A Comparison of Classifiers on Mobile Applications

Nealan Vettivelu 311201105
Supraja Sridharan 460474472

Abstract—With the growing number of mobile applications available datasets containing information about specific applications are growing exponentially. A classification algorithm is needed in order to attempt and group similar applications together for comparison. This paper compares several preprocessing techniques and classifiers on a provided dataset and evaluates the effects of both preprocessing and row reduction have.

1 Introduction

The invention of smart-phone can be considered as one of the greatest invention of this century. This invention helped the mobile market to grow at a breakneck speed with continual increased usage of smart-phone users. The mobile applications played a vital role for the people to use smart phone. As per 2017 statistics, the Apple application store possesses roughly about 2.2 million applications while Google play has close to 2.8 million application [1]. The worldwide revenue of mobile application in the year 2016 was 88.3bn USD and it was estimated to increase to 188.9bn USD by 2020 [2]. As the number of mobile applications are developing at an enormous rate, it is difficult for the application store providers to categorize the mobile applications at a rapid phase. If applications were wrongly categorized it will be difficult for a user or developer to find an application as they were unsure in which category they have to choose for the application .For instance, there was a report on Sun Daily about a case on Health application where more than 50% of applications were categorized wrongly leading to fewer downloads [3].Many mobile application users feel that existing categorizations are ineffective because applications are assigned to some categories arbitrarily by the developer [4]. This had lead for a demand of an automatic mobile application categorization, which can be achieved through machine learning. Machine learning as a set of methods that can auto detect patterns in data, and then use it for the uncovered patterns to predict future data, or to perform other kinds of decision making under uncertainty [5].Deciding a category to upload an application can be challenging for developers, machine learning can be used to suggest suitable categories for developers based on the details they provide.

By employing machine learning techniques for the classification of newly developed applications, there will be a tremendous improvement in the reduction of time taken to perform this activity, which also improves the effectiveness of the categorization. An organized category maximizes application discoverability and hence there is chances for obtaining more profit by developers. Developers could also study feature patterns among similar application for improvement and maintenance [4]. The current study will provide a new solution to improve the existing processes.

In this study, a large collection of data set containing the mobile application name, description, labels and term frequency inverse document frequency (TF-IDF) values has been provided. The objective of this study is to build a classifier model using training data, execute it for test data and finally matching it to predicted data which determines how accurate classification model has been built. We have used Naive Bayes/KNN classifier model. [3].

2 Approach

In attempting to classify data, we must first try and understand the problem. To do this, we will be using a systematic approach presented by Brownlee [6]. The first step is defining the problem as a machine learning problem. Problem definition framework is a simple framework which helps in defining a new problem which can be solved using machine learning. To answer this question, we believe that in order to classify data correctly, we must look towards similarities between corpus to make assumptions about the likelihood of data belonging to the same category.

The second step defined by Brownlee is how to prepare the data. It is critical to feed the right data to solve machine learning problems.In spite of the fact that we posses good data it is necessary to ensure the format and verify the features are meaningful.

The next step suggested by Brownlee is the use of spot checking algorithms [7]. Spot checking is the use of algorithms to see what algorithms work best on the dataset provided. We can improve our results using algorithm tuning which looks at combining the results of multiple methods in order to get improved results.

Finally the last step is to understand the results from the model in order to be used for further predictions.

2.1 Data set description

There are three input files given to build the classifier. They are training data, training label and training description. Each files consist of 20104 rows. In the training data.csv file, the first attribute is the mobile application name and remaining columns consist of TF-IDF values. Next file is

training_desc.csv which has first attribute as mobile application name and application description, finally the training label file contains the mobile application name and definite labels.

The fourth file is test data which is subset of original data. Basing on this we are supposed to generate data in predicted label. Predicted Label has mobile application name and application labels respectively. The structure of training data and test are similar.

3 METHODS

3.1 Pre-processing

Working with high dimensional spaces is not only computationally expensive but can also lead to degradation of accuracy due to excessive noise created by other dimensions. In order to reduce the dimensions present techniques such as removing largely sparse features, term frequency variance and singular value decompositions can be used to compress the dataset whilst still maintaining much of the information present.

3.1.1 Counting non-Zero Rows

By a quick observation of the dataset we can see that it is primarily 0 values. Upon counting the number of 0's we are able to see that 99.5% of the data is 0 values. The average number of 0's for a single feature was 20,006.By removing rows which had more 0 values than this average, accuracy took a small decrease, but the computation time was reduced over 90% (Table 1). This idea can also be used without removing the features (and thus restoring the same accuracy) by converting from a standard array to a sparse matrix. [8]

3.1.2 Singular Value Decomposition(SVD)

A Singular value decomposition is the decomposition of a large matrix into 3 smaller ones, which when multiplied together again will recreate the original matrix. It's decomposition can be represented as the following :

$$X = U\Sigma W^T$$

with U being an orthogonal matrix, W also a orthogonal matrix and Σ which is a diagonal matrix which represents the 'importance' of each feature value as an eigenvalue.

However, by reducing the matrix to three smaller matrices we are able to manage smaller data sets. This does come at a cost, as computing the SVD does take significant time. Furthermore, [9] has shown that an SVD may not be in its most optimum form if there has not been adequate preprocessing done to the data. As the data has been preprocessed for us, it is not possible to know if there may have been any intricacies that may have allowed for a stronger SVD candidate to be created(such as the use of a collocations or bi-grams instead of uni-grams). However, it is also important to note that the use of an SVD does allow for much more sensitivity towards the data due to its ability to be reconstructed at a later point.

Due to the uncertainty around feature selection for a classifier coupled with the long compute times, we felt that an SVD would not be a suitable approach.

3.1.3 Term Frequency Variance(TfV)

Term frequency variance looks at the variance of each words term frequency. The lower the variance, the less likely that the word can be used as part of a classifier and as such should be removed. It is given by the equation

$$\sum_j^n tf_j^2 - \frac{1}{n}[\sum_j^n tf_j]^2$$

where j is the document in question, and tf is the term frequency of the word in that respective document. Tang et. al. have shown that using a TfV is much more effective than using both the raw document frequency or mean term frequency. [10]

Variances of term frequencies ranged from 0 to 224, by removing any variances less than 1, our new reduced matrix reduced in size from 13626 to 4552 number of words, equivalent to a 66% reduction in size. Furthermore in our development stage results, we found that this new reduced matrix actually yielded a higher total correct classification rate (an increase of 1.5%) suggesting it is not only a smaller model, but a more accurate one as well1.

3.1.4 Information Gain

Information gain looks at the entropy or distribution within a given feature to see how much information is gained from a feature. It can be thought of as the more entropy required for a feature, the more uncertain each data point within the feature set is and thus more data is required to store the feature. In order to choose the best feature set, we should aim to choose the data with the highest entropy, as it will provide the most information per feature set. Information gain is given by the formula

$$I(T, X) = H(T) - H(T, X)$$

where entropy is denoted as

$$H(X) = \sum_{i=1}^{M} p_i log_2(\frac{1}{p_i})$$

Duan [11] has shown that a combination of both Information gain theory as well as a Naive Bayes model results in a higher overall accuracy due to the fundamental cornerstone of the Naive Bayes model which is for each feature set to be as linearly of each other as possible to remove noise.

3.2 Classifier

A classifier acts as a translator applying probabilistic models to a data set in order to determine the most likely class that a set of features belongs to. Each classifier will work best in certain types of data sets and all of them have both pro's and con's depending on the type of work being done.

3.2.1 K Nearest Neighbour

Our assignment uses TF-IDF for preprocessing of application description. TF-IDF which is used for numerical statistics for reflecting the importance of a word in a document or in a collection of a document. Today, TF-IDF is used mostly used for scoring or ranking of document depending

on the rate of query to that particular keyword. After TF-IDF procedure we get two datasets 1 dataset containing an applications package name and there category label whereas other dataset contains the application package name and values obtained form TF-IDF.

After that we merged dataset with respect to package column names. First we calculate Euclidean distance between data instances after this we find neighbours based on calculated distance by sorting distance into ascending order then we get response of a test instance across training dataset and we do a pole between classes

$$d_E(x,y) = \sum_{i=1}^{N} \sqrt{x_i^2 - y_i^2}$$

After computing the distance between two points we are able to find the closest points based on their distance apart and form clusters.By setting k to the number of unique labels we were given, we are able to find the 30 closest groups and thus determine which cluster any test data belongs to by computing its Euclidan distance. It is at the extreme end of the tradeoff between computational efficiency and model flexibility [12].

3.2.2 Naive Bayes

The Naive Bayes machine learning model is a popular statistical learning system that has been successful in many applications where features are independent of each other.Naive Bayes classifier used in this assignment is a simplified probabilistic classifier based on the Bayes theorem. Bayes theorem defines the probability of an event based on the conditions relating to the event [3].

Bayes rule is defined mathematically as

$$P(C|X) = \frac{P(X,C)}{P(X)} = \frac{P(X|C)P(C)}{P(X)}$$

where C and X are two events such that P(C) which is the prior probability and P(X) is the probabilities of A and B independent of each other. $P(C|X)$, is the posterior probability described as the conditional probability of observing event A given that B is true. $P(X|C)$, is the likelihood described as the probability of observing event B given that A is true.

Naive Bayes classifiers can handle an arbitrary number of independent variables whether continuous or categorical. Given a set of variables, $X = x1, x2, x..., xd$, we want to construct the posterior probability for the event Cj between a set of possible outcomes $C = c1, c2, c..., cd$. In a more familiar language, X is the predictors and C is the set of categorical levels present in the dependent variable. Using Bayes' rule:

$$P(X|C) = \Pi_i P(x_i|C)$$

Given a set of variables, we would like to build posterior probability for an event C_j between a possible set of outcomes $C = c_1, c_2...c_n$.

Having identified that the data is was multi-classed, we needed to know the distribution of the dataset. Although we had a large dataset, we wanted to be sure that the proportion of classes weren't skewed. This was proven to be correct with a majority of classes having between 620 to 720 instances (B). Due to the even distribution of classes, a Multinomial Naive Bayes approach was feasible.

4 Experiments

Due to the ever increasing volume of data that needs to be classified, we felt that it was necessary to be able to handle the vast arrays of data and as such our primary goal was to handle large amounts of data efficiently and also to maintain a baseline accuracy level.We recognised that there is an exponential increase in the volume of data available and as such further reinforced our belief that this was a good approach [13].

For this reason we attempted to reach a baseline and then attempt to optimise the code so as to improve on speed, whilst maintaining a respectable accuracy. Due to our primary concerns we used the approach presented by Tang, Term frequency variance to do our pre-processing.By using this pre-processing technique we were able to reduce our matricies by 80% using an automated method which took the average of the variances and used that as the minimum variance in our matrix. This method took 4 seconds to compute for both the training and test data set. When compared to an SVD which took 40 minutes to deconstruct for a gain of 5% in total, we felt that it was not a viable option due to the time constraints on the task.

Furthermore, we were able to determine the specific number of features wanted if necessary.In order to determine the number of features to use, we tested an array of feature sets to determine the highest accuracy keeping all others constant. As seen in Appendix C we were able to show that time increased at an ever increasing rate, whilst accuracy only stayed constant after reaching 0.5. As such for this reason we believed that a number between 1000 and 1500 should be used. As there was bound to be noise in the data, we chose the midpoint 1250 as our 'optimal' number of features.

As explained earlier, the classifier chosen for our experiment was the multinomial Naive Bayes classifier. We chose this classifier as we believed that it would be the most promising when in conjunction with the TfV technique. When working with Naive Bayes, we need to be able to assign a probability to events that may not be represented within the current model. As such the δ value added can range from (0,1]. By trying these features out to best understand the model we see that there is not much difference in accuracy or duration suggesting that the lidstone value will not have an impact on either value (Table 2). As such we chose the median value again and have used a lidstone value of 0.5 for our optimal model.

5 Analysis

It is important to note that there are two types of accuracy that need to be looked at when looking at a multi-class classifier.However in both cases they are calculated the same way

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

The first being the macro accuracy that is how well the classifier performs overall, as well as each individual class

and how well it performed. As shown in Table 3 we can see that the the average for the classifier was 0.52. with a standard deviation of 0.01, it had a 95% confidence interval of [0.49 and 0.56]. The macro precision value was calculated to be 0.58,recall 0.51,F-score 0.50. These values show that the model is only able to explain parts of half of the data, and as such a better feature selection may have been needed to increase the accuracy (Table 4.Furthermore, the range of values presented in both precision, recall and F-score, suggest that an inconsistent set of features was chosen resulting in a skewed result. In order to understand why these values were not high we need to look at the micro values for each category.

5.1 Extensive Analysis

5.1.1 Precision

Precision looks at how many of the items are relevant to the class chosen. Precision had a large range of values from 0.31 to 1. Precision can be weighted also by how many items are in a category as generally smaller categories will have higher precision due to less room for error. Precision is calculated with the formula

$$Precision = \frac{TP}{TP + FP}$$

. We can also say that a high precision means that there is a possibility we don't have enough sensitivity in the model to allow for changes in class. Weather, also had a high precision, suggesting that the feature set chosen was able to capture keywords relevant to describe the class effectively. Conversely, Productivity had did not have a defined feature set and as a result had a very low precision.

5.1.2 Recall

Recall looks at how many of the items chosen for a class are correct and is given by the formula.

$$Recall = \frac{TP}{TP + FN}$$

Recall had a very wide array of values ranging from 0.05(comics) to 0.84 (Cards and Casino). This happens due to certain categories having more unique identifiers and as such allowing for more subtleties to be present in the model. This can be seen with the Cards and Casino category when in comparison to the Comics category. Furthermore, we can see that these subtleties were not present in categories such as entertainment, lifestyle and productivity. This shotgun approach of adding multiple features to each group has resulted in a large amount of false positives for each group.

5.1.3 F-score

F-Score is the weighted average of both precision and recall computed by using both false positives and false negatives of each class. An F-Score is calculated with the following formula:

$$F = \frac{2 * precision * recall}{precision + recall}$$

An f score is useful for weighting both the precision and recall in order to get a better understanding of the values present. By looking at F scores we can see that Categories such as comics and sports games both have extremely low F-scores (less than 0.1) suggesting that both these categories are lacking in terms of precision and recall. In the case of Comics, it has a perfect precision (1) but a recall of 0.05. By looking at the output, we can see that comics got often referenced as 'Books and Reference'. We can assume that may have occurred to a mismatch with similar words such as 'read'. By having a well defined class with lots of features we are able to increase our F-score and as such improve overall accuracy.

6 Discussion

We have built two classifiers Naive Bayes and K-Nearest Neighbours in order to choose the optimum approach. While working we have found that KNN performance was very poor while working with a larger dataset.As such we pursued a Naive Bayes model as it worked much faster and produced an accuracy close to 50%.

During our initial stage of the project, we were trying to evaluate data by analysing the best methods available for reducing the size of the dataset. First, we have counted number of zero present in the dataset and then removed rows with contains zeros, this had resulted in faster computation, however a decrease in accuracy.We then tried using an SVD, but realised it was computationally expensive and as such would not be aligned with our goals. Finally we chose the TfV approach, as we believed it was the most balanced of all the preprocessing models we had available.

It is important for us to note that this model has had a varying level of success across the range of categories, and we feel that it is important to acknowledge the flaws in the current bag of words model. From our results we can see that there is clear improvement that can be done to ensure that smaller groups such as Comics and Sports Games are more well defined, and for larger groups such as productivity, entertainment, business and lifestyle have less indiscriminate words as shown by [14].This can be seen by categories such as Cards and Casino, Racing and Music and audio which are all well defined and as such have both high precision and recall rates. As such, by doing this we should be able to improve overall accuracy whilst maintaining our goal of not increasing the runtime.

6.1 Language Choice

We chose to use Python for this analysis, as we believed that with the large array of libraries available optimisation would be done at an algorithm level by changing inputs rather than having to modify the algorithm completely. Furthermore, the use of pandas data frame exponentially sped up the read time of both training_data.csv as well as test_data.csv when compared to using both numpy arrays or dicts.

7 Conclusion

We have successfully been able to categorise 50% of data within an extremely short period of time. We believe that improvements can be made to this model without hindering speed through much more rigorous pre-processing. However, if we decided to sacrifice speed for accuracy we

can utilise other advanced classification algorithms such as Decision Trees, Support Vector Machines (SVM) and Hidden Markov Models (HMM).

8 Future Work

We can improve categorization accuracy using other advanced classification algorithms such as Decision Trees, Support Vector Machines (SVM) and Hidden Markov Models (HMM).In spite of the fact that computation is intensive it can still perform better than the multinomial Naive Bayes model.Improvements can be made regarding over fitting, by using a k-fold cross validation rather than a single split of the training data to test the model. Furthermore, more work can be done into utilising the names of the applications in order to use hypernyms and hyponyms to derive relations.Finally, increasing the baseline accuracy of this model whilst maintaining its speed will result in a classification model that will be able to sort large corpus with ease.

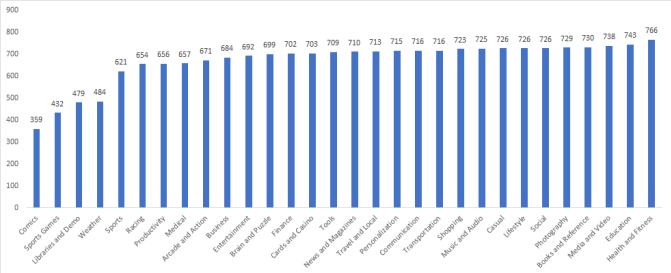
Appendix A

TABLE 1
Naive Bayes tests on preprocessed data

Dataset	Accuracy(%)	Time(s)
Training Data	46.95	369.16
SfV Reduced Data	48.45	293.88
Removed 0	46.06	26.67
SfV + Removed 0	46.66	25.01

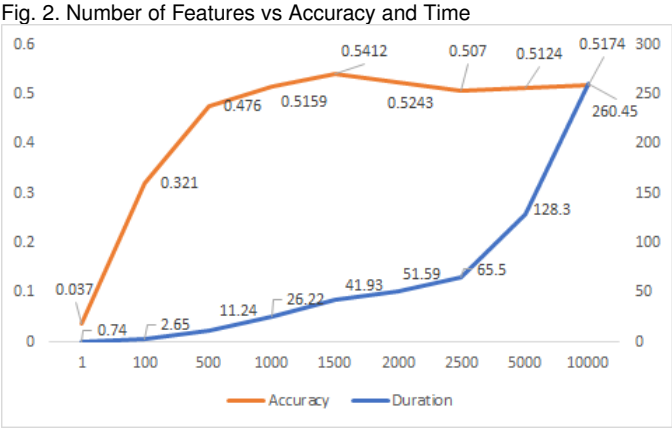
Appendix B

Fig. 1. Number of Applications per Category



Appendix C

Fig. 2. Number of Features vs Accuracy and Time



APPENDIX D

TABLE 2
Lidstone Values vs Accuracy and Duration

Lidstone Value	Accuracy	Duration
0.01	0.51	29.65
0.1	0.5	30.35
0.3	0.52	32.85
0.5	0.52	31.41
0.6	0.52	33.42
0.8	0.51	31.58
1	0.53	31.44

APPENDIX E

TABLE 3
Results for optimum classifier

Iteration	Accuracy
1	0.550746269
2	0.533333333
3	0.536318408
4	0.512935323
5	0.514925373
6	0.490049751
7	0.514427861
8	0.530845771
9	0.510945274
10	0.535820896
Average	*0.52303482*

TABLE 4
Output from MultiNomial Naive Bayes Classifier

	TP	TN	FN	FP	Precision	Recall	FScore
Sports Games	2	1967	40	1	0.67	0.05	0.08
Comics	2	1970	38	0	1.00	0.05	0.09
Productivity	12	1917	54	27	0.31	0.18	0.22
Entertainment	16	1915	57	22	0.42	0.22	0.29
Business	17	1919	53	21	0.45	0.24	0.31
Lifestyle	19	1907	46	38	0.33	0.29	0.31
Medical	27	1894	47	42	0.39	0.36	0.37
Libraries and Demo	13	1959	38	0	1.00	0.25	0.41
Tools	42	1850	39	79	0.35	0.52	0.42
Travel and Local	32	1905	31	42	0.43	0.51	0.47
Arcade and Action	28	1919	41	22	0.56	0.41	0.47
Social	45	1866	23	76	0.37	0.66	0.48
Casual	43	1874	30	63	0.41	0.59	0.48
Media and Video	39	1890	23	58	0.40	0.63	0.49
Education	40	1894	24	52	0.43	0.63	0.51
Communication	41	1893	38	38	0.52	0.52	0.52
Books and Reference	34	1918	42	16	0.68	0.45	0.54
Brain and Puzzle	35	1916	40	19	0.65	0.47	0.54
Health and Fitness	45	1895	32	38	0.54	0.58	0.56
Personalization	53	1876	10	71	0.43	0.84	0.57
Sports	30	1936	33	11	0.73	0.48	0.58
Photography	52	1889	22	47	0.53	0.70	0.60
Transportation	50	1902	26	32	0.61	0.66	0.63
Finance	46	1916	18	30	0.61	0.72	0.66
Weather	23	1966	18	3	0.88	0.56	0.69
Shopping	59	1897	16	38	0.61	0.79	0.69
News and Magazines	61	1897	21	31	0.66	0.74	0.70
Music and audio	53	1913	30	14	0.79	0.64	0.71
Racing	40	1943	15	12	0.77	0.72	0.74
Cards and Casino	56	1932	10	12	0.82	0.84	0.84

REFERENCES

[1] App stores: number of apps in leading app stores 2017 — Statista.

[2] Mobile app usage - Statistics & Facts — Statista.

[3] Babatunde Olabenjo. Applying Nave Bayes Classification to Google Play Apps Categorization.

[4] Ji Zhou. *Automated App Categorization using API analysis*. PhD thesis.

[5] Kevin Murphy. *Machine Learning A Probabilistic Perspective*. MIT Press, 2012.

[6] Jason Brownlee. How to Use Machine Learning Results - Machine Learning Mastery, 2016.

[7] Jason Brownlee. Why you should be Spot-Checking Algorithms on your Machine Learning Problems - Machine Learning Mastery, 2014.

[8] Ionia Veritawati, Ito Wasito, and Mujiono. Sparse data for document clustering. In *2013 International Conference of Information and Communication Technology (ICoICT)*, pages 38–43. IEEE, mar 2013.

[9] Michael E Wall, Andreas Rechtsteiner, and Luis M Rocha. Singular value decomposition and principal component analysis. pages 91–109, 2003.

[10] Bin Tang, Michael Shepherd, Evangelos Milios, and Malcolm I Heywood. Comparing and Combining Dimension Reduction Techniques for Efficient Text Clustering. 2005.

[11] Weil Duan and Xiangyang Lu. Weighted Naive Bayesian Classifier Model Based on Information Gain. In *2010 International Conference on Intelligent System Design and Engineering Application*, pages 819–822. IEEE, oct 2010.

[12] J Johnson. K-Nearest Neighbors — The Shape of Data, 2013.

[13] John Gantz and David Reinsel. Extracting Value from Chaos. 2011.

[14] Lianjing Jin, Wei Gong, Wenlong Fu, and Hongbin Wu. A Text Classifier of English Movie Reviews Based on Information Gain. In *2015 3rd International Conference on Applied Computing and Information Technology/2nd International Conference on Computational Science and Intelligence*, pages 454–457. IEEE, jul 2015.