Figure 5.6: Reviewer rating deviation from average rating

Model Implementation & Evaluation :

After tuning, we settled up with a deep neural network, containing two hidden layers with ReLU activation function.

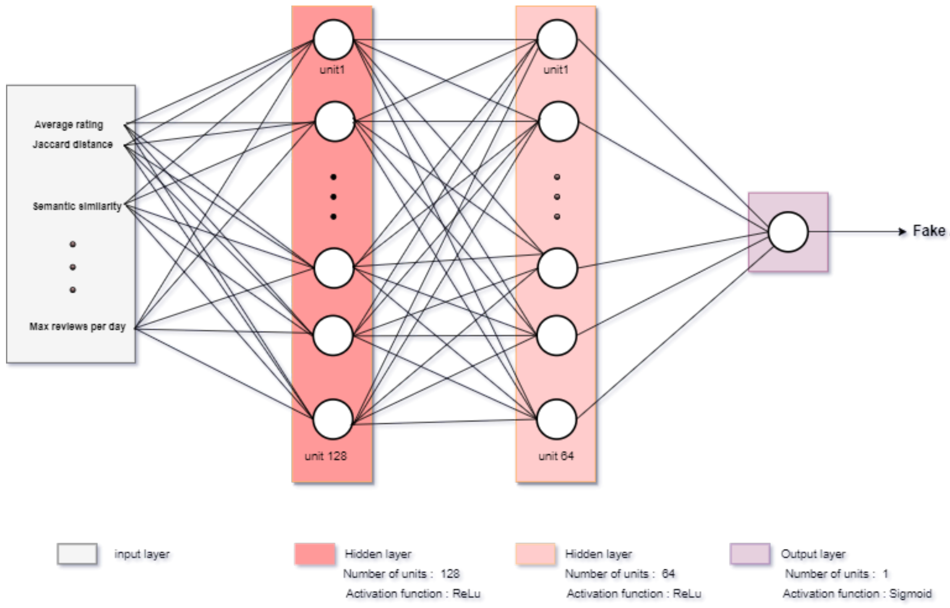


Figure 5.7: Overview of Behavioural Model

Using only behavioral features, neither the precision nor the recall results were improved.(66.92% as precision, 61.17% as recall)

5.2.4 Combined Model :

The two previous models gave similar and average results, but if we inspect the features used in each one we can notice that they are independent which made us think about combining the two models to exploit both the textual and behavioural side of a review.
After we run the models separately we froze the hidden layers parameters to conserve the knowledge gained in the training part, then we added a common hidden layer which aims to merge and extract combined knowledge.This layer is linked to the prediction layer as shown in the figure below.

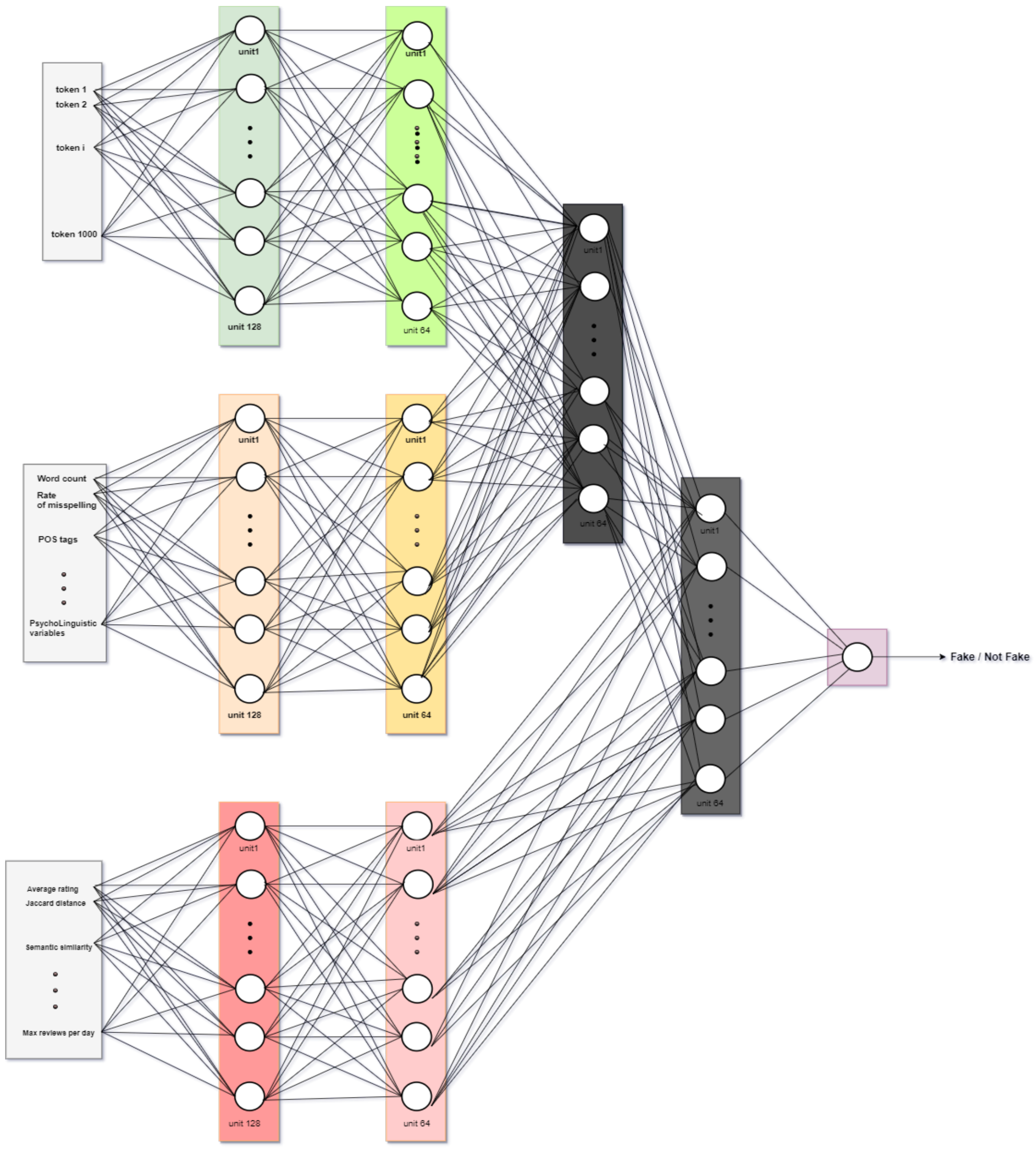


Figure 5.8: Overview of Combined Model

We can notice that the combined model has the best performance in terms of precision (81.13%), recall (87.4%) and F1 score(79.27%).

Results and analysis
The combined model achieved much better results compared to the previous approaches. The overall gain proves a strong complementarity between textual and behavioral features in fake reviews detection. It is to be observed that our combined approach presented a high recall score (87%) confirming the model's sensitivity in detecting spam feedback which is in our case valid and efficient. Without regard to f1 score (79%) which was a bit lower comparing results obtained for some research papers like what yelp might be doing] and [Fake Review Detection on Yelp] respectively 84% and 81.42%. This disproportion may be explained by the fact that during our model implementation we didn't consider yelp-specific features like geo-locations, distance between home and restaurant/hotel/shop, number of people that voted the review as cool or funny... In order to make our model as generic as possible. To confirm this hypothesis a new implementation was conducted including the previous model along with the missing features (we achieved f1 score of 85.60%).

5.2.5 Applying the combined model on mobile App reviews dataset

At the beginning of the project, the lab tried to access a mobile app reviews labelled dataset related to the paper "Towards Understanding And Detecting fake reviews in app stores"[11].
We managed to receive the dataset in the last few days of our projects so we wanted to apply our previous approach on the mobile app reviews thus evaluating our work. Some of the behavioural features were not extracted due to the unavailability of all the reviews of each reviewer. The dataset is composed of 16 000 reviews with 50 % / 50 % fake and non fake reviews, Since the dataset is already balanced we directly moved to the models creation, the combined model achieved an f1 score of 97.48 % which proves the validity of our approach.

Chapter 6
Conclusion

The contributions of this work can be summarized as following. We first created a system that categorizes the mobile app reviews and we had the opportunity to work on a large dataset beating other models trained on small datasets. Working with a larger dataset allowed us to implement complex neural networks that scale well with data thus detecting more categories. This will help developers quickly track the most trending user issues better than previous researches.
After developing this model we had the opportunity to externalize it by creating an API ready to be integrated with the ISELAB app.
Secondly, we proposed a combined model to detect fake mobile app reviews. A neural network model that combines review-centric features (statistical and textual) with reviewer-centric features. We first started with applying the approach on Yelp dataset, the behaviour of yelp reviewers is much more complex than mobile app reviewers since they post hundreds of long reviews and each one of them can have simultaneously fake and legit reviews. This complexity led us to extract more features ending with over a hundred. Moving to mobile app reviews that we obtained late in our research, the previously extracted features were highly sufficient to end up with more than 97% f1-score. These results proved the validity of our approach which can be integrated later in the ISELAB application. The development of these two models will permit developers to have a clear insight on their clients' needs and to spot spammers' attacks.
On the other hand, fake reviews filtering and reviews categorization as app store features would be very useful for potential app users since they would be able to quickly filter a research in the review section if they faced any issue and check if someone else had a similar problem.

Threats :
For the potential threats of our work we can list :
In the first model, the dataset was clustered before assigning labels to each category, this approach while being fast and efficient, has made some mistakes in labeling some of the reviews, to mitigate this threat we have labeled manually some reviews and generated some artificial reviews to boost the number of the reviews for some categories.
In the second model, the mobile app dataset was not sufficiently big which would pose a threat to external validity and generalisation of the results of our work. However, the results were near perfect and having slightly lower scores with a bigger dataset won't hinder our approach.

Bibliography

[1] Powers, David M W (2011). "Evaluation: From Precision, Recall and F-Measure to ROC, Informedness, Markedness Correlation". Journal of Machine Learning Technologies

[2] Chen N, Lin J, Hoi SCH, Xiao X, Zhang B (2014) Ar-miner: mining informative reviews for developers from mobile app marketplace. In: Proceedings of the 36th international conference on software engineering. ICSE 2014. ACM, New York, pp 767–778.

[3] E. Guzman, M. El-Halaby, and B. Bruegge, “Ensemble Methods for App Review Classification: An Approach for Software Evolution.” in Proc. of the Automated Software Enginering Conference (ASE), 2015, pp. 771–776.

[4] Ciurumelea, A., Schaufelbuhl, A., Panichella, S., and Gall, H. C. 2017. Analyzing reviews and code of mobile apps for better release planning. In Software Analysis, Evolution and Reengineering (SANER), 2017 IEEE 24th International Conference on. IEEE, 91–102.

[5] McIlroy, S., Ali, N., Khalid, H., and E. Hassan, A. 2016.Analyzing and automatically labelling the types of user issues that are raised in mobile app reviews. Empirical Software Engineering 21, 3, 1067–1106.

[6] Ryan Rifkin and Aldebaro Klautau. Parallel networks that learn to pronounce english text. Journal of Machine Learning Research, pages 101–141, 2004.

[7] Aly, M. Survey on multi-class classification methods ,2005.

[8] Rajaraman, A., Ullman, J. (2011). Data Mining. In Mining of Massive Datasets (pp. 7-8). Cambridge: Cambridge University Press.

[9] Ee-Peng Lim, Viet-An Nguyen, Nitin Jindal, Bing Liu, and Hady Wirawan Lauw. Detecting product review spammers using rating behaviors. In Proceeeding of the 19th ACM International Conference on Information and Knowledge Management, CIKM'10, pages 939-948, Toronto, ON, 2010.

[10] Arjun Mukherjee, Vivek Venkataraman, Bing Liu, Natalie Glance, “What Yelp Fake Review Filter Might Be Doing,” ICWSM (2013), February 2017.

[11] Daniel Martens1, Walid Maalej.Towards Understanding And Detecting fake reviews in app stores.Empirical Software Engineering (2019) 24:3316–3355.