
# Identifying actionable app reviews

**TDDE16 Text Mining - Project Report**

Wilhelm Hansson

Autumn 2020

# Abstract

In this project, three methods for classifying reviews as relevant or not based on the review content was devised and tested. One method used TF-IDF vectorization and Nearest Neighbor for classification. The second and third method both utilized different configurations of a Multi-layer Perceptron classifier. Each method was evaluated using precision, recall and F1 score. The results were then compared to those of a "random guess" baseline. Compared to the baseline, the second and third method both improved the F1 score, but the first method did not. The data set used for analysis contained information about "apps" from the *Shopify apps marketplace* and was provided by the data service Kaggle. A Gold standard with 499 entries was created from randomly sampling the original data set. This Gold standard was used for both training and testing using a 80/20 split.

# Contents

# 1 Introduction

Shopify is an eCommerce Platform allowing for anyone to get started with eCommerce by providing the services required to run an online store. Shopify has also created a *Shopify apps marketplace* which allow their customers to customize and enhance their online store by installing "apps" created by third-party developers. The *Shopify apps marketplace* can be compared to the app stores of Apple and Google. Many of the concepts from these app stores have also been brought over to the *Shopify apps marketplace*. People who have created Shopify online stores and are customizing them using apps can rate an app and also leave written feedback in the form of a *review*. Developers are also able to state the *Key benefits* of their app. *Key benefits* describe the core purpose of the app and can be perceived as a promise from the developers to their user base. *Key benefits* can be both functional and non functional, a non functional benefit could be *24/7 customer support*.

Pagano and Maalej conducted an empirical study where they explored the impact app stores, mainly Apple AppStore, have had on the requirements engineering processes [6]. The study focused on the aspect of user written feedback; how and when feedback is submitted, the content of feedback and its impact on the requirements engineering processes. It is concluded that user reviews can contain useful content such as feature requests, bug reports and user experience, but there is also a large amount of superficial and non-specific feedback [6]. The authors also conclude that developers would benefit from tools to filter, aggregate and classify the reviews, a functionality these app stores do not provide.

In comparison to the app stores mentioned by Pagano and Maalej [6] the *Shopify apps marketplace* does provide a possibility for developers to answer user comments and since the publishing of Pagano and Maalej study, changes have also been made to the mentioned app stores. However, even if it exists a possibility to answer individual user reviews there is still a problem for developers to get an overview of the written feedback and to identify reviews of relevance to requirements engineering and service development. A potential solution to this problem would be to apply methods used within text mining and information retrieval.

The data-science service Kaggle provides a data set named **Shopify app store**[1] containing "Web scraped data of apps and reviews (from Shopify apps marketplace)". This data set will be used to answer the research questions and the set is described in more detail in section 3.

The purpose of this project is to identify how well certain methods within text mining can solve the task of identifying *relevant* reviews on apps in Shopify apps marketplace. Where *relevant* refers to reviews that provide feedback regarding an apps *Key benefits*. The following research questions will be answered.

- **Question 1 (RQ1)**: Can a method utilizing TD-IDF-vectorization and Nearest Neighbours achieve a higher F1 score than a random guess on the task of identifying *relevant* reviews by comparing reviews to the stated Key benefits?
- **Question 2 (RQ2)**: Can a method utilizing MLP and focusing on the structural patterns in the review achieve a higher F1 score than a random guess on the task of identifying *relevant* reviews?
- **Question 3 (RQ3)**: Can a method utilizing MLP and comparing reviews to Key benefits achieve a higher F1 score than a random guess on the task of identifying *relevant* reviews?

[1] https://www.kaggle.com/usernam3/shopify-app-store *Downloaded: 2021-01-06*

# 2 Theory

This section starts by explaining the metrics used for evaluating the classification task. Then a short explanation about two concepts highly relevant is presented and lastly related work in the area is summarized.

## 2.1 Precision, Recall and F1 score

During evaluation of a binary classification task a predicted label can fall into one of the categories illustrated by Figure 1 based on the actual label of the item. This table is called a **confusion matrix** and can be used to calculate evaluation metrics. In this project the metrics **precision**, **recall** and **F1 score** will be used, equations 1, 2 and 3 shows how the metrics are calculated based on the confusion matrix

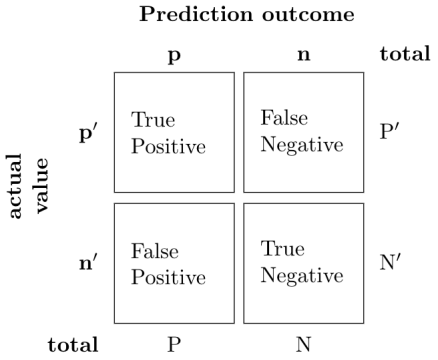


Figure 1: A confusion matrix for binary classification problems.

$$Precision = \frac{TruePositive(TP)}{TruePositive(TP) + FalsePositive(FP)} \tag{1}$$

$$Recall = \frac{TruePositive(TP)}{TruePositive(TP) + FalseNegative(FN)} \tag{2}$$

$$\text{F1 score} = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{3}$$

## 2.2 TF-IDF vector

C. Zhai and S. Massung have written a book described as a *practical introduction to information retrieval and text mining* [8]. Among other things C. Zhai and S. Massung explain the concept of representing a text document as a vector. The TF-IDF vector is one such representation and will be used for this project. The **Term Frequency (TF)** is a feature describing how many times a specific word occurs within a given document. Another feature is **Document Frequency (DF)** which describes the number of documents that a given word occurs in. Both of these are statistical measurements trying to say something about a given word and its relation to the document collection. There is an extension to the DF measure, namely the **Inverse Document Frequency (IDF)** (Equation 4) which tries to penalize words that occur more often throughout the whole document collection. TF and IDF can be combined to form the TF-IDF weight for a given word in a document being part of a collection, this is illustrated by Equation 5. To generate a TF-IDF vector representation of a given document one most calculate the TF-IDF weight for each word in the document.

$$IDF(word) = log(\frac{M+1}{DF(word)}), \text{ where } M \text{ is the total number of documents in the collection.} \quad (4)$$

$$\text{TF-IDF(word,document)} = TF(word, document) * log(\frac{M+1}{DF(word)}) \quad (5)$$

### 2.3 Multi-layer Perceptron (MLP)

Multi-layer Perceptron (MLP) is a type of Artificial Neural Network consisting of an acyclic graph. It consists of an input layer, an arbitrary number of hidden layers and one output layer. The network is a feed-forward network meaning that the output of each layer acts as input for the next layer and this makes it relatively simple compared to other neural networks. During training, data is given to the input layer and fed forward, at the output node the error is calculated and propagated backwards (back-propagation algorithm) to adjust the weights and biases. Each layer has an activation function which is adjusted during training. Common activation functions are *ReLU* (Equation 6) and *Sigmoid* (Equation 7).

$$f(x) = max(0, x) \quad (6)$$

$$f(x) = \frac{1}{(1+e^{-x})} \quad (7)$$

### 2.4 Related work

T. Johann et al. presents and tests a method for extraction of information about app functionality mentioned in human-generated text [4]. The source of text was descriptions of apps and reviews about the same apps from both Google Play and Apple App Store. The goal of the study was to identify features mentioned in the description and the reviews and then trying to match these features. The authors identified 18 different Part-of-speech patterns which were often used when describing a feature, both in-app descriptions and reviews [4]. Furthermore, some sentence patterns were identified, which were used in conjunction with the POS patterns to identify app features. The authors suggest future work regarding improving the results by adding subsequent machine learning steps. This paper was published in 2017.

Using neural networks to classify app reviews is the subject of a paper by N. Aslam et al. [1]. The classifier is a Convolutional Neural Network (CNN) that tries to put each review in one of four classes and uses a combination of extracted textual features and non-textual features to achieve this. Textual features revolve around word embeddings. However, before generating the vector representation of words, several pre-processing steps are performed. Among other things is spell checking performed, there is also removal of stop words. The authors conclude that the proposed approach show potential to significantly surpass the existing state of the art approaches. This paper was published in 2020.

A paper by F. Rustam et al. [7] uses the same data set as will be used in this project. The main goal of F. Rustam et al. is to use different feature engineering techniques singly and combined to solve the task of classifying a review as *happy* or *unhappy*. Among the techniques used are TF-IDF and Logistic regression. The results of each pipeline are evaluated using precision, recall, and F1 score. The authors conclude by a comparative analysis that among the techniques used, a pipeline based on feature engineering using a combination of TF-IDF and Chi2 and then classifying by Logistic regression yields the best result.

# 3 Data set

For this project the data set **Shopify app store**[2] provided by the service Kaggle was used. The set contains data about 4750 apps from the Shopify marketplace and is composed of several subsets as described in Table 1. For this project, only a subset containing data for 5 apps was used and the selection of these 5 apps is described in further detail in subsection 3.1.

| **Subsets** | **Description** |
|---|---|
| Apps | Details of the apps on Shopify apps marketplace |
| categories | Categories of the apps. Each app has multiple categories |
| key benefits | The main "key" benefits which are highlighted by the developer. |
| reviews | Each review contains a users opinion about a related app. |
| price plan | Each app has multiple pricing plans, each plan has multiple pricing plan features. |
| price plan features | Features included in pricing plans and displayed on the app page |

Table 1: The different subsets available in the data set.

## 3.1 Processing of dataset

To answer the stated research questions there is a need to know whether a given review is *relevant* (discusses the Key benefits of the given app) or not. Since the definition of relevance was stated within the scope of this report, this feature is not available in the data set. To be able to evaluate the effectiveness of the methods and thereby answering the research questions the relevance feature needed to be created and added. Since the data set contains *441 167* reviews it would not be possible to generate this feature for every review. To tackle this problem a subset of the apps were selected and all corresponding reviews were collected. In Figure 2 the whole selection and filtering process is illustrated. The selection and labelling will be discussed further in section 4.

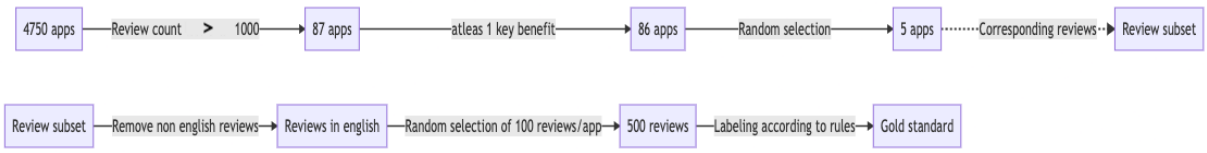


Figure 2: Selection process to generate Gold standard data set

Table 2 presents the composition of the labelled data set, also referred to as the *Gold standard*. Note that the intention was to have 100 reviews for each app, but one non-English review slipped through the filtering process and needed to be discarded. In Table 3 one can read about the variables in the Gold Standard data set. These variables all come from the original data set and to the review set *mentions_kb* was manually generated and added. This is the *Gold standard label* and Figure 3 shows the number of samples from each class (True/False) that is present in the set.

| **Subset** | **# items** | **Comment** |
|---|---|---|
| Apps | 5 | Number of apps used |
| Reviews | 499 | 100 for each app, one app with 99 |
| Key benefits | 15 | 3 for each app |

Table 2: The content of the Gold standard data set.

[2]https://www.kaggle.com/usernam3/shopify-app-store *Downloaded: 2021-01-06*

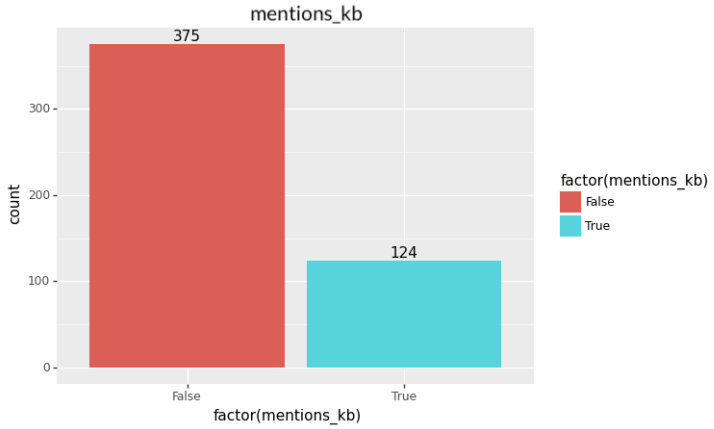


Figure 3: The number of reviews of each class in the labelled data set (Gold standard).

| **Subsets** | **Variable** | **Description** |
|---|---|---|
| Apps | id | Id of the app in this dataset (not the Shopify id) |
| | title | Title of app |
| | reviews_count | Number of the reviews from app page |
| key benefits | app_id | Application id, corresponds to *id* above |
| | title | Title of the app benefit (feature) |
| | description | Description of the app benefit (feature) |
| reviews | app_id | Application id, corresponds to *id* above |
| | body | The body (text) of the review |
| | mentions_kb | Manually added Gold label for 499 reviews |
| | | *Refer to 3.1 Preprocessing of dataset.* |

Table 3: Details about variables kept from the original data set

Table 4 contains information about the apps ending up in the Gold standard and Table 5 describes the Key benefits for each app. This table shows only an expert of the feature *description* and the full length can be found in the original data set. The column *id* in Table 4 shows the ids that each app was given by the creator of the original data set.

| id | title | reviews_count | # of reviews selected |
|---|---|---|---|
| 94e77128-... | EU Cookie Bar GDPR - 100% FREE | 3279 | 100 |
| 4ee3c37d-... | Swell Loyalty & Rewards | 2133 | 100 |
| d4d7e4ca-... | Omnisend Email Marketing | 3697 | 99 (!) |
| 2264513f-... | MailChimp Forms by MailMunch | 1289 | 100 |
| c769a28f-... | Printful: Print-on-Demand | 3025 | 100 |

Table 4: The 5 apps choosen for project.

| app | title | description |
|---|---|---|
| Swell Loya... | Loyalty, Rewards, Referrals | Supercharge your business with a loyalty ... |
| | Increase Lifetime Value | Build a beautiful and immersive loyalty ... |
| | Acquire New Customers | Build a referral marketing program that ... |
| EU Cookie ... | Cookie consent bar for GDPR | Comply with EU law and display a GDPR ... |
| | One-Click Cookie & GDPR bar | One step towards more security for your ... |
| | FREE cookie compliance - GDPR | The app is - and always will be FREE. The ... |
| Omnisend E... | Drive sales on autopilot | Recover abandoned carts, reactivate lost ... |
| | Get started in minutes | Skip the tedious work with our single-click ... |
| | Combine email with SMS | Integrate SMS into your email marketing ... |
| MailChimp ... | Popups, Bars and Banners | Grow your email list with high converting ... |
| | Coupons and Offers | Timely popups let you present your customers ... |
| | MailChimp Emails | Send manual or automated emails, create ... |
| Printful: ... | No Upfront Costs | No setup, minimums, or monthly fees—only ... |
| | Impressive product catalog | Choose from more than 230 products, from ... |
| | 24/7 Customer Support | Our dedicated customer support team always ... |

Table 5: The Key benefits of choosen apps

# 4 Method

In this section, the method for conducting the study will be presented. The first matter discussed is the creation of the Gold standard used throughout the study. Then a short explanation of how the data set can be split in different ways is discussed. Then follow three sections, each describing a separate method related to one of the research questions stated.

## 4.1 Creation of Gold standard

To answer the study's purpose and research questions the generation of a new feature was needed. Generating this feature for each of the *441 167* reviews was not feasible due to time constraints but also since all reviews were not in a language known to the report author. To address this issue a subset of reviews were selected. The selection process is illustrated in Figure 2 and Table 6 describes each step of the process in more detail. The steps were executed sequentially according to their *Step index*. The composition of the Gold standard data set can be found in Tables 2, 4 and 5. The *499* reviews are not presented as a whole but a sample can be found in Appendix C.

| Step | What | Comment |
|---|---|---|
| 1 | Filtering: review count | Apps with less than 1000 reviews were removed |
| 2 | Filtering: Key benefits | Apps most have at least 1 Key benefit (KB) |
| 3 | Selection of apps | 5 apps were selected at random |
| 4 | Collecting reviews | Created subset containing only reviews from selected apps |
| 5 | Filtering: non-english | Reviews not in english were removed |
| 6 | Selection of reviews | Selecting 100 reviews at random from each app = 500 reviews |
| 7 | Gold labeling | Each of the 500 reviews was manually labelled with<br>0 (*not mentioning any KB*) or 1 (*mentioning at least one KB*)<br>*It was at this stage a non-english review*<br>*was found and removed. Resulting in 499 labelled reviews.* |
| END | Gold Standard | The final data set being used for the study was saved. |

Table 6: The process of creating the Gold standard data set.

For clarification some comments are elaborated on, refer to the steps in Table 6:

- **Step 3 & 6**: These samples were drawn using the a random number generator drawing from a uniform distribution.
- **Step 5**: To identify and filter out non-english reviews the package **spacy-langdetect**[3] was used. There were no longer process behind this decision, but rather "gets the job done".
- **Step 7**: To be considered *relevant* a review needed to fulfil all of the following criteria:
  1. It most be clear that the reviewer has used the app.
  2. The review mentions something correlated to the Key benefits.
  3. Feedback in a review should be specific and not something applicable to multiple apps.

## 4.2 Splitting data horizontal and vertical

During training and testing of methods explained in subsection 4.5 and 4.6 the data was split in two different ways yielding two results for each method. Figure 4 shows all information relevant regarding the two types of data splits. Going forward, these terms will be used to refer to the different types of splits. In all cases were a horizontal split was used the **data was split 20%/80% (testing, training)**.

[3]https://pypi.org/project/spacy-langdetect/ *Visited 2021-01-12*

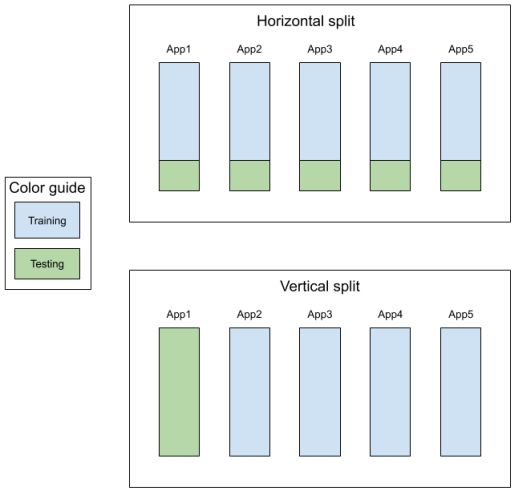


Figure 4: Illustration of how the data set can be split horizontal or vertical.

## 4.3 Baseline

To get a point of reference a baseline was implemented. This was done by using a "dummy classifier" which randomly guessed the label for a given review. The guesses were based on random draws from a uniform distribution

## 4.4 Classification method 1: Document similarity (RQ1)

The main idea using this method is that of a search engine, using the Key benefit documents as the search query and return all review documents that reach a certain level of similarity. The similarity was defined as being within a certain distance in a vector space. The vectors were generated using TD-IDF vector representations of each document and then calculating the min-distance to the vector representations of each Key benefit document using a Nearest Neighbors method. Algorithm 1 shows the procedure in more detail. The variable $THRESHOLD = 1.13$ was decided using grid search with a target of highest F1 score for the **True** label using a subset of the labelled data set.

### Custom tokenizer for tf-idf vectorizer

Pre-processing of each string was performed before it was given to the *TF-IDF vectorizer*. The Natural Language Processing package **spaCy**[4] was used to lemmatize each word in the string and then remove all stop words and non-alpha characters. The result was returned as an array which was then passed to the *TF-IDF vectorizer*. In Algorithm 1 whenever a string is passed to *TfidfVectorizer* this pre-processing is performed.

## 4.5 Classification method 2: Identifying patterns in reviews (RQ2)

This classification method uses a *Multi-layer perceptron classifier* with 300 input nodes. The idea behind this method is that *relevant* reviews contain textual structures and patterns not present otherwise. The strategy is to train an MLP using a subset of the labelled data and see if the MLP can learn these

[4]https://spacy.io *visited 2021-01-12*

**Algorithm 1** Classifying reviews based on TF-IDF and NearestNeighbors

**procedure** CLASSIFY(Apps)
  **for all** Apps **do**
    $cr \leftarrow$ reviews for app
    $kbs \leftarrow$ key benefits for app ▷ title and description
    $vectorizer \leftarrow$ initiate TfidfVectorizer with custom tokenizer.
    $vectorizer.fit(kbs.title + kbs.description)$ ▷ Fit the vectorizer to the key benefits
    $kbv \leftarrow vectorizer.transform(kbs.title + kbs.description)$
    $neighb \leftarrow$ initiate NearestNeighbors
    $neighb.fit(kbv)$ ▷ Fit NearestNeighbors to vectors of key benefits
    $mindist \leftarrow []$ ▷ Assign empty array for storing min distances
    **for all** Reviews of current App **do**
      $review\_vector \leftarrow vectorizer.transform(review.body)$
      $distances \leftarrow neighb.calcdistance$ ▷ Calculate distance from review to key benefits
      $mindist \leftarrow min(distances)$ ▷ Append shortest distance
    **end for**
    $predictions \leftarrow []$
    **for all** mindist **do** ▷ Doing classification
      **if** $mindist < THRESHOLD$ **then** ▷ Either True or False is appended to *predictions*
        $predictions \leftarrow TRUE$
      **else**
        $predictions \leftarrow FALSE$
      **end if**
    **end for**
  **end for**
  **return** *predictions*
**end procedure**

patterns and then identify them in new reviews. This method completely ignores the presence of stated Key benefits based on the findings by T. Johann et al. [4].

The package **spaCy**'s large language model[5] provides a word embedding for each word contained in the model. Each embedding is a vector in 300-dimensional space, stored as an array of length 300.

Figure 5 shows the conceptual structure of the MLP and one can observe that the number of input nodes is 300. Since a review can vary in length, each review was pre-processed before it was fed to the MLP classifier. This pre-processing was a sum of the word embedding vector for each word in the review (Equation 8) and resulted in a new 300-dimensional vector. This pre-processing allowed for reviews of arbitrary length.

$$\sum_{word \subset Review} (word.vectorRepresentation) = InputforMLP \tag{8}$$

[5] https://spacy.io/models/en *Visited 2021-11-12*

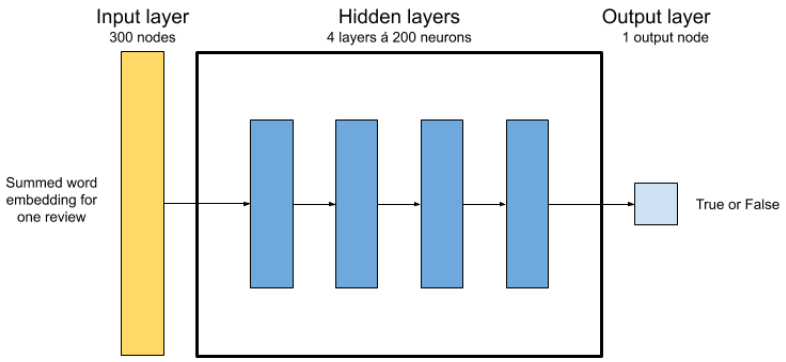


Figure 5: Conceptual structure of the MLP used in classification method 2.

As mention in subsection 4.2, the model was trained and tested two times using different composition of the training and testing set. The pre-processing described above was performed both times.

## 4.6 Classification method 3: Comparing reviews to Key benefits (RQ3)

This classification method uses a *Multi-layer perceptron classifier* with 600 input nodes. The idea behind this method is that of comparison between the stated Key benefits and a given review. On a conceptual level, this method most closely resembles how a human probably would try solving the given task. The MLP is given 600 input nodes, 300 for the summed vector of the Key benefits and 300 for a review. The pre-processing to generate the two input vectors is the same as described in subsection 4.5 using Equation 8, in the case of Key benefits the variable $Reviews$ is changed to $Keybenefits$. Figure 6 illustrates the MLP classifier.

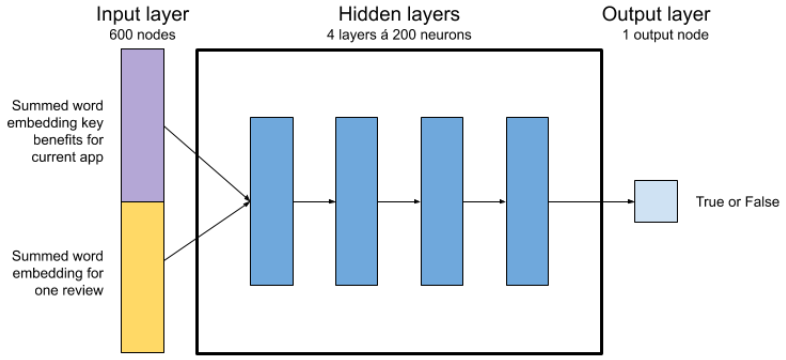


Figure 6: Conceptual structure of the MLP used in classification method 3.

As mention in subsection 4.2, the model was trained and tested two times using different composition of the training and testing set. The pre-processing described above was performed both times.

## 4.7 Calculation of results

A horizontal split 20%/80% (test/training) allow for multiple configurations of the data samples and multiple rounds can be executed. This allows for generating multiple prediction sets. Then the mean of the total of these sets can be calculated. This is of interest because the precision and recall of the full set allow for more general conclusion than just one test run.

The structure of a vertical split does not allow for as many different test/training splits. Instead, one can apply a "leave one out" methodology and use all apps but one for training and then testing on the last app. This can be iterated overall apps and then the precision and recall for all predictions can be calculated.

**Algorithm 2** Testing and calculating results in horizontal split

```
predictions ← []
true_vals ← []
i ← 0
repeat
    test, train ← create_test_train(labelled_data)
    classifier.fit(train)
    predictions ← classifier.predict(test)          ▷ Appending
    true_vals ← test.true_vals                      ▷ Appending
    i ← i + 1
until i >= n                                        ▷ n can be set to an arbitrary value.
precision ← calc_precision(true_vals, predictions)
recall ← calc_recall(true_vals, predictions)
f1 ← calc_f1(true_vals, predictions)
```

**Algorithm 3** Testing and calculating results in vertical split

```
predictions ← []
true_vals ← []
for all Apps do
    classifier.fit(App.key_benefits)
    predictions ← classifier.predict(App.reviews)   ▷ Appending
    true_vals ← App.true_vals                       ▷ Appending
end for
precision ← calc_precision(true_vals, predictions)
recall ← calc_recall(true_vals, predictions)
f1 ← calc_f1(true_vals, predictions)
```

# 5 Results

Examining figures 7 and 8 on can observe that the False-label generally has higher scores than the True-label. It is further observed that in all cases but one the recall task for the True-label had a lower precentage than the precision. The F1 score in Figure 9 follows the same pattern. Table 7 summarizes how the F1 score for each method changed compared to the Baseline. Method 2 and 3 has a larger change in F1 score than method 1. The change in F1 score for the two classes is roughly the same in all cases, **M2:h-split:True** = 0.23 and **M2:h-split:False** = 0.26 to give one example. However, **M3:v-split:True** = 0.15 is smaller.

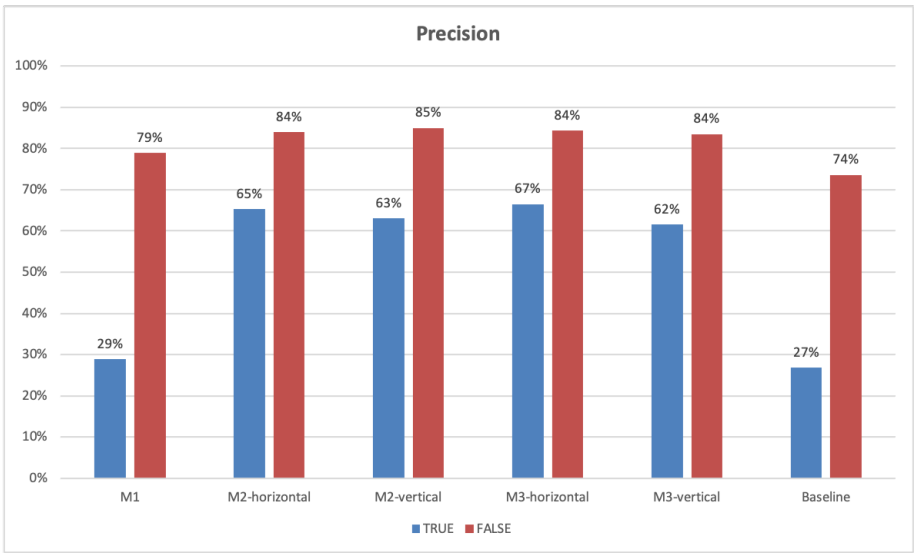


Figure 7: Precision for each method, Baseline for comparison

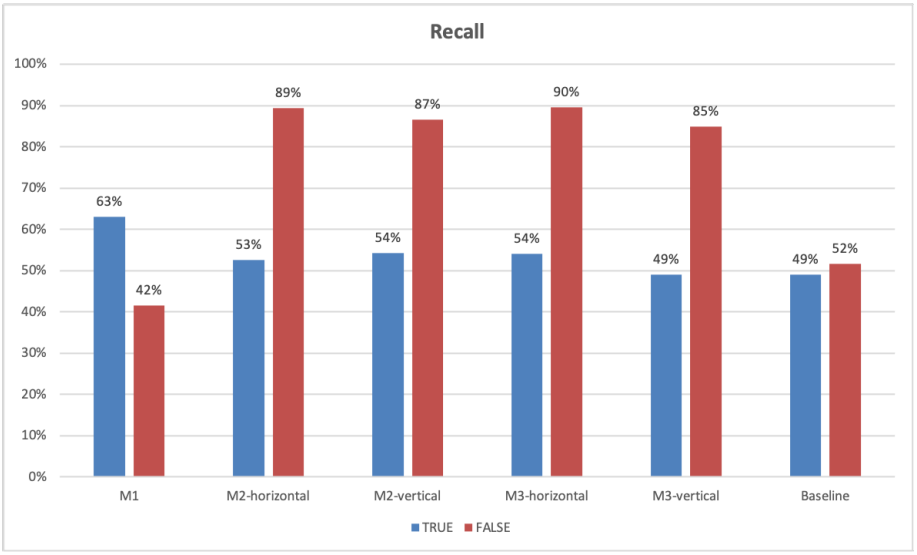


Figure 8: Recall for each method, Baseline for comparison

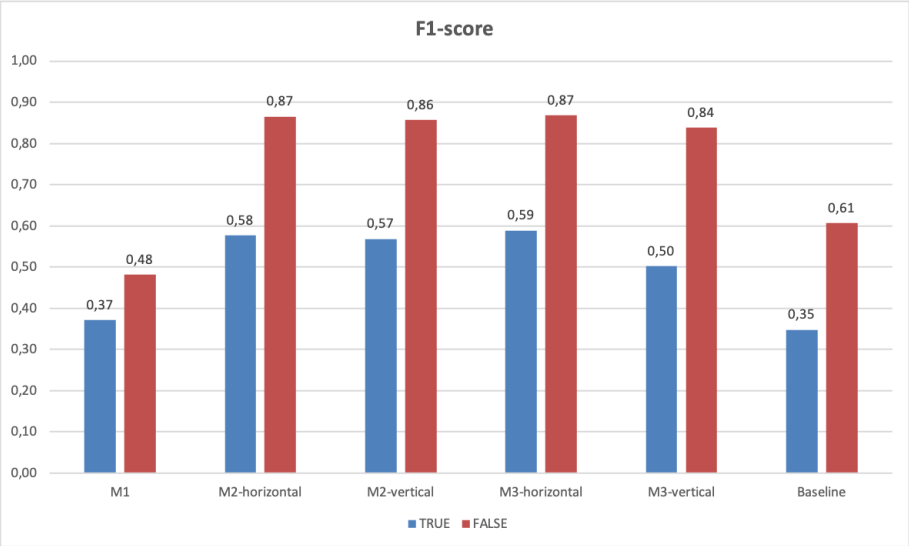


Figure 9: F1 score for each method, Baseline for comparison

| | | M1 | M2 | M3 |
|---|---|---|---|---|
| True | h-split | X | 0.23 | 0.24 |
| | v-split | 0.02 | 0.22 | 0.15 |
| False | h-split | X | 0.26 | 0.26 |
| | v-split | -0.13 | 0.25 | 0.23 |

Table 7: Change in F1-score when comparing to the Baseline.

*For tabular form of the results in Figures 7, 8 and 9, see Appendix B.*

# 6 Discussion

Both precision and recall are of equal interest when filtering a collection of reviews and one can argue that it is of more interest to get a high metric value for the True-labels. The reasoning behind this is that the team behind an app is interested to get as much relevant feedback as possible without having to wade through general and non-actionable reviews. The F1 score was calculated because it contains the information of both the precision and recall (Equation 3).

Compared to the baseline, method 1 does not seem to affect the outcome of the F1 score. Regarding recall method 1 presents a higher percentage than the baseline for the True-label. Method 2 outperforms the baseline on all metrics in both the case of a horizontal and vertical split. Method 3 outperform the baseline in almost all cases, but not in the case of recall for the True-label in the vertical split. Comparing the horizontal split there is no increase in recall for the True-label but a higher precision is observed. Considering the research questions and examining the change in F1 score for each method compared to the baseline, one can conclude that the current implementation of method 1 (RQ1) is not able to perform better than a random guess when solving the classification task. However, all cases of both method 2 (RQ2) and 3 (RQ3) show results that indicate success in solving the task better than the baseline.

Splitting data horizontal or vertical does only seem to have an observable effect on method 3, where the performance for predicting the True-label is worse in the vertical case. One possibility for this is that during evaluation after a vertical split the MLP is fed not only previously unseen reviews but also previously unseen Key benefits. This is an indication that method 3 is bad at generalizing when presented with a new app. Not seeing the same difference in method 2 is probably a consequence of not including the Key benefits in the prediction process for method 2.

Regarding study scope and evaluation work by N. Aslam et al. [1] and T. Johann et al. [4] exhibit some similarities with this project. N. Aslam et al. uses a CNN to solve a multi-class classification problem, where three of the classes are *Bug reports*, *Enhancement reports*, and *User experiences*. When comparing reviews to Key benefits, the content of the types classified by N. Aslam et al. is relevant. N. Aslam et al. approach yielded a precision of 95.49% and a recall of 93.94% resulting in an F1 score of 0,9448 (using Equation 3) when averaging over all classes. As part of T. Johann et al. results, they present precision and recall for the task of **correctly identify matches** (true positives) found in descriptions with those found in reviews for the corresponding app. Their approach achieved a precision of 70.4% and a recall of 56.0% resulting in a F1 score of 0,624 (using Equation 3) which could be compared to results in Figure 9.

One reason that method 1 perform so bad could be that the pre-processing step in the pipeline does not prepare the data well enough. In previous work [1, 3, 4] by others the pre-processing contains multiple steps and N. Aslam et al. [1] perform spell checking and tries to correct words as the first step in their pre-processing of user reviews. Method 1 in this project could potentially be improved by implementing some of these steps. The most obvious would be to also account for the semantic relationship between words in a given review and the corresponding Key benefits. An article by Y. Li et al. [5] focuses on finding sentence similarity based on semantics and corpus statistics and their work is referenced by T. Johann et al. [4]. The work by Y. Li et al. could be used for feature engineering when trying to improve method 1. Another approach would be to use Logistic regression for the classification rather than Nearest neighbours with a threshold, this suggestion is based on the previously mentioned work by F. Rustam et al. [7]. In hindsight, Logistic regression seems to better fit the task at hand.

Examining the results of method 2, where the Key benefits were left out, they seem to support the idea that reviews presenting more actionable or specific feedback share a structural pattern. This idea came from the previously presented work by T. Johann et al. [4]. One aspect to consider further is how the data is processed before given to the MLP's input layer. This consideration applies to both method 2 and 3, where the current implementation is a summed vector of word embeddings. Pre-processing will always be needed to fit the input to a fixed length. However, the actual formula for combining word embeddings could be changed and one could also consider adding more steps to the pre-processing. Inspired by previously mentioned work [1], one step to add could be the removal of stop words.

The ratio of relevant and non-relevant reviews in the Gold standard data set (Figure 3) is imbalanced. A. Fernández et al. [2] writes about working with imbalanced data sets and techniques to handle the imbalance. One such technique is creating a balanced subset. In this project, the Gold standard makes up a small subset of the original data set and the imbalance could have been lower or possibly avoided. An imbalanced data set can result in the algorithm under-performing on the minority class [2] and one can see this effect being present in the project results, the F1 score for the majority class (False) is higher in all cases. However, the fact that the data set was imbalanced was partly taken into account by favouring higher scores for the minority class during tuning of the algorithms.

# 7 Conclusion

The study results indicate that an MLP classifier could potentially be utilized to identify relevant app reviews. The reviews need to be pre-processed before presented to the MLP input layer. A method using TF-IDF vector representations of reviews and classified using the Nearest Neighbour algorithm was also tested. However, this method had the same success rate as randomly guessing whether a review was relevant or not.

The generation of a Gold standard allows for a high level of tailoring regarding project scope and research question but the manual process of labelling samples is a highly time-consuming task and introduces the risk of biases or other factors of error. However, the task of creating a labelled data set makes for an interesting task and one is forced to really reflect upon the process in order to identify potential missteps or actions that could give a biased result. One such thing is that the labelling should take place before any experiments are performed, which is also how it was done during this project.

Future work could investigate refinement of the pre-processing steps, for example trying to introduce "understanding" of the semantic relationship between words and generate features based on these. This is mostly relevant for the TF-IDF with Nearest Neighbour classification.

# References


[1] N. Aslam et al. "Convolutional Neural Network Based Classification of App Reviews". In: *IEEE Access* 8 (2020), pp. 185619–185628. DOI: 10.1109/ACCESS.2020.3029634.

[2] Alberto Fernández et al. *Learning from Imbalanced Data Sets*. Oct. 2018. ISBN: 9783319980744.

[3] E. Guzman and W. Maalej. "How Do Users Like This Feature? A Fine Grained Sentiment Analysis of App Reviews". In: *2014 IEEE 22nd International Requirements Engineering Conference (RE)*. 2014, pp. 153–162. DOI: 10.1109/RE.2014.6912257.

[4] T. Johann et al. "SAFE: A Simple Approach for Feature Extraction from App Descriptions and App Reviews". In: *2017 IEEE 25th International Requirements Engineering Conference (RE)*. 2017, pp. 21–30. DOI: 10.1109/RE.2017.71.

[5] Y. Li et al. "Sentence similarity based on semantic nets and corpus statistics". In: *IEEE Transactions on Knowledge and Data Engineering* 18.8 (2006), pp. 1138–1150. DOI: 10.1109/TKDE.2006.130.

[6] D. Pagano and W. Maalej. "User feedback in the appstore: An empirical study". In: *2013 21st IEEE International Requirements Engineering Conference (RE)*. 2013, pp. 125–134. DOI: 10.1109/RE.2013.6636712.

[7] F. Rustam et al. "Classification of Shopify App User Reviews Using Novel Multi Text Features". In: *IEEE Access* 8 (2020), pp. 30234–30244. DOI: 10.1109/ACCESS.2020.2972632.

[8] ChengXiang Zhai and Sean Massung. *Text Data Management and Analysis*. June 2016. ISBN: 9781970001174.

# Appendix

## A Project code

Visit the link below to access the Jupyter notebook containing the project code.

https://github.com/wiiha/tdde16-project-report

# B Precision, Recall and F1 score for each method.

This study resulted in five different cases being tested, the results for each case can be found in Tables 9, 10, 11, 12 and 13. In Table 8 the results of a dummy classifier is presented. As mentioned in section 4 this classifier "predicts" labels at random based on draws from a uniform distribution. The mean scores presented in Tables 8, 10 and 12 were calculated from 100 rounds of predictions using Algorithm 2. The mean scores presented in Tables 9, 11 and 13 were calculated from the individual predictions for each app using Algorithm 3.

| Metric | (100 rounds) | True | False |
|---|---|---|---|
| precision | mean | 0.269 | 0.736 |
| recall | mean | 0.49 | 0.516 |
| f1-score | mean | 0.348 | 0.607 |

Table 8: Results of random baseline

| Metric | (5 apps) | True | False |
|---|---|---|---|
| precision | mean | 0.289 | 0.789 |
| recall | mean | 0.631 | 0.416 |
| f1-score | mean | 0.371 | 0.481 |

Table 9: Results of method 1

| Metric | (100 rounds) | True | False |
|---|---|---|---|
| precision | mean | 0.653 | 0.839 |
| recall | mean | 0.526 | 0.894 |
| f1-score | mean | 0.577 | 0.865 |

Table 10: Method 2, with horizontal split.

| Metric | (5 apps) | True | False |
|---|---|---|---|
| precision | mean | 0.631 | 0.85 |
| recall | mean | 0.542 | 0.865 |
| f1-score | mean | 0.568 | 0.857 |

Table 11: Method 2, vertical split.

| Metric | (100 rounds) | True | False |
|---|---|---|---|
| precision | mean | 0.665 | 0.844 |
| recall | mean | 0.54 | 0.896 |
| f1-score | mean | 0.589 | 0.868 |

Table 12: Method 3, horizontal split.

| Metric | (5 apps) | True | False |
|---|---|---|---|
| precision | mean | 0.616 | 0.835 |
| recall | mean | 0.49 | 0.849 |
| f1-score | mean | 0.502 | 0.838 |

Table 13: Method 3, vertical split.

# C Sample of reviews in Gold standard data set

| app_id | body | mentions_kb |
|---|---|---|
| 4ee3c37d-... | Very easy to setup and customize. Looking forward to utilizing it as traffic ramps up. Great app! | True |
| d4d7e4ca-... | The email editor is great and has enough power to make really nice looking and engaging emails. The analytics are good and the interface is easy to navigate, The little bit of customer support I have needed has been very responsive and helpful. | True |
| 94e77128-... | I will install it and will see | False |
| c769a28f-... | Great App, i love it very much. I recommend it to you! Iam selling lots of T-Shirts, its great! VEry easy to use | False |
| 2264513f-... | I am not sure why they haven't gotten back to me after opening multiple support tickets. The software seems great, but the popup form isn't displayed on my site, even when I try via Incognito browser. I will update the review if I hear back from them!<br>They replied very quickly. Form works well :) | True |

Table 14: A sample of labelled reviews, part of the Gold standard