TEXT CLASSIFICATION WITH DIMENSION REDUCTION ON NEWS ARTICLES

NG KANG WEI

FACULTY OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY
UNIVERSITY OF MALAYA
KUALA LUMPUR

2019

TEXT CLASSIFICATION WITH DIMENSION REDUCTION ON NEWS ARTICLES

NG KANG WEI

RESEARCH PROJECT SUBMITTED TO THE FACULTY OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY UNIVERSITY OF MALAYA, IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF DATA SCIENCE

2019

UNIVERSITI MALAYA

ORIGINAL LITERARY WORK DECLARATION

Name of Candidate: (I.C./Passport No.:)

Registration/Matric No.:

Name of Degree:

Title of Project Paper/Research Report/Dissertation/Thesis ("this Work"):

Field of Study:

I do solemnly and sincerely declare that:

(1) I am the sole author/writer of this Work;
(2) This work is original;
(3) Any use of any work in which copyright exists was done by way of fair dealing and for permitted purposes and any excerpt or extract from, or reference to or reproduction of any copyright work has been disclosed expressly and sufficiently and the title of the Work and its authorship have been acknowledged in this Work;
(4) I do not have any actual knowledge nor do I ought reasonably to know that the making of this work constitutes an infringement of any copyright work;
(5) I hereby assign all and every rights in the copyright to this Work to the University of Malaya ("UM"), who henceforth shall be owner of the copyright in this Work and that any reproduction or use in any form or by any means whatsoever is prohibited without the written consent of UM having been first had and obtained;
(6) I am fully aware that if in the course of making this Work I have infringed any copyright whether intentionally or otherwise, I may be subject to legal action or any other action as may be determined by UM.

Candidate's Signature Date:

Subscribed and solemnly declared before,

Witness's Signature Date:

Name:
Designation:

TEXT CLASSIFICATION WITH DIMENSION REDUCTION ON NEWS ARTICLES

ABSTRACT

Text classification is one of the main tasks in Natural Language Processing (NLP). Text classification assigns text into pre-defined categories. Text classification is needed in news articles to distinct the articles into different categories for ease of searching and viewing. Some news articles in different categories has quite similar features which is a challenge to text classification method. The common document representation method used to extract features from text usually produce a large and sparse matrix. The high dimensionality of the matrix would be a hindrance to the accuracy of the classification models. Thus, dimension reduction algorithm is applied to reduce the dimension of these large and sparse matrix. This study analyse effect of dimension reduction has on different document representation algorithms and subsequently the performance of the classification models. It is found that Support Vector Machine (SVM) has the best overall performance compare to k-Nearest Neighbours (kNN) and Neural Network (NN). Based on the result of this research, Term Frequency-Inverse Document Frequency (TF-IDF) is the better document representation method compare to term frequency. The performance of classification models decreased when dimension reduction is applied since the number of features available is lesser. Dimension reduction is a computationally expensive process thus it increase the time taken. The best result is achieved without dimension reduction but dimension reduction still would serve its purpose when there is memory constraint to store the features.

Keywords: text classification, dimension reduction, TF-IDF, kNN, SVM, NN.

ABSTRAK

Klasifikasi teks adalah salah satu tugas utama dalam NLP. Klasifikasi teks memberikan teks ke dalam kategori yang telah ditentukan sebelumnya. Pengelasan teks diperlukan dalam artikel berita untuk membezakan artikel ke dalam beberapa kategori untuk memudahkan pencarian. Artikel berita dalam kategori yang berbeze menpunyai ciri yang agak serupa, ini adalah satu cabaran untuk kaedah klasifikasi teks. Kaedah perwakilan dokumen biasa yang digunakan untuk mengekstrak ciri-ciri dari teks biasanya menghasilkan matriks yang besar dan jarang. Keamatan dimensi matriks akan menjadi penghalang kepada ketepatan model klasifikasi. Oleh itu, algoritma pengurangan dimensi digunakan untuk mengurangkan dimensi matriks besar dan jarang ini. Kajian ini menganalisis kesan pengurangan dimensi kepada algoritma perwakilan dokumen yang berlainan dan seterusnya prestasi model klasifikasi. Pendapatan kajian ini mendapati bahawa SVM mempunyai prestasi keseluruhan yang terbaik berbanding dengan kNN dan NN. Berdasarkan hasil kajian ini, TF-IDF adalah kaedah perwakilan dokumen yang lebih baik berbanding dengan kekerapan terma. Prestasi model klasifikasi menurun apabila pengurangan dimensi diterapkan sejak bilangan ciri yang tersedia kurang. Pengurangan dimensi adalah proses perhitungan mahal, ia memerlukan masa yang lebih panjang. Hasil terbaik dicapai tanpa pengurangan dimensi tetapi pengurangan dimensi masih dapat memenuhi tujuannya apabila terdapat kendala memori untuk menyimpan ciri-ciri tersebut.

ACKNOWLEDGEMENTS

I would like to take this opportunity to appreciate the helps and guidance given freely by my supervisor, Dr Hoo. Without his guidance and encouragement, this study would not has seen the light of day.

I am thankful and indebted to the lecturers who persevered and pour their hearts out to educate fellow students like me. It is through their hardwork and sacrifices that I have gain knowledges that are crucial to this work.

I am also grateful to my fellow friends who are by my side as we work on our own projects. It is comforting to have someone working by your side towards the same goal.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS AND ABBREVIATIONS

AI	:	Artificial Intelligence.
BoW	:	Bag of Words.
BPLion	:	Back Propagation Lion.
BSS	:	Blind Source Separation.
kNN	:	k-Nearest Neighbours.
LDA	:	Linear Discriminant Analysis.
LSA	:	Latent Semantic Analysis.
NLP	:	Natural Language Processing.
NMF	:	Nonnegative Matrix Factorization.
NN	:	Neural Network.
PCA	:	Principal Component Analysis.
RBM	:	Restricted Boltzmann Machine.
SVD	:	Single Value Decomposition.
SVM	:	Support Vector Machine.
TF-IDF	:	Term Frequency-Inverse Document Frequency.

LIST OF APPENDICES

CHAPTER 1: INTRODUCTION

1.1 Introduction

In this digital age, data is being generated rapidly and in large quantity. Much of the data generated is unstructured data which includes text messages, scientific articles, news articles, blog posts and others. (Çakir & Güldamlasioğlu, 2016). News articles is one of the main sources of these data. News articles are generated around the world at all times. News articles are generated in large quantity and in high velocity that is beyond one's capability to analyse each of them in time. Thankfully, as the news articles generation capacity increases so do the capability of Artificial Intelligence (AI). A branch of AI is created specifically to automate the process with text and speech. The branch of AI is Natural Language Processing (NLP).

The objective of NLP is to allow computer to understand human natural languages. If the objective is achieved, NLP would be able to read and understand all form of human natural languages including text and speech as well as any human, if not better. In the meantime, although the capability of NLP has not reach the ideal yet, it has improved over the years. NLP would be able to analyse the sentiment of text (sentiment analysis), recognized named entity (named entity recognition), classify text into different groups (text classification) and others.

Text classification is the main topic in this research. It is one of the NLP method that group text into different topics or categories. This classification would be helpful for users to narrow down a search quickly, or to know the main topic of the text in a short time. This is especially the case with news articles, it would be great if the news articles are being categorized into distinctive categories according to the user's preferences. Categorized news would be much easier to search through.

The technology breakthrough in recent years, machine learning algorithms, processing power of processors have been a boost to the NLP field. With the breakthroughs, there has been a rapid development of new techniques in NLP and AI that can work without domain experts' supervision. These breakthrough has brought new attempts at text classification at news articles as well. For instance, researchers in Indonesia has created models to classify Indonesian news. (Wongso, Luwinda, Trisnajaya, Rusli, & Rudy, 2017). Indonesian news is still based on Latin characters in which the hurdles are less intimidating. There are researches that create classification models on Arabic news articles. (Einea, Elnagar, & Debsi, 2019). Arabic characters have distinctive differences with Latin characters therefore novel methods have to be applied.

There are several approaches to the text classification problem, multi-label and classification. In multi-label, each document can belong to several categories. In classification, each document can only belong to one category. Besides classification and multi-labeling, there is clustering, which is an unsupervised machine learning method. In clustering, the data is not labeled and is grouped by similarities based on a similarity measure. This research shall focus on classification rather than multi-label and clustering.

In text classification, most of the algorithms used vector space model to represent the unstructured textual data. (Ababneh, Almanmomani, Hadi, El-Omari, & Alibrahim, 2014). This vector space model represent the sequence of the textual features and their weight, it is easy to implement and provide uniform representation for documents. However, it has a drawback, since it represent all the words in the documents, the dimension of the vector would be huge. This huge vector space model would impact the performance of the machine learning tasks. (Moldagulova & Sulaiman, 2018). Dimension reduction algorithms would be able to reduce the dimension of the vector space model but the reduction would certainly affect the performance of the classification models. Therefore,

this study would investigate the effect of dimension reduction on vector space model and subsequently the performance of classification models in text classification.

1.1.1 Problem Statement

Vector space model is one of the most commonly used document representation algorithms in most of the fields, news article text classification included. However, dimension of the feature space from vector space model can be too large and the vectors can be too sparse. This large and sparse vector space would present difficulties to the classification models. Dimension reduction could reduce the dimension of the vector space model so that the vector space is not that large and sparse but this reduction would certainly affect the performance of the classification models. Different document representation method would produce different vector space model on the same news article. Different dimension reduction method applied on the same vector space model also produces different outcomes. With the same reduced vector space model, different classification models would produced varied result. This research would attempt to provide an insight into which combination of the document representation, dimension reduction and classification models would produce a satisfactory result.

1.1.2 Research Objectives

1. To identify a document representation algorithm that can extract features from news articles in optimum condition.
2. To investigate the effect of dimension reduction on feature matrix has on the performance of text classification models.

3. To evaluate the performance of the chosen text classification models with different document representation method and dimension reduction method.

1.1.3 Research Questions

1. Which document representation method is optimized to extract the features from news articles?
2. How would dimension reduction influence the accuracy and performance of text classification models?
3. Which of the combination of document representation method, dimension reduction method and text classification model has the better performance?

1.1.4 Research Motivation

In this age of consumerism, people are eager to consume everything. News being one of the most consumed product. Events are happening everywhere in this world and news articles are the vehicle that make the events public knowledge. There are numerous news agency around the world and they are churning out news around the clock. The amount of news generated is more than what a single human can consume and analysed. This is where NLP would be able to help. This study is about text classification, one of the pillars of NLP.

In text classification, the unstructured text or news articles in this case would be given a label or multiple labels depend on the method used. These labels would make the news articles more meaningful and searchable. Users can search for a topic just by selecting the text with the particular label rather than performing a manual key word search on all the news articles. In another scenario, users could know what is the topic of a news article in real time if they feed the news article to a text classification model.

Bag of Words (BoW) is the most commonly used document representation method

in text classification. BoW would produce a vector space model of the textual data representation. The dimension of this vector space model would be huge because the amount of words in the documents is large. This huge dimension of vector space model would be a problem to text classification models as the models need a large amount of memory to store this huge and sparse matrix. Furthermore, in order for the classification models to learn from the patterns of the large matrix and able to classify another text correctly, this would need an immense amount of computing power to process the huge and sparse matrix.

With the emergence of dimension reduction algorithms, the dimension of the vector space could be decreased, less memory would be needed to store the matrix and less computing power would be needed to process the matrix. Compressing or transforming the huge matrix into a matrix with lesser dimension would need some computing power as well. This is the cost to reduce the dimension. This reduction of dimension in the vector space model would certainly affect the performance of the text classification models. This study would find out what are those effects.

This study would investigate the effect of dimension reduction algorithms on different document representation method and subsequently the performance of text classification models.

1.1.5 Research Significance

This study would classify news articles into different categories. In order to do that, first the features have to be extracted from the text of the news articles, document representation. The features might need to be compressed or transformed, dimension reduction. Then the transformed features are used to train classification models. After that, the trained models are validated and tested to evaluate its performance.

There are a number of document representation methods, each of these methods would have its advantages and disadvantages over one another. In addition to dimension reduction and classification algorithms, there are myriad of combinations of these algorithms from the 3 stages in order to produce a satisfactory classification models.

This study would explore some of these methods in each of the stages namely document representation, dimension reduction and classification models. This study is trying to determine which combination of document representation method, dimension reduction and classification model would have the best performance. The way to get to the bottom of these question is to apply the chosen methods to a dataset and analyse the result.

Text classification on news articles would work fine if not better without dimension reduction, but dimension reduction still has its uses. In the situation where memory is a constraint, dimension reduction could reduce the amount of memory needed to store the feature matrix. Even with this reduction in memory space, the accuracy must be proven to be still at a satisfactory level. This study would attempt to find a combination of method that could find the said sweet spot where memory space needed is lesser and yet a comparable accuracy is achieved. If this is found and proven, text classification would be more widespread. Smaller devices with less memory would be able to perform text classification on the fly.

1.1.6 Expected Outcome

1. A prototype text classification model that can achieve a satisfactory accuracy in optimum condition
2. A working pipeline of converting raw text to vector space model or features to be processed by classification models
3. A suitable text classification algorithm is applied on the text classification application

4. A performance evaluation on the text classification models when dimension reduction algorithms are applied

CHAPTER 2: LITERATURE REVIEW

2.1 Introduction

This study is investigating the effect of dimension reduction on the feature matrix extracted from news articles. The effect would be shown in the results of the classification models trained with the reduced feature matrix. This would be the whole pipeline of text classification. Document representation extracts features from news articles. Dimension reduction reduces the dimension of the feature matrix. Classification is the part where the data is used to train classification models and the results are analysed. Thus, in this section, methods from these 3 stages in text classification would be reviewed.

2.2 Document Representation

2.2.1 Term Frequency

Term frequency is the simplest form of document representation method in BoW approach. It simply convert the words appeared in all the documents or text into a matrix or vector space model.

The columns are all the words appear in the document, each column represent one word. Each of the rows represent each of the document. The value in the cells represent the number of times a word appear in a document, in other word, the value is the term frequency.

Even though it is a simple document representation method, it is proven to be able to achieve a satisfactory results with the right processing and methods. (Moldagulova & Sulaiman, 2018). In addition, term frequency would produce a big and sparse matrix which is exactly what this study is focusing on. Therefore, term frequency would be a suitable document representation algorithm to be applied in this study.

2.2.2 Term Frequency - Inverse Document Freqeuncy (TF-IDF)

Term frequency-Inverse document frequency (TF-IDF) is also a BoW approach. It is build on the concept of term frequency but take it a step further to overcome the setback of term frequency.

Besides term frequency, TF-IDF also take into account the inverse document frequency which means that it also take the frequency of words in other documents into account. TF-IDF provide a measure of weight or importance to the words. The value of TF-IDF estimate the amount of information provided by each word in its document. The value of TF-IDF increases proportionally to the number of times a word appears in a document but is offset by the frequency of the word in the corpus. (Vijayarani, Ilamathi, & Nithya, 2015). The characteristic of TF-IDF can counter the high frequency of some common words, such as the stop words in a language. TF-IDF can calculated as shown below:

$$TFIDF = tf \times \log_e \frac{N}{df} \tag{2.1}$$

where:

tf = the number of times a word appears in a document

N = the total number of documents in the corpus

df = the number of documents that contain the word

The main differences between term frequency and TF-IDF is that in term frequency, if a word has a high frequency in many documents it would be a prominent feature of the data. In TF-IDF however, term frequency alone would not make a word a prominent feature. If a word has high frequency in many documents, the inverse document frequency of TF-IDF would offset its high frequency therefore the word would not be a prominent feature. A word with high TF-IDF score would appear many times in a document but not in many documents.

The value in the vector space model resulting from TF-IDF would be in decimals as the values assigned are scores or weights for each of the words. The values in the resulting vector space model from term frequency on the other hand, consists of integers which represent the frequency of the words appear in the text.

TF-IDF is a simple document representation method that has proven to be efficient and reliable. It has been applied in the field of text classification on news articles by Wongso, Luwinda, Trisnajaya, Rusli and Rudy (2017) from Indonesia. The researchers applied TF-IDF on Indonesian news articles to extract the features from the text. (Wongso et al., 2017). Even though this study is focus on news articles in English, the method applied would be similar since Indonesian is still a Latin character based language.

TF-IDF has a few setbacks as it only take into account the term frequency and inverse document frequency and not the semantic of the words. (Qu, Wang, & Zou, 2008). Another setback of TF-IDF is when a prominent term is found in many documents, such a term would given a lower weight or points. Chen (2016) found this problem while applying TF-IDF in the big news retrieval system. (C.-H. Chen, 2017). When an important term is referred to noise by TF-IDF, news retrieval without the important term would prove to be difficult. However, this situation only occur when there is a bias in the dataset when an important term appear in many of the news articles or document. As long as the dataset is not bias, TF-IDF would still be an effective document representation method.

2.2.3 Summary

Both term frequency and TF-IDF are chosen as the document representation method in this study. Term frequency and TF-IDF are simple and relatively easy to implement. Both of the methods are efficient and tried and true over the years. Most importantly, the output of huge and sparse matrix from term frequency and TF-IDF is the topic of interest in this

study.

2.3 Dimension Reduction

2.3.1 Naive Dimension Reduction

Naive dimension reduction is the simplest of all dimension reduction algorithms. It is simply removing terms that appear less frequent from the feature matrix. As explained in the document representation section, term frequency and TF-IDF would produce a large and sparse matrix based on the frequency of all the words in the corpus or news articles.

Naive dimension reduction would remove those words or terms that only appear less than a certain number of times in the whole corpus. For instance, a special word is only used in one of the document out of all the documents in the corpus. This word would create a column in the feature matrix while only one row of this column has a value while all the other rows of this column would be 0. This column would be removed under naive dimension reduction. By removing columns such as this, the dimension of the feature matrix is reduced and only the important features of the matrix remained.

Moldaguva and Rosnafisah (2018) has applied this naive dimension reduction on news articles dataset and has found that it increase the performance of the classification models by reducing the dimension of the feature matrix. The performance enhancement is due to the computation complexity of the classification model is decreased as the number of dimension decreased. (Moldagulova & Sulaiman, 2018).

2.3.2 Principal Component Analysis (PCA)

Principal Component Analysis (PCA) is probably one of the most popular multivariate statistical technique in dimension reduction. PCA would transform a data table with values from inter-related variables into new orthogonal variables. The variables would

be the principal components of the data. In other words, PCA transform data from high dimensional space into low dimensional space with linear transformation while preserving the original data features as much as possible. (Ma & Yuan, 2019). The output from PCA is the principal components of the data which hold 80% to 90% of the information of the original data.

PCA has several objectives, it would extract the most important information from the dataset, the dimension of the dataset would be compressed so that only the important information remained. The dataset is also simplified after PCA has processed it. (Abdi & Williams, 2010). This simplification of the information would reduce its size in memory, thus requiring less memory to store the dataset.

PCA has been successfully applied to many fields, one of them is text classification problems. Narhari and Shedge (2017) has proposed to apply PCA in a text clustering algorithm for Marathi regional language. The proposed method replaced Single Value Decomposition (SVD) with PCA in the existing text categorization algorithm. The findings suggest PCA has a better performance than SVD. (Narhari & Shedge, 2017). However, it is important to stress that this study has a different scope that the research of Narhari and Shedge (2017). This study is focus on English text and would apply classification rather than clustering.

PCA is proven to be efficient in pattern recognition and image analysis and has been extensively applied in face recognition system. However, it is found that PCA is not capable of processing data with high dimensionality and sparsity. PCA is only effective when reducing tens or few hundreds of dimensions. (Rajashekharaiah, Chikkalli, Kumbar, & Babu, 2018). Thus, PCA might not be suitable to be applied in this study which involves news articles data. The news articles would be converted into a large and sparse matrix.

2.3.3 Nonnegative Matrix Factorization (NMF)

Nonnegative Matrix Factorization (NMF) is a multiplicative updating solution to decompose a nonnegative temporal-frequency data matrix into the sum of individual components. The sum of individual components are calculated from a nonnegative basis matrix. (Chien, 2019).

NMF has been applied to find latent topics hidden within unstructured text, which is similar to text classification. Chen, Zhang, Liu, Ye and Lin (2018) applied NMF in "Knowledge-Guided Non-Negative Matrix Factorization for Better Short Text Topic Mining" (KGNMF), which is a topic mining engine for short text. The short text applied by Chen and others include news articles. NMF is compared with Linear Discriminant Analysis (LDA) and it is found to perform better than LDA. The resulting model built from NMF is a time-efficient algorithm and has a better performance than the popular text mining algorithm, LDA. (Y. Chen, Zhang, Liu, Ye, & Lin, 2019).

In PCA and SVD, the signs of the data is not restricted in any way but NMF has a non-negativity constraint. This means that NMF can only described by using additive components only. This is due to the influences of classical studies where most of the values are in the positives. This non-negativity constraint of NMF has proved to be problematic when the data matrix is not strictly in the positives. (Allab, Labiod, & Nadif, 2017).

NMF has been applied to learn the latent space of the input text data from Twitter. With NMF the data is decomposed into bi-orthogonal non-negative 3-factor, the rows and columns from the different axis are simultaneously clustered. (Lahoti, Garimella, & Gionis, 2018).

NMF might be more suitable for clustering as it took the least amount of time in clustering the data compared to SVD and PCA. (Li & Ding, 2018). This advantage of NMF over SVD and PCA is negligible since this study would focus on classification rather

than clustering.

2.3.4 Truncated Single Value Decomposition (SVD)

Single value decomposition (SVD) is one of the most commonly used dimension reduction algorithms. It generalizes a complex matrix with many dimensions into a matrix of lower dimension via an extension of the polar decomposition. SVD detects the part of the data that contains the maximum variance in a set of orthogonal basis vectors. The data with the maximum variance would be the most prominent features of the data. (Sweeney et al., 2014).

The mathematical equation for SVD of a matrix X is shown as follows:

$$X = USV^T \tag{2.2}$$

where:

U = an $m \times n$ matrix, columns are left singular vectors

S = an $n \times n$ non-zero diagonal matrix, the singular values

V^T = an $n \times n$ matrix, rows are right singular vectors

Latent Semantic Analysis (LSA) a technique applied in natural language processing that apply SVD in its process. SVD is applied in LSA to transform the features by dropping the least significant values in the matrix thus reducing the dimensions of the matrix. (Karami, 2017).

SVD has been applied extensively in text classification, it is an established method in text classification field. A new method has been derived from SVD for more sophisticated feature projection. SVD also has comparable performance with other state of the art dimension reduction method such as PCA and LDA. (Mirończuk & Protasiewicz, 2018).

Wongso and others (2017) also applied SVD on the output matrix of TF-IDF for feature selection before training the classification models. This shows that SVD still trusted and able to perform well in text classification. (Wongso et al., 2017).

Truncated SVD is a variant of SVD. Similar to PCA, truncated SVD is a matrix factorization technique that factors matrix M into 3 matrices namely, U, Σ and V. There is a slight difference between truncated SVD and PCA. PCA performed the factorization on the covariance matrix while truncated SVD performed the factorization on the data matrix directly. Truncated SVD differ slightly from SVD in the way of that SVD would always produce matrices of n columns if given $n \times n$ matrix but truncated SVD given the same matrix can produce matrices with specified number of columns. (Hauck, 2014).

Truncated SVD can reduced the dimension of the data into lesser dimension when compare to other dimension reduction algorithms such as LDA and PCA. The classification model trained with the output of SVD is also higher than that of LDA and PCA. (Rajashekharaiah et al., 2018). Thus, SVD would be a great choice for the dimension reduction algorithm in this study.

2.3.5 Summary

The 3 dimension reduction algorithms above are considered to be Blind Source Separation (BSS) methods, unsupervised learning algorithms. Out of the 3 dimension reduction algorithms, truncated SVD is the most suitable to be applied in this study. This is because PCA is not efficient in reducing high dimensional and sparse data and NMF is only efficient in clustering algorithm. Most importantly, truncated SVD overcome the drawback of PCA and has been proven to be effective and achieve better result than PCA and others. Thus, truncated SVD is chosen to be the dimension reduction algorithm to be applied on the feature matrix extracted from the text.

2.4 Classification Models

2.4.1 Support Vector Machine

Support Vector Machine (SVM) is a machine learning algorithm that construct a hyperplane to separate the data points into different classes. It has been proven to be very effective in dealing with high dimensional data. (Shinde, Joeg, & Vanjale, 2017). It is also proven to produce dramatically better results in text classification shown in experiments with the Reuters dataset. (Dumais, Platt, Heckerman, & Sahami, 1998). However, various issues need to be considered when applying SVM in text classification, the processing of the data, which kernel to use, and the parameters of SVM. A variant of SVM, called one-class SVM which is trained only with positive information has been used in text classification. (L. M. Manevitz & Yousef, 2002). The authors experimented with different kernels of SVM with different type of document representation method. The kernels tested include linear, sigmoid, polynomial, and radial basis. The document representation methods applied are binary representation, frequency representation, TF-IDF, and Hadamard representation. The best result, F1 score of 0.507 is achieved with binary representation, feature length 10 and with linear kernel function.

In another research, the researchers apply SVM in the classification on web document including news articles and ordinary text document. The document representation method used in this research is vector space model, just the nouns term on the web pages. The researchers experimented with different SVM kernels and varying the size of the training sets. As expected, the precision, recall and accuracy increased as the size of the training set increase. Linear kernel achieved the best result out of the various SVM kernels, a classification accuracy of 80% is achieved. (Shinde et al., 2017).

SVM is proven to be effective in many fields including text classification. The only drawback with SVM is that it can be tricky to find an appropriate kernel for the problem,

but from the result of several researches above, the most suitable kernel in text classification is most probably the linear kernel.

2.4.2 k-Nearest Neighbours (kNN)

K-Nearest Neighbours (kNN) is a classification machine learning algorithm that classify data based on Euclidean distance between the new data point with the existing data points in the feature space. It is a simple yet effective classification technique, as it only need 3 prerequisites. The 3 prerequisites are training dataset, similarity measure and the value of k which is the number of closest neighbours to be considered.

The similarity measure used in kNN is usually Euclidean distance. The mathematical formula for Euclidean distance is as follows:

$$p = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \tag{2.3}$$

where:

p = the distance between 2 data point

x_1 = the x-coordinate of a data point A

x_2 = the x-coordinate of data point B

y_1 = the y-coordinate of data point A

y_2 = the y-coordinate of data point B

KNN needs minimal training, it only needs to plot the training samples on a feature space and calculate the Euclidean distance between the new data point with the training samples to determine which category the new data point should be classified as. KNN has been applied in text classification before, it is found that kNN take significantly longer time to classify a document. This is because kNN need to compute the distance between

the new data point with the existing data points and find the nearest data points. Since the authors are using term vector space document representation method, the dimension of the feature space is high, thus more time is needed for kNN to compute all the distance between the new data point with the training data points. Other than the time taken to compute the distance, the k value is another obstacle in kNN algorithm. In a high dimensionality feature space and the points are not evenly distributed, the k value is hard to be determined.

To overcome the problems mentioned above, the authors applied naive term vector space reduction method, divide the document feature matrix into parts. Term vector space reduction reduces the sparsity of the document term matrix by removing the features less appeared in the corpus. By reducing the term vector space, a slight deterioration in the classification accuracy but the time cost is dramatically reduced. kNN still achieved an accuracy of 92.7% but the time taken reduced from 53 minutes to 11 minutes. (Moldagulova & Sulaiman, 2018).

Therefore, it is shown that kNN though simple, can still perform well if properly used.

2.4.3 Neural Network

Neural network (NN) has a resurgence in recent years as there is a breakthrough in the neural network since Geofrey Hinton discovered a technique called Contrastive Divergence that could quickly model inputs in a Restricted Boltzmann Machine (RBM). RBM is a 2-layer neural network that model the input by bouncing it through the network. This process is less computationally complex than backpropagation. (Hinton, 2002).

Currently, neural network is applied in deep learning to solve various problems, text classification being one of them. Ranjan and Prasad (2018) applied Lion Fuzzy Neural Network on text classification. The researchers used WordNet ontology to retrieve the semantic of the words, and then added context information onto it, thus the features

obtained are semantic-context features. The classification part is performed by Lion Fuzzy Neural Network, which is a variant of Back Propagation Lion (BPLion) Neural Network that includes fuzzy bounding and Lion Algorithm. The neural network model used is trained incrementally. It achieves a higher accuracy than Naïve Bayes and other variant of the Lion Neural Network. (Ranjan & Prasad, 2018).

Besides the modified neural network shown above, a simple feed-forward neural network is also proven to be efficient in text classification. By using the Hadamard product as document representation method, a simple neural network also can achieve a good classification accuracy in text classification compare to Naïve Bayes, kNN, and SVM. (L. Manevitz & Yousef, 2007).

2.5 Conclusion

In the document representation algorithm, both term frequency and TF-IDF would be applied in this study. Both of the document representation algorithm would produce matrix with high dimension. This is the purpose of this study and it would be interesting to observe how different document representation algorithm affect the result.

For dimension reduction algorithm, truncated SVD is chosen as the dimension reduction algorithm to be used in this study. SVD has proven to be efficient and achieve satisfactory result from past researches.

In machine learning algorithms for text classification, all 3 machine learning algorithms reviewed above would be applied. One of the objectives of this study is to investigate the performance of different machine learning algorithms in text classification. Therefore, the same dataset would be used to train 3 models and the performance of the 3 classification algorithms would be evaluated in order to identify the best classification model.

CHAPTER 3: RESEARCH METHODOLOGY

3.1 Introduction

This section would illustrate the process flow to carry out the experiments and fulfill the aforementioned objectives of this study. The following flow charts would illustrate the steps taken to conduct the several experiments.

3.2 Text preprocessing process flow

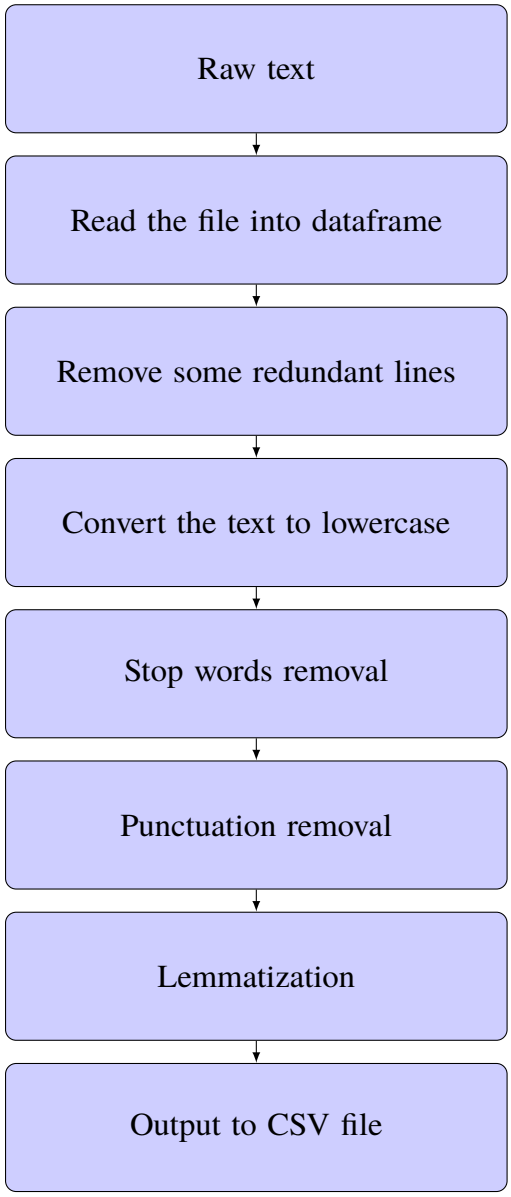

Figure 3.1: Process flow of preprocessing the news article dataset

The flow chart above illustrates the preprocessing process. The news articles dataset is in the form of raw text. Some preprocessing is needed before the features can be extracted from the text to build the text classification models. First, the raw text would be read into a dataframe or a data structure for ease of access and processing.

There are some lines in the raw text files that are irrelevant to text classification, these lines would be removed to reduce the noise in the dataset. All the text in the dataset would be converted to lowercase for uniformity and ease of processing. There are some words in the text that are useful in human speech but do not convey meanings in text analysis. These words are stop words, these words would be removed. After stop words removal, the text should contain only the words that are vital to text analysis but there would be some punctuations and symbols in the text. These punctuations and symbols would be removed as well so that the text is cleaner and contains less noise.

After that, there is an important step, lemmatization. Lemmatization would transform most of the words in the text to their root form which is known as a lemma. This would reduce the noise in the data by transforming similar words into a single word. The alternative to lemmatization would be stemming. Stemming would be faster and need less processing power than lemmatization but stemming has a drawback. The result of stemming may not be a real word because stemming just chop off the end of the words. On the other hand, lemmatization uses vocabulary and morphological analysis of words to remove the inflectional endings only, producing the root form of words. (Nicolai & Kondrak, 2016). Therefore, the output of lemmatization would be better than stemming.

The purpose of lemmatization is to reduced the number of distinct words in the document. The words, "good", "better", and "best" might appear in a news articles dataset. Without lemmatization, these words would be 3 distinct words. With lemmatization, all 3 of these words would be transformed to their root word which is "good". Therefore, by using

lemmatization, the number of distinct words in the text would be reduced and the noise in the data would be minimized.

After all the words in the text file are lemmatized, the dataframe that contains the text is written to a CSV file. The training process would be able to read the CSV file easily and train the classification models with the method chosen.

3.3 Overall Process Flow

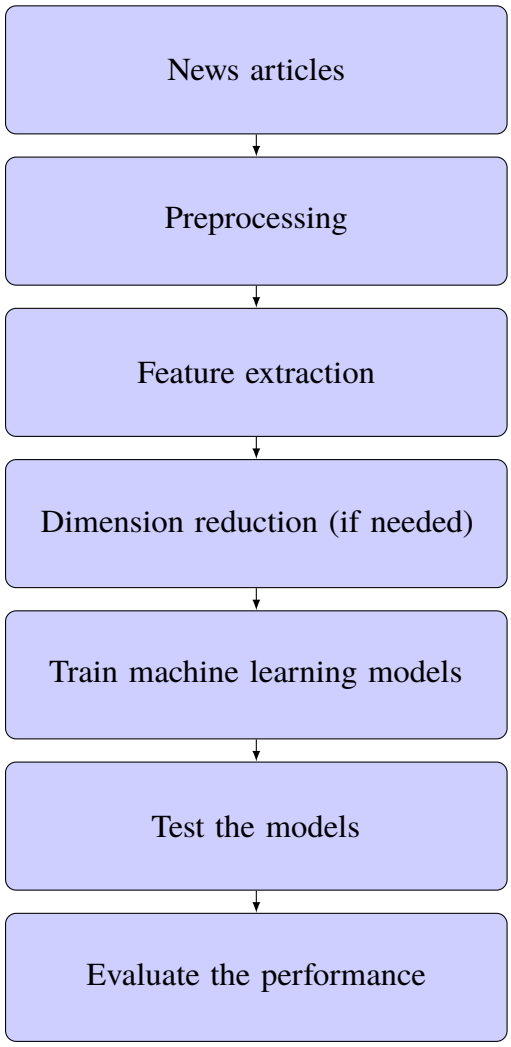

Figure 3.2: Overall process flow

The process flow chart above is the overall process for the few experiments. As mentioned above, the news articles dataset has to be preprocessed before it can undergo feature extraction and being used to train machine learning models.

After preprocessing, the resulting data would be used in several experiments with a few subtle differences. The differences in the experiments would be in the feature extraction stage and the dimension reduction stage.

There would be 2 different document representation method or feature extraction method in the experiments. An experiment would be conducted with each of the feature extraction method without dimension reduction. Several classification models would be trained with the features from both of the feature extraction methods without dimension reduction. By evaluating the performance of the classification models, a more optimized feature extraction method would be identified thus fulfilling the first objective of this study.

In a number of experiments, the feature matrix output of the 2 feature extraction method would undergo dimension reduction. There is a parameter to the dimension reduction methods to control to what extends should the feature matrix be reduced to. By manipulating this parameter, feature matrices of different dimension would be produced. These matrices with different dimension would be used as training data for classification models. Different dimension of the feature matrices would certainly has an effect on the performance of the classification models. The performance of the classification models trained with feature matrices of different dimension would be evaluated. Thus, the effect of dimension reduction has on the performance of classification models could be determined. This would fulfill the second objective of this study.

Since there are 2 document representation method and 2 dimension reduction method, there would be a combination of 4 experiments. In addition to the experiments without dimension reduction, there would be a total of 6 experiments. These experiments use different combination of document representation method, dimension reductions and all the 3 classification models. From the results of all the experiments, the best combination of document representation, dimension reduction and classification models for news articles

would be identified, which is the third objective of this study.

3.4 The dataset

The dataset used in this research is the 20 news group dataset. It is freely available at `http://qwone.com/~jason/20Newsgroups/`. It is a collection of news articles that can be divided almost evenly to 20 groups. Some of the groups are closely related to one another and some are widely different. Categories with the same prefix for example *comp* would have high similarity with one another but also subtle differences. The categories with the different prefixes, for example *rec* and *talk* would be very different from one another.

This characteristic of the dataset make it a good dataset to test text classification. A good classification model should be able to differentiate the different groups of news article even the groups that closely resembled one another.

The table below show the number of count for each category. There is a total of 18790 articles and most of the categories have almost 1000 articles in them.

The bar chart below visualized the data displayed on the table for ease of viewing. Most of categories have almost 1000 articles except alt.atheism, talk.politics.misc and talk.religion.misc.

As mentioned there are 18790 articles in the data, the articles would be split at random into 8:2 ratio. 80% of the data would be used as training data to train the classification model while 20% of the data would be use as the test data. The training data would consist of 15032 articles while the test data would consist of 3758 articles.

Category	count
alt.atheism	798
comp.graphics	970
comp.os.ms-windows.misc	980
comp.sys.ibm.pc.hardware	978
comp.sys.mac.hardware	957
comp.windows.x	978
misc.forsale	962
rec.autos	988
rec.motorcycles	993
rec.sport.baseball	993
rec.sport.hockey	998
sci.crypt	991
sci.electronics	981
sci.med	988
sci.space	986
soc.religion.christian	997
talk.politics.guns	910
talk.politics.mideast	940
talk.politics.misc	774
talk.religion.misc	628

Table 3.1: The count of each category in the dataset

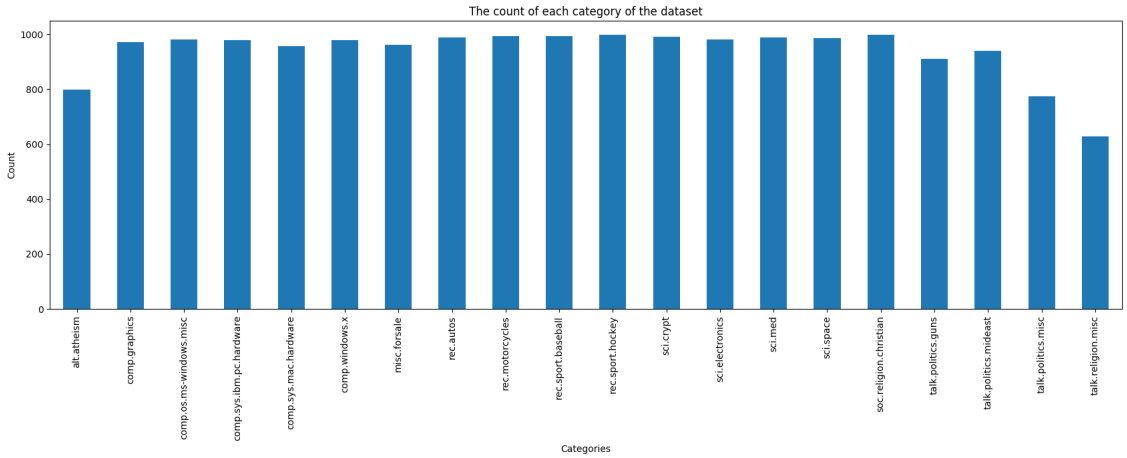

Figure 3.3: The count of each category in the dataset

3.5 The experiments

The experiments that has been conducted in this research is as follows:

1. Term frequency
2. Term frequency with naive dimension reduction
3. Term frequency with truncated SVD

4. TF-IDF
5. TF-IDF with naive dimension reduction
6. TF-IDF with truncated SVD

Basically, there are 2 feature extraction method used in the experiments, namely term frequency and term frequency - inverse document frequency (TF-IDF). Each of the feature extraction method would be tested with and without dimension reduction algorithm.

There are 2 dimension reduction algorithms involved, one is a naive method, which means that the features or columns lesser than a certain value would be removed. In other words, words or terms that do not appear much in the dataset would be removed. Another dimension reduction algorithm is truncated single value decomposition (SVD). This method would retain the essence of the data, the part of the data with maximum variance.

After feature extraction and dimension reduction (if needed) are applied, the resulting features would be used to train text classification models. There are 3 text classification algorithms chosen in these experiments namely k-nearest neighbour (kNN), support vector machine (SVM), and neural network (NN). All 3 of the machine algorithms would be applied to all the different resulting features and the accuracy scores would be evaluated.

3.6 Conclusion

With the method mentioned above, the experiments would be carry out and the results would be recorded. The results would be compared and the differences would be analysed.

CHAPTER 4: RESULTS AND DISCUSSIONS

4.1 Introduction

The results of the experiments are shown in this section. The differences in the accuracy and performance of each of the methods would be discussed and analysed. Hopefully, the results and discussions would be able to answer the research questions posted in the beginning of this research.

4.2 Term Frequency

Table 4.1: Term frequency

ML	no of features	accuracy	time taken (s)
kNN	8000	0.31	3.74
SVM	8000	0.81	5.27
NN	8000	0.84	91.82

Term frequency is one of most common feature extraction method in text. It convert all the words in the dataset into a matrix where each column is a word and the value in each of the cells are the number of times each word appear in the text. Each row in the matrix is a document. This vector space model representation of the words would result in a sparse matrix since each of the documents only contains a subset of all the words in the whole dataset.

In this experiment, the number of features of vector space model from term frequency extraction is limited to 8000, this is because of the memory constraint of the machine, if it is unlimited, the resulting matrix would be of bigger size and would have the risk of running out of memory while processing the matrix.

In the results shown above, NN achieved an accuracy of 0.84 which is the best accuracy out of the 3 classification algorithms but it is also the one that takes the longest to process

which is 91.82s. KNN took the least time to process, 3.74s but has the lowest accuracy score, 0.31. Overall performance, SVM is the best, it only took 5s to process, which is 80s less than NN and has an accuracy of 0.81 which is comparable with NN.

KNN accuracy is low in this scenario is possibly due to the sparsity of the feature matrix or vector space model. The data points are few and far in between. KNN is designed for low dimensional data therefore it doesn't perform well in large and sparse data. The underlying reason would be due to kNN classify a new data point based on the distance of the new data point with the nearby data points. Since the other data points are few and sparse, the classification of the new data points would be biased to the few data points that are near. This bias would reduce the accuracy of the classification. (Wang, Lin, Huang, Wang, & He, 2010).

Neural Network (NN) took the longest time to process the features, this is due to the number of hidden layers in NN. The vector space model even though sparse has high dimension, NN would need as much if not more neurons as the number of features to process the data. This processing would takes both time and computing power.

SVM depends on the prominent features of the dataset, the support vectors to classify the dataset so it is relatively faster and as accurate.

4.3 Term Frequency with Naive Dimension Reduction

The feature extraction method used in this experiment is same as the experiment above but dimension reduction method is applied to the resulting matrix before training. The dimension reduction algorithm used in this experiment is a naive one which means that the columns with the least occurrence of word are removed assuming those words are unimportant and would not have much influence on the accuracy.

In this experiment, kNN still has the worst performance among the 3 which is at 0.25

Table 4.2: Term frequency with naive dimension reduction

ML	no of features	accuracy	time taken (s)
kNN	6026	0.25	4.16
kNN	4058	0.26	4.22
kNN	2020	0.27	4.52
kNN	1030	0.30	5.08
kNN	508	0.32	5.14
kNN	108	0.30	5.10
kNN	54	0.26	4.77
SVM	6026	0.81	6.57
SVM	4058	0.76	6.63
SVM	2020	0.74	7.85
SVM	1030	0.68	9.52
SVM	508	0.64	20.86
SVM	108	0.42	37.48
SVM	54	0.31	39.71
NN	6026	0.85	67.65
NN	4058	0.83	63.72
NN	2020	0.78	43.38
NN	1030	0.73	42.89
NN	508	0.65	50.37
NN	108	0.41	82.84
NN	54	0.34	82.39

with 6026 number of features. As the number of features decreases to 508, the accuracy increases slightly to 0.32. As the number of features decreases, the time taken for kNN to produce the result increases from 4.16s to 4.77s.

SVM and NN have the same trend in this experiment, the accuracy decreases as the number of features decreases. Accuracy of SVM decreases from 0.81 to 0.31 as the number of features decreases from 6026 to 54. The time taken by SVM increases as the amount of features decreases, it increases from 6.57s to 39.71s.

The time taken by SVM increases quite drastically when the number of features decreased. Since the dimension reduction method used is a naive dimension reduction, no high computational cost functions are involved, the reduction would not take much time. The bulk of the time taken is for SVM to classify the text. The number of features in the matrix decreases thus the sparsity of the vector space model decreases, resulting

in a denser vector space model. In this case, SVM need to take into account more data points to calculate an optimum hyperplane to separate the data points into different classes. Therefore, more time is taken to compute the optimum hyperplane.

On the other hand, the accuracy of NN decreases from 0.85 to 0.34 as the number of features decreases from 6026 to 54. However, the time taken by NN fluctuates from 67.65s to 42.29s and then to 82.39s when the amount of features decreases from 6026 to 1030 then to 54. The overall decrement in time taken is possibly due to the reduction of features, less neurons are needed to process the data and thus the speedup.

SVM and NN performs better when there are more features, the time taken would be relatively lesser and the accuracy would be higher than in the scenarios with lesser number of features. This show that these 2 classification algorithms are able to process features in a large and sparse vector space effectively.

4.3.1 Graph of Term Frequency with Naive Dimension Reduction

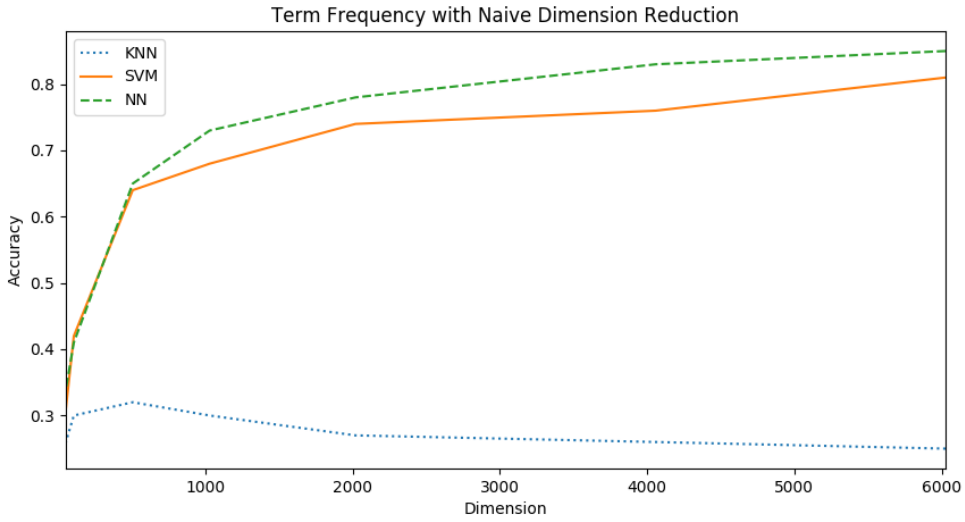

Figure 4.1: Term Frequency with Naive Dimension Reduction

The figure above illustrates more clearly the results shown in this section. The graph shows the change in accuracy of the classification models as the number of features decreases.

In this graph, it is clearly shown that kNN is not as efficient as SVM and NN, its accuracy is much lower than that of SVM and NN across the spectrum of dimension. The accuracy of kNN increases slightly as the number of features decreases.

NN has a slightly higher accuracy than SVM all the scenarios when the number of features is greater than 1000. When the number of features is lesser than 1000, both NN and SVM has similar performance. The accuracy of these classification models decreases as the number of features decreases, which is expected. When the number of features decreases, the classification models would have less information to classify a document, thus the result is more error prone.

4.4 Term Frequency with Truncated SVD

In this experiment, the feature extraction or document representation method used is still term frequency but the dimension reduction algorithm has changed. Instead of reducing the dimension of the feature matrix naively by removing the terms that has the lowest frequency, truncated SVD is used. Truncated SVD would reduce the dimension of the vector space model by retrieving the features with maximum variance in the data.

The number of features shown in the table above is the number of columns in the resulting matrix after truncated SVD is applied.

Similar with the trend in the previous experiment (term frequency with naive dimension reduction), the accuracy of kNN increases slightly, from 0.29 to 0.55 when the features decreases from 6026 to 108, but the resulting accuracy is still far from satisfactory. The time taken by kNN decreases from 809.33s to 7.76s as the number of features decreases.

Table 4.3: Term frequency with truncated SVD

ML	no of features	accuracy	time taken (s)
kNN	6026	0.29	809.33
kNN	4058	0.31	489.86
kNN	2020	0.39	222.31
kNN	1030	0.45	112.66
kNN	508	0.48	58.00
kNN	108	0.55	15.52
kNN	54	0.53	7.76
SVM	6026	0.81	669.01
SVM	4058	0.79	391.88
SVM	2020	0.79	165.82
SVM	1030	0.78	93.1
SVM	508	0.77	78.48
SVM	108	0.65	54.03
SVM	54	0.57	41.11
NN	6026	0.81	370.22
NN	4058	0.80	213.13
NN	2020	0.79	82.52
NN	1030	0.79	54.10
NN	508	0.77	35.85
NN	108	0.69	25.51
NN	54	0.64	23.01

The trend of accuracy increment as the number of features decrease cease when the number of features is reduced to 54. KNN accuracy decrease from 0.55 to 0.53 at that point. The fluctuation of the accuracy would be due to kNN dependency on Euclidean distance between the data points to classify a new data. As the amount of features decrease, the feature matrix becomes less sparse, kNN would need to process less data points thus the speedup. A less sparse matrix resulting from the decrement of features also make kNN classification more accurate and less biased but when the features decrease to an extent where there is no sufficient data to classify a data point correctly, the accuracy dropped.

The accuracy of SVM and NN also have the same trend with the previous experiment, the accuracy decreases slightly when the number of features decreases. Accuracy of SVM decreases from 0.81 to 0.57 and the accuracy of NN decreases from 0.81 to 0.64 as the number of features decreases from 6026 to 54.

As expected, the time taken by the classification model to reduce the dimension and predict the result decreases as the number of features decreases. However, when compare with previous experiments, the time taken in this experiment is astoundingly higher in the case where the features amount to 6026. SVD would be the culprit, dimension reduction comes at a cost which is processing power. To calculate the maximum variance of the features and transform the feature matrix into a smaller dimension would require no small feats of calculation, this would consume both time and processing power.

However, the resulting accuracy is not as good as the experiment with term frequency without any dimension reduction. This is expected because the number of features decreased, the classification would have lesser information to classify a new document correctly. Therefore, it is shown that the reduction in features could reduce the memory space needed to store the feature matrix but it came at the cost of more processing power, longer processing time and most importantly a less accurate result.

Comparing the performance of the classification models in this experiment with the second experiment, term frequency with naive dimension reduction, the performance of the classification models are slightly better with SVD. All the 3 classification models has a slightly higher accuracy across the dimension size when compare with the results of the second experiment. This shows the advantage of SVD over naive dimension reduction. SVD transformed the feature matrix into smaller dimension but retaining the maximum variance or the most prominent feature of the data. Therefore, models trained with SVD should have a better performance than those that trained with naive dimension reduction.

This experiment shows that SVD is a better dimension reduction algorithm than naive dimension reduction method. In this and the previous two experiments, term frequency is applied, it might be too simplistic and the features obtain are not optimum. The next experiments would apply a different document representation algorithm to investigate

further the effect of dimension reduction has on classification model performance and compare the performance of different document representation method.

4.4.1 Graph of Term Frequency with SVD

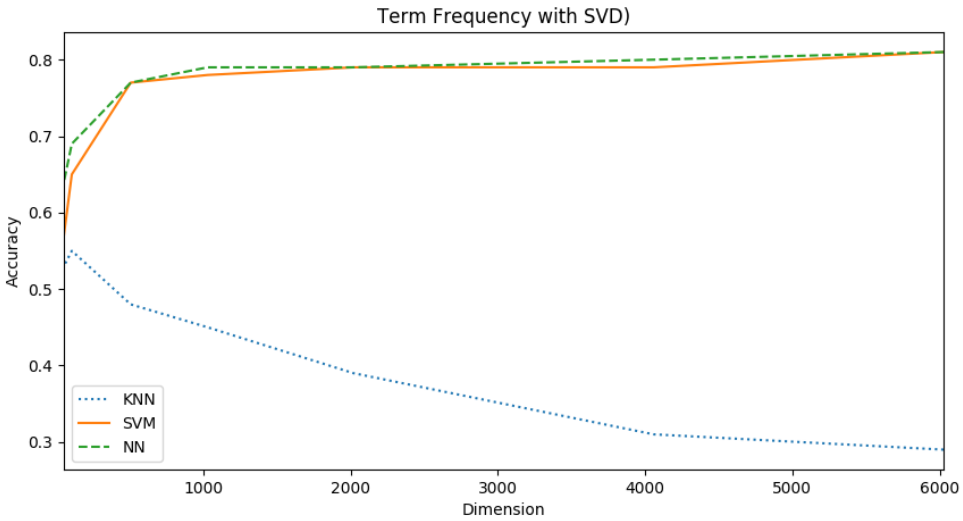

Figure 4.2: Term Frequency with SVD

The figure illustrates what have been described above. This graph show the advantage of SVD has over naive dimension reduction. The accuracy of kNN increases further when the dimension decreases when compare to the experiment of term frequency with naive dimension reduction. At the extreme point, kNN has an accuracy of 0.53 while in the previous experiment, kNN achieve a meager 0.26 with same number of features.

The accuracy of SVM and NN are more stable when SVD is applied. The accuracy of both classification models does not decrease as much when the dimension decreases. Even at the extreme decrement, the accuracy achieved is better.

By comparing the result of these 2 experiments, it is seems that SVD is a better dimension reduction method. The next few experiments would explore another document representation method and determine which of the document representation methods is

more efficient and optimised in extracting features from news articles.

4.5 TF-IDF

Table 4.4: TF-IDF

ML	no of features	accuracy	time taken (s)
kNN	8000	0.76	3.71
SVM	8000	0.87	2.39
NN	8000	0.88	55.82

The previous experiments found that SVD has a slight edge over naive dimension reduction. In this and the next experiments, the effect on dimension reduction is further explored. The performance of the classification models in the previous experiments might be limited by the document representation method used, namely term frequency. It is a simple algorithm and the vector space model generated may not contain enough prominent features for the classification models to classify the documents accurately. Therefore, in this and the next few experiments, a different document representation method is used.

The document representation algorithm applied in this experiment is TF-IDF. In contrast with term frequency which only take the frequency of each word into account, TF-IDF takes both the frequency of each word and its rarity into account. If a term or word appear in high frequency but in many documents, this word may not be of importance and consequently is not a meaningful feature. If a word appear rarely and only in a few documents, this word would have high importance and would be a meaningful feature of the few documents.

With TF-IDF, kNN can achieved a satisfactory accuracy score of 0.76 even though the number of features in the resulting matrix of TF-IDF is the same with term frequency which is 8000. The vector space model of TF-IDF would not be as sparse as that of term frequency which is more suitable for kNN.

NN is still provide the highest accuracy score of 0.88 but the time taken also the longest at 55.82s. NN take the longest time to compute the result in most of the scenarios. This would be due to the number of hidden layers and the number of neurons in the NN. In the experiments, the NN applied has at least 1 hidden layer and the hidden layers would consist of 100 neutrons. Each of these neutron is a processing unit to compute the input feature. Due to the large size of the hidden layer and large size of the feature matrix, NN would take a long time to feed each of the feature through the hidden layers and each of the neutron. The structure of NN caused the long time taken, unless special hardware such as graphic processors or higher power processor is used, the time taken by NN would be higher than other classification models.

SVM achieved an accuracy of 0.87 which is just 0.01 shy of what achieved by NN and the time taken is the lowest among the 3 which is 2.39s. SVM can achieve high accuracy even with high dimension data, because SVM uses the prominent features or support vectors from the data to perform classification, SVM's computational complexity is independent of the dimension of the data. (Rajashekharaiah et al., 2018).

In comparison with the first experiment that apply term frequency document representation without dimension reduction, the performance of the classification models significantly improve. KNN accuracy has more than doubled from 0.31 to 0.76. SVM accuracy increases from 0.81 to 0.87 while accuracy of NN increases from 0.84 to 0.88. Besides accuracy, the time taken also improved, time taken by SVM reduces from 5.27s to 2.39s while time taken by NN reduces from 91.82s to 55.82s. All these improvements are achieved with the same number of features in the vector space, which is 8000.

The performance of the classification models in this experiment is also better than those in the 2 experiments above with dimension reduction. However this may not be a fair comparison because the different number of features are used and dimension reductions

are not applied in this experiment. Dimension reduction algorithms would be applied to the vector space model from TF-IDF in the following experiments in order to have a fair comparison.

The performance increment from term frequency to TF-IDF seems to prove that TF-IDF is a better document representation method and more optimised to extract features from news articles.

4.6 TF-IDF with Naive Dimension Reduction

Table 4.5: TF-IDF with naive dimension reduction

ML	no of features	accuracy	time taken (s)
kNN	6026	0.76	3.76
kNN	4058	0.74	3.89
kNN	2020	0.69	4.00
kNN	1030	0.59	4.35
kNN	508	0.49	4.98
kNN	108	0.08	12.28
kNN	54	0.06	21.20
SVM	6026	0.87	2.49
SVM	4058	0.85	2.45
SVM	2020	0.82	2.50
SVM	1030	0.75	2.64
SVM	508	0.68	2.98
SVM	108	0.41	4.43
SVM	54	0.32	11.63
NN	6026	0.87	48.32
NN	4058	0.85	42.75
NN	2020	0.81	37.77
NN	1030	0.74	48.2
NN	508	0.65	80.45
NN	108	0.45	76.17
NN	54	0.35	73.92

Similar with the experiment with term frequency, naive dimension reduction is applied to the vector space model generated from TF-IDF. The trend over all the 3 machine learning models when the number of features decreases are similar. The accuracy of the

classification models decreases and the time taken increases.

The accuracy achieved by kNN with TF-IDF plus naive dimension reduction is still passable at 0.76 when the features reduced from 8000 to 6026. KNN's accuracy dropped slightly to 0.69 when the number of features decreases to 2020. When the number of features decreases from 2020 to 54, the accuracy of kNN dropped by quite a large margin, from 0.69 to 0.06. The time taken by kNN increases from 3.76s to 21.20s as the features decreases.

SVM has the same behaviour with kNN in this experiment, its accuracy decreases from 0.87 to 0.75 and then to 0.32 when the number of features decreases from 6026 to 1030 to 54. The time taken increases from 2.49s to 11.63s as the number of features decreases.

NN also has the similar trend in accuracy as the number of features decreases. NN's accuracy decreases from 0.87 to 0.74 when the number of features decreases from 6026 to 1030. The time taken increases from 48.32s to 73.92s.

In the scenarios where the number of features are almost halved, from 8000 to 4058, the performance of the classification models are still comparable with the performance achieved with just TF-IDF without any dimension reduction. The accuracy are almost the same, the time taken is similar except NN which has quite a speedup when the number of features is reduced to 4058. This might a sweet spot between dimension reduction and accuracy. At this point, the accuracy is still satisfactory and the time taken is lesser. If NN and dimension reduction is applied in the real world, the dimension of the news articles would have to be at this level. At this level, the classification model is best positioned to reap the reward of dimension reduction, less memory to store the feature matrix, less time is used to compute the result and a comparable accuracy is achieved.

It can be deduced that with naive dimension reduction, the number of features and memory needed to store the vector space model is reduced. With this reduced number of

features, the classification models can still achieve comparable performance at a slightly decreased capacity.

The slight reduction in performance is mainly because of the reduction in dimension. As the amount of information became lesser, less information is available to train a comprehensive model. Therefore, the accuracy of the models decrease. Even though the dimension reduction applied is a naive one and less computing intensive, it still increases the time taken compared with just with TF-IDF. The more reduction is performed, the time taken would increase as well.

4.6.1 Graph of TF-IDF with Naive Dimension Reduction

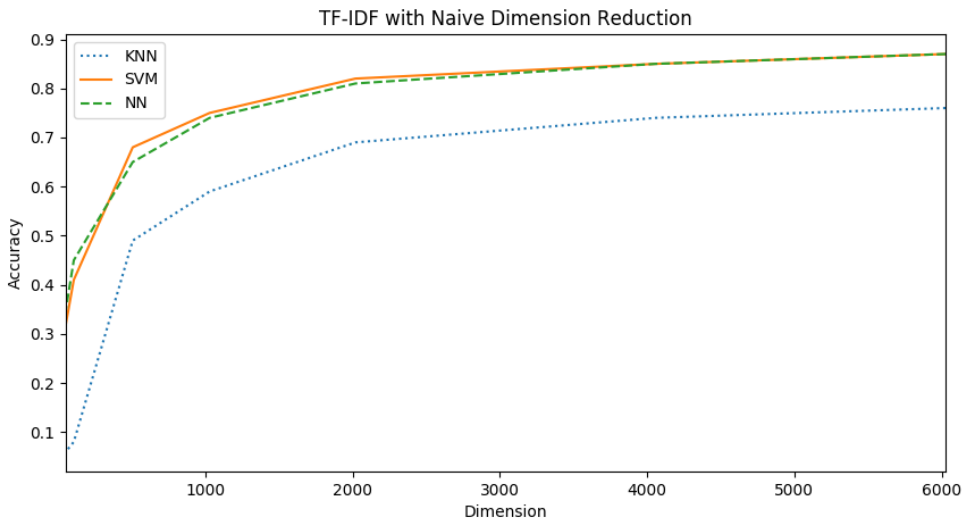

Figure 4.3: TFIDF with Naive Dimension Reduction

The graph above illustrates the results of this experiments. All three classification models have the same decreasing trend in their accuracy as the dimension decreases.

SVM and NN have similar performance with each other. KNN performs much better with tfidf than with term frequency document representation method but its accuracy is still lesser than that of SVM and NN.

4.7 TF-IDF with truncated SVD

Table 4.6: TF-IDF with truncated SVD

ML	no of features	accuracy	time taken (s)
kNN	6026	0.77	833.65
kNN	4058	0.77	487.82
kNN	2020	0.58	221.26
kNN	1030	0.56	111.36
kNN	508	0.54	57.40
kNN	108	0.69	15.48
kNN	54	0.70	8.04
SVM	6026	0.87	374.90
SVM	4058	0.87	194.99
SVM	2020	0.86	72.16
SVM	1030	0.85	35.19
SVM	508	0.83	18.41
SVM	108	0.77	6.40
SVM	54	0.74	4.68
NN	6026	0.86	384.91
NN	4058	0.86	203.26
NN	2020	0.84	86.47
NN	1030	0.82	52.29
NN	508	0.81	46.11
NN	108	0.79	27.63
NN	54	0.78	25.36

In this last experiment, truncated SVD dimension reduction is applied to the resulting vector space model from TF-IDF. Similar with experiment before, each of the classification models would be tested with several set of vector space model, each with different number of features. To put it in perspective, the number of features without reduction is 8000.

When the number of features are at 4058 which is about halved, kNN still can produced an accuracy of 0.77 which is similar with what is achieved with TF-IDF without dimension reduction. The same goes to SVM and NN, at 4058 features, the accuracy are quite similar with TF-IDF without dimension reduction. However, when the dimension of the vector space model is reduced to 2020, the accuracy across the 3 classification models dropped. KNN being the most drastic, its accuracy dropped to 0.58 while SVM and NN dropped to 0.86 and 0.84 respectively, which is slightly worse than before but it is still satisfactory.

Comparing with the results of the experiment with TF-IDF and naive dimension reduction, this performance of the classification models with truncated SVD has a slight advantage. The accuracy is slightly better with truncated SVD than that with naive dimension reduction. This would be due to the advantage of truncated SVD obtaining the maximum variance of the features over naive dimension reduction.

The time taken in this experiment is much higher than the experiment of TF-IDF without dimension reduction and TF-IDF with naive dimension reduction, which is expected. This would be due to SVD reduction takes more time, as more calculations are needed to transform the data. However, the time taken decreased when further reduction is done, kNN take 833.65s to reduce the vector space model from 8000 to 6026 but just 8s to reduce the vector space model from 8000 to 54. Keep in mind that the time recorded here includes the time taken for the classification model to predict the test set as well as the dimension reduction time. This trend appear in SVM and NN as well. SVM took 374.90s to reduce 8000 to 6026 but just 4.64s to reduce 8000 to 54 while NN took 384.91s to reduce 8000 to 6026 and 25.36s to reduce 8000 to 54.

The decrement in time could be due to SVD reduced the features and only return the most prominent features of the news articles. These prominent features would make the differences between the categories of news articles more obvious to the classification models. Thus, the time taken to process the features is also reduced, a speedup. This speedup comes at a cost which is accuracy. For SVM and NN, the reduction in accuracy is slight thus it would be logical to trade the slight accuracy with the speedup but in kNN, the trade off would be lopsided in the favour of time taken.

4.7.1 Graph of TF-IDF with SVD

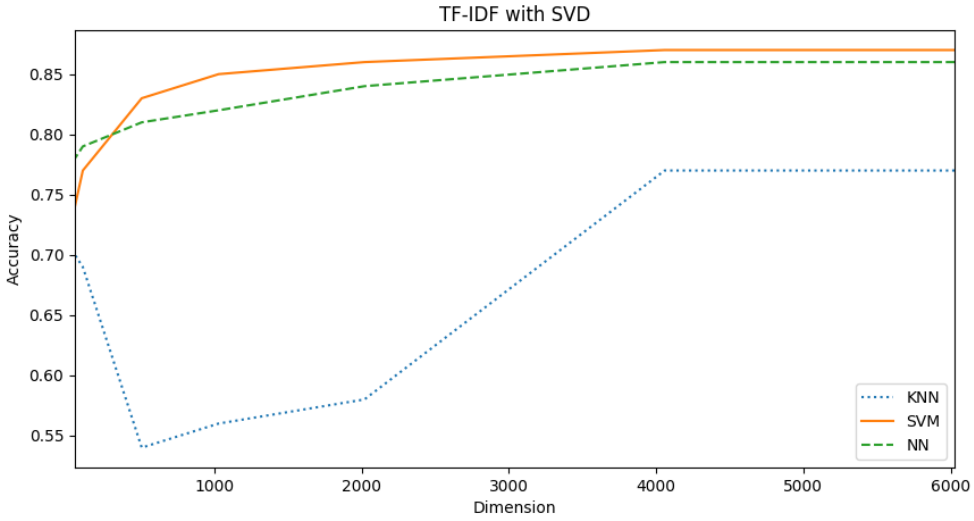

Figure 4.4: TFIDF with SVD

This figure show the fluctuations of the accuracy of the classification models as the dimension decreases. The feature extraction method used here is TF-IDF and the dimension reduction algorithm used is truncated SVD.

KNN accuracy remain stable at around 0.77 as the dimension decrease from 8000 to around 4000. After 4000, further decrement in dimension resulting in a decrease in accuracy. The accuracy of kNN deflected and increase from 0.54 to 0.70 when the dimension reduce from 508 to 54.

Similar to previous result, SVM and NN accuracy remain stable around 0.8 as the dimension reduces from 8000 to 508. When the dimension of the feature matrix is reduced to an extreme of 54 then the accuracy of both SVM and NN deteriorated to around 0.7.

SVM is the best classification model out of the 3, it has slightly better performance than NN in most of the scenarios. It is most resilient to the changes in dimension. This is because SVM depend on the prominent features of the data to classify the data into different classes, the dimension of the data do not has much influence on SVM computational complexity. (Rajashekharaiah et al., 2018). NN has comparable accuracy with SVM but it

would take longer time than SVM to produce the similar result.

4.8 Conclusion

From the results of the 6 experiments above, it is found that TF-IDF is a better document representation algorithm than term frequency. The resulting vector space model from TF-IDF can achieve a higher accuracy than that of term frequency.

The effect of dimension reduction on the accuracy of the classification models is analysed. Dimension reduction, naive and truncated SVD, do reduce the dimension of the vector space model, reducing the memory needed to store the matrix. However, this reduction in features and information would result in a loss of accuracy. truncated SVD would be the better dimension reduction algorithm compared to the naive method because the accuracy achieved with truncated SVD is higher than that of the naive method.

Among the 3 classification models tested in the experiments, SVM is the most efficient and versatile. SVM can achieve high accuracy (> 0.80) in most scenarios. NN can also achieve high accuracy in most of the cases tested but NN is more time consuming. SVM has an advantage over NN on the aspect of processing time. Therefore, SVM would be the most efficient text classification model among the 3 classification models.

CHAPTER 5: CONCLUSION

It is known that TF-IDF should be better than term frequency but Moldagulova and Sulaiman (2018) proposed that with term frequency and naive dimension reduction, kNN classification model would be able to achieve a satisfactory result. (Moldagulova & Sulaiman, 2018). Therefore, this research compare term frequency and TF-IDF and apply different dimension reduction methods to both. The implementation of the kNN algorithm in this study might differ from that proposed by Moldagulova and Sulaiman (2018) which divide a large feature matrix into parts and the classification model is trained by part.

From the results of the experiments, it is shown that TF-IDF is a better document representation method than term frequency. Classification models can achieved a better result with TF-IDF than term frequency. With term frequency as the document representation method, SVM and NN are able to achieve satisfactory accuracy but the accuracy achieved with TF-IDF is slightly higher. The advantage of TF-IDF is more obvious, with kNN, with term frequency the accuracy is a meager 0.31 but with TF-IDF, the accuracy is 0.76 which is close to 0.80.

By applying the 2 types of dimension reduction to both the document representation method, the effect of dimension reduction on the performance of the classification models is analysed. The classification models accuracy from both naive dimension reduction and truncated SVD are quite similar in most cases. However, truncated SVD has a slight edge over naive dimension reduction. Truncated SVD can provide classification models with more prominent features and subsequently higher accuracy.

As the dimension of the vector space model is reduced, the accuracy of the classification models would decreased due to the lack of features or information. This reduction of features would reduce the memory used to store the vector space model. However, the

reduction process would need intensive computing power and time, especially in the case of truncated SVD. This the trade off between dimension reduction and classification accuracy. The gain in less memory to store the matrix would incurred an increment in computing power consumption and time taken.

Out of the 3 classification models tested in this research, SVM has the best overall performance. SVM can achieve a satisfactory accuracy in most scenarios and the time taken for SVM to predict the test data is among the shortest. Thus SVM would be the best classification models out of the 3 classification models studied in text classification. The best combination of all these 3 methods is TF-IDF with SVM without dimension reduction. However, if dimension reduction is needed, SVD would be applied.

For future works, more document representation method, such as n-gram or tree could be applied on the news article aside from TF-IDF. TF-IDF does not take the semantic of the words into consideration, only term frequency and term rarity which is its major drawback. (Qu et al., 2008). N-gram and tree representation method would take the semantic of the word into consideration, this document representation method might be able to extract more features and information from news articles, and subsequently increase the performance of the classification models.

REFERENCES

Ababneh, J., Almanmomani, O., Hadi, W., El-Omari, N., Alibrahim, A. (2014, 02). Vector space models to classify arabic text. *International Journal of Computer Trends and Technology (IJCTT*, *7*, 219-223. doi: 10.14445/22312803/IJCTT-V7P109

Abdi, H., Williams, L. J. (2010). Principal component analysis. *Wiley interdisciplinary reviews: computational statistics*, *2*(4), 433–459.

Allab, K., Labiod, L., Nadif, M. (2017, Jan). A semi-nmf-pca unified framework for data clustering. *IEEE Transactions on Knowledge and Data Engineering*, *29*(1), 2-16. doi: 10.1109/TKDE.2016.2606098

Chen, C.-H. (2017). Improved tfidf in big news retrieval: An empirical study. *Pattern Recognition Letters*, *93*, 113 - 122. Retrieved from http://www.sciencedirect.com/science/article/pii/S0167865516303178 (Pattern Recognition Techniques in Data Mining) doi: https://doi.org/10.1016/j.patrec.2016.11.004

Chen, Y., Zhang, H., Liu, R., Ye, Z., Lin, J. (2019). Experimental explorations on short text topic mining between lda and nmf based schemes. *Knowledge-Based Systems*, *163*, 1 - 13. Retrieved from http://www.sciencedirect.com/science/article/pii/S0950705118304076 doi: https://doi.org/10.1016/j.knosys.2018.08.011

Chien, J.-T. (2019). Chapter 5 - nonnegative matrix factorization. In J.-T. Chien (Ed.), *Source separation and machine learning* (p. 161 - 229). Academic Press. Retrieved from http://www.sciencedirect.com/science/article/pii/B9780128045664000176 doi: https://doi.org/10.1016/B978-0-12-804566-4.00017-6

Dumais, S., Platt, J., Heckerman, D., Sahami, M. (1998). Inductive learning algorithms and representations for text categorization. In *Proceedings of the seventh international conference on information and knowledge management* (pp. 148–155). New York, NY, USA: ACM. Retrieved from http://doi.acm.org/10.1145/288627.288651 doi: 10.1145/288627.288651

Einea, O., Elnagar, A., Debsi, R. A. (2019). Sanad: Single-label arabic news articles dataset for automatic text categorization. *Data in Brief*, *25*, 104076. Retrieved from http://www.sciencedirect.com/science/article/pii/S2352340919304305 doi: https://doi.org/10.1016/j.dib.2019.104076

Hauck, T. (2014). *Scikit-learn cookbook*. Packt Publishing.

Hinton, G. E. (2002, August). Training products of experts by minimizing contrastive divergence. *Neural Comput.*, *14*(8), 1771–1800. Retrieved from http://dx.doi.org/10.1162/089976602760128018 doi: 10.1162/089976602760128018

Karami, A. (2017, Nov). Taming wild high dimensional text data with a fuzzy lash. In *2017 ieee international conference on data mining workshops (icdmw)* (p. 518-522). doi: 10.1109/ICDMW.2017.73

Lahoti, P., Garimella, K., Gionis, A. (2018). Joint non-negative matrix factorization for learning ideological leaning on twitter. In *Proceedings of the eleventh acm international conference on web search and data mining* (pp. 351–359).

Li, T., Ding, C.-c. (2018). Nonnegative matrix factorizations for clustering: A survey. In *Data clustering* (pp. 149–176). Chapman and Hall/CRC.

Ma, J., Yuan, Y. (2019). Dimension reduction of image deep feature using pca. *Journal of Visual Communication and Image Representation*, *63*, 102578. Retrieved from http://www.sciencedirect.com/science/article/pii/S1047320319301932 doi: https://doi.org/10.1016/j.jvcir.2019.102578

Manevitz, L., Yousef, M. (2007). One-class document classification via neural networks. *Neurocomputing*, *70*(7), 1466 - 1481. Retrieved from http://www.sciencedirect.com/science/article/pii/S092523120600261X (Advances in Computational Intelligence and Learning) doi: https://doi.org/10.1016/j.neucom.2006.05.013

Manevitz, L. M., Yousef, M. (2002, March). One-class svms for document classification. *J. Mach. Learn. Res.*, *2*, 139–154. Retrieved from http://dl.acm.org/citation.cfm?id=944790.944808

Mirończuk, M. M., Protasiewicz, J. (2018). A recent overview of the state-of-the-art elements of text classification. *Expert Systems with Applications*, *106*, 36 - 54. Retrieved from http://www.sciencedirect.com/science/article/pii/S095741741830215X doi: https://doi.org/10.1016/j.eswa.2018.03.058

Moldagulova, A., Sulaiman, R. B. (2018, Oct). Document classification based on knn algorithm by term vector space reduction. In *2018 18th international conference on control, automation and systems (iccas)* (p. 387-391).

Narhari, S. A., Shedge, R. (2017, Dec). Text categorization of marathi documents using modified lingo. In *2017 international conference on advances in computing, communication and control (icac3)* (p. 1-5). doi: 10.1109/ICAC3.2017.8318771

Nicolai, G., Kondrak, G. (2016, aug). Leveraging inflection tables for stemming and lemmatization. In *Proceedings of the 54th annual meeting of the association for computational linguistics (volume 1: Long papers)* (pp. 1138–1147). Berlin, Germany: Association for Computational Linguistics. Retrieved from https://www.aclweb.org/anthology/P16-1108 doi: 10.18653/v1/P16-1108

Qu, S., Wang, S., Zou, Y. (2008, Nov). Improvement of text feature selection method based on tfidf. In *2008 international seminar on future information technology and management engineering* (p. 79-81). doi: 10.1109/FITME.2008.25

Rajashekharaiah, K., Chikkalli, S. S., Kumbar, P. K., Babu, P. S. (2018). Unified framework of dimensionality reduction and text categorisation. *International Journal of Engineering & Technology*, *7*(3.29), 648-654.

Ranjan, N. M., Prasad, R. S. (2018). Lfnn: Lion fuzzy neural network-based evolutionary model for text classification using context and sense based features. *Applied Soft Computing*, *71*, 994 - 1008. Retrieved from http://www.sciencedirect.com/science/article/pii/S1568494618304046 doi: https://doi.org/10.1016/j.asoc.2018.07.016

Shinde, S., Joeg, P., Vanjale, S. (2017, 09). Web document classification using support vector machine. In *2017 international conference on current trends in computer, electrical, electronics and communication (ctceec)* (p. 688-691). doi: 10.1109/CTCEEC.2017.8455102

Sweeney, E., Vogelstein, J., Cuzzocreo, J., A Calabresi, P., Reich, D., M Crainiceanu, C., T Shinohara, R. (2014, 04). A comparison of supervised machine learning algorithms and feature vectors for ms lesion segmentation using multimodal structural mri. *PloS one*, *9*, e95753. doi: 10.1371/journal.pone.0095753

Vijayarani, S., Ilamathi, M. J., Nithya, M. (2015). Preprocessing techniques for text mining-an overview. *International Journal of Computer Science & Communication Networks*, *5*(1), 7–16.

Wang, J., Lin, L., Huang, T., Wang, J., He, Z. (2010). Efficient k-nearest neighbor join algorithms for high dimensional sparse data. *arXiv preprint arXiv:1011.2807*.

Wongso, R., Luwinda, F. A., Trisnajaya, B. C., Rusli, O., Rudy. (2017). News article text classification in indonesian language. *Procedia Computer Science*, *116*, 137 - 143. Retrieved from http://www.sciencedirect.com/science/article/pii/S1877050917320872 (Discovery and innovation of computer science technology in artificial intelligence era: The 2nd International Conference on Computer Science and Computational Intelligence (ICCSCI 2017)) doi: https://doi.org/10.1016/j.procs.2017.10.039

Çakir, M. U., Güldamlasioğlu, S. (2016, June). Text mining analysis in turkish language using big data tools. In *2016 ieee 40th annual computer software and applications conference (compsac)* (Vol. 1, p. 614-618). doi: 10.1109/COMPSAC.2016.203

APPENDIX A: DATA PREPROCESSING IMPLEMENTATION

The code in this section show how the 20 news group dataset is processed before it is used to train classification models.

```
"""
Read the raw dataset downloaded from uci repository into a csv
    file in 2 columns, namely text and category
"""
import os
import pandas as pd
import logging
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import WordNetLemmatizer
from nltk.stem import PorterStemmer
from nltk.corpus import wordnet
import time
import re

nltk.download('stopwords')
nltk.download('wordnet')
ps = PorterStemmer()
lemmatizer = WordNetLemmatizer()
logging.basicConfig(level=logging.INFO, format='%(asctime)s - 
    %(message)s')

output_loc = '../output/20newsGroup18828.csv'


def get_files():
    path = '/home/dante/development/datasets/20news-18828/'
    # path = '/home/db/development/datasets/20news-18828/'

    # Getting all the files path
    files = []
    for r, d, f in os.walk(path):
        for file in f:
            files.append(os.path.join(r, file))

    # Read the content of the file
    text = []
    text_class = []
    for f in files:
        logging.info('reading file: %s', f)
        text_class.append(get_category(f))
```

```
        with open(f, 'r', encoding='utf-8',
            errors='backslashreplace') as reader:
            text_str = reader.read()
            text_str = text_str.encode('utf-8').decode('utf-8',
                'replace')
            # text_str = text_str.replace('\n', ' ')
            text.append(pre_process_text(text_str).strip())

    # Create the dataframe from the data read
    list_of_tuple = list(zip(text, text_class))
    df = pd.DataFrame(list_of_tuple, columns=['text', 'category'])
    # Remove rows with empty text cell
    df = df[df.text != '']
    df.to_csv(output_loc, index=False, encoding='utf8')


def get_category(path):
    """
    Obtain the category of the text by getting it's parent
        directory name
    :param path: the file path
    :return: the category
    """
    split_filepath = path.split('/')
    return split_filepath[-2]


def pre_process_text(string):
    prefixes = ['Xref', 'Path', 'From', 'Newsgroup', 'Subject',
        'Summary', 'Keywords', 'Message-ID', 'Date',
                'Expires', 'Followup-to', 'Distribution',
                    'Organization','Approved', 'Supercedes', 'Lines',
                'X-Newsreader', 'References', 'NNTP-Posting-Host',
                    'In-reply-to', 'Sender', 'News-Software',
                'Article-I.D.', 'Article␣I␣D']
    string_list = string.split('\n')
    new_line = []
    for line in string_list:
        if line.startswith(tuple(prefixes)):
            continue
        else:
            # Remove email addresses
            tmp_line = re.sub('\S*@\S*\s?', '', line)
            # Lower the case
            tmp_line = tmp_line.lower()
            # Remove stopwords
            tmp_line = stopwords_removal(tmp_line)
            # Remove all symbols, retain only alphabets and numbers
            # tmp_line = re.sub('[^A-Za-z0-9]+', ' ', tmp_line)
            tmp_line = re.sub('[^A-Za-z]+', '␣', tmp_line)
```

```
            # Remove all the single letter
            tmp_line = re.sub(r"\s+[a-z]{1}[\s+]", "␣", tmp_line)
            tmp_line = re.sub(r"\s+[a-z]{1}$", "", tmp_line)
            tmp_line = re.sub(r"^[a-z]{1}\s+", "", tmp_line)
            # Remove all extra whitespace
            tmp_line = re.sub(r"\n", "␣", tmp_line)
            tmp_line = re.sub(r"\s+", "␣", tmp_line)
            # Stemming or lemmatization
            # tmp_line = stemming(tmp_line)
            tmp_line = lemmatization(tmp_line)
            # Strip the leading and trailing whitespace
            tmp_line = tmp_line.strip()
            new_line.append(tmp_line)
    new_text = '␣'.join(new_line)
    return new_text


def stopwords_removal(words: str) -> str:
    stop_words = set(stopwords.words('english'))
    more_stop_words = ['article', 'writes', 'write', 'say',
       'nntp', 'posting', 'host', 'berkeley', 'edu', 'hmm', 'll',
                       'wo', 't', 'reply', 'think', 'u', 'go', 've',
                          'repost', 'e', 'mail', 're', 'r', 'o',
                          'hey',
                       'hi', 'n', 'dear', 'reader']
    stop_words.update(more_stop_words)
    tokens = word_tokenize(words)
    filtered_words = [w for w in tokens if not w in stop_words]
    return '␣'.join(filtered_words)


def stemming(words: str) -> str:
    tokens = word_tokenize(words)
    stemmed = [ps.stem(word=w) for w in tokens]
    return '␣'.join(stemmed)


def lemmatization(words: str) -> str:
    tokens = word_tokenize(words)
    pos = nltk.pos_tag(tokens)
    lemmatized = [lemmatizer.lemmatize(word=item[0],
       pos=get_wordnet_pos(item[1])) for item in pos]
    return '␣'.join(lemmatized)


def get_wordnet_pos(treebank_tag):
    if treebank_tag.startswith('J'):
        return wordnet.ADJ
    elif treebank_tag.startswith('V'):
        return wordnet.VERB
```

```
    elif treebank_tag.startswith('N'):
        return wordnet.NOUN
    elif treebank_tag.startswith('R'):
        return wordnet.ADV
    else:
        return wordnet.NOUN


def main():
    start_time = time.time()
    get_files()
    logging.info("Time␣taken:␣%.2fs", (time.time() - start_time))


if __name__ == "__main__":
    main()
```

APPENDIX B: DATA EXPLORATORY IMPLEMENTATION

The code below is used to check the distribution of the categories in the dataset.

```
import pandas as pd
import matplotlib.pyplot as plt


def input_data_explore():
    input_df = pd.read_csv('../output/20newsGroup18828.csv')

    print(input_df.shape)

    fig, ax = plt.subplots(figsize=(17, 7))
    agg_df =
       input_df.groupby('category').count().sort_values('category')
    print(agg_df)
    fig_plot = agg_df.plot(kind='bar', ax=ax, title='The␣count␣of␣
       each␣category␣of␣the␣dataset',
               legend=False)
    fig_plot.set_xlabel('Categories')
    fig_plot.set_ylabel('Count')

    plt.tight_layout()
    # plt.autoscale()
    # plt.show()
    plt.savefig('../output/count.png', format='png')


def tf_dimension_effect():
    tf_data = {
        'KNN': [0.28, 0.29, 0.31, 0.39, 0.45, 0.49, 0.52, 0.52,
           0.54, 0.55, 0.53, 0.3],
        'SVM': [0.81, 0.81, 0.8, 0.78, 0.77, 0.77, 0.74, 0.71,
           0.68, 0.64, 0.56, 0.27],
        'NN': [0.84, 0.8, 0.8, 0.79, 0.78, 0.76, 0.73, 0.72, 0.7,
           0.69, 0.63, 0.34]
           }

    df = pd.DataFrame(tf_data, index=[8000, 6000, 4000, 2000,
       1000, 500, 250, 200, 150, 100, 50, 10])
    print(df)
    line_styles = [":", "-", "--"]
    fig, ax = plt.subplots(figsize=(17, 7))
    fig_plot = df.plot(kind='line', ax=ax, title='Effect␣of␣
       dimension␣reduction␣on␣accuracy␣(term␣frequency␣with␣SVD)',
                      legend=True, style=line_styles)
    fig_plot.set_xlabel('Dimension')
```

```
    fig_plot.set_ylabel('Accuracy')
    plt.savefig("../output/tfsvd.png", format='png')
    plt.show()


def tfidf_dimension_effect():
    tfidf_data = {
        'KNN': [0.77, 0.77, 0.77, 0.58, 0.55, 0.55, 0.61, 0.65,
           0.67, 0.69, 0.69, 0.55],
        'SVM': [0.87, 0.87, 0.87, 0.86, 0.84, 0.83, 0.8, 0.8,
           0.78, 0.76, 0.74, 0.5],
        'NN': [0.88, 0.86, 0.85, 0.84, 0.82, 0.8, 0.8, 0.81, 0.8,
           0.8, 0.77, 0.59]
           }

    df = pd.DataFrame(tfidf_data, index=[8000, 6000, 4000, 2000,
       1000, 500, 250, 200, 150, 100, 50, 10])
    print(df)
    line_styles = [":", "-", "--"]
    fig, ax = plt.subplots(figsize=(17, 7))
    fig_plot = df.plot(kind='line', ax=ax, title='Effect␣of␣
       dimension␣reduction␣on␣accuracy␣(TF-IDF␣with␣SVD)',
                       legend=True, style=line_styles)
    fig_plot.set_xlabel('Dimension')
    fig_plot.set_ylabel('Accuracy')
    plt.savefig("../output/tfidfsvd.png", format='png')
    plt.show()


def main():
    # input_data_explore()
    # tf_dimension_effect()
    tfidf_dimension_effect()


if __name__ == '__main__':
    main()
```

APPENDIX C: FEATURE EXTRACTION AND DIMENSION REDUCTION IMPLEMENTATION

The code shown in this section is the code that used to generate the feature matrix and apply the dimension reduction to the feature matrix.

```
"""
Move the feature extraction part out of every script for ease of
   maintenance
"""
from sklearn.feature_extraction.text import CountVectorizer,
   TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.decomposition import NMF, TruncatedSVD,
   LatentDirichletAllocation
from sklearn import preprocessing
from sklearn import pipeline
from sklearn.base import BaseEstimator, TransformerMixin
from numpy import ravel
import pandas as pd
import numpy as np
import scipy
from scipy.sparse import csr_matrix, csc_matrix
import logging


logging.basicConfig(level=logging.INFO, format='%(asctime)s - %
   (message)s')
pd.set_option('display.width', 320)
np.set_printoptions(linewidth=320)
input_file = "../output/20newsGroup18828.csv"


def generate_tfidf():
    df = pd.read_csv(input_file)
    label = preprocessing.LabelEncoder()
    df['category_id'] = label.fit_transform(df.category)

    # Some details about the data
    # print(df.groupby(['category',
       'category_id']).count().sort_values('category_id'))

    # Limit the features
    tfidf = TfidfVectorizer(analyzer='word', stop_words='english',
       token_pattern=r'\w+', max_features=8000, lowercase=True,
                            use_idf=True, smooth_idf=True)
    x = tfidf.fit_transform(df.text)
```

```
    y = df['category_id']

    x_train, x_test, y_train, y_test = train_test_split(x, y,
        test_size=0.2, random_state=42)
    # To make space in memory
    del df
    return x_train, y_train, x_test, y_test


def generate_tfidf_reduced(factor: int):
    df = pd.read_csv(input_file)
    label = preprocessing.LabelEncoder()
    df['category_id'] = label.fit_transform(df.category)

    # Some details about the data
    # print(df.groupby(['category',
        'category_id']).count().sort_values('category_id'))

    # Limit the features
    tfidf = TfidfVectorizer(analyzer='word', stop_words='english',
        max_features=8000, lowercase=True,
                            use_idf=True, smooth_idf=True)
    # tokenizer=r'\w+'
    x = tfidf.fit_transform(df.text)
    y = df['category_id']

    print(x.shape)
    mat = x.tocsc()
    greaterThanOne_cols = np.diff(mat.indptr) > factor
    new_indptr = mat.indptr[np.append(True, greaterThanOne_cols)]
    new_shape = (mat.shape[0],
        np.count_nonzero(greaterThanOne_cols))
    x2 = csc_matrix((mat.data, mat.indices, new_indptr),
        shape=new_shape)
    x2 = x2.tocsr()
    print(x2.shape)

    x_train, x_test, y_train, y_test = train_test_split(x2, y,
        test_size=0.2, random_state=42)
    # To make space in memory
    del df
    return x_train, y_train, x_test, y_test


def generate_tfidf_svd(no_of_features: int):
    df = pd.read_csv(input_file)
    label = preprocessing.LabelEncoder()
    df['category_id'] = label.fit_transform(df.category)

    # Limit the features
```

```
    tfidf = TfidfVectorizer(analyzer='word', stop_words='english',
       token_pattern=r'\w+', max_features=8000,
                           lowercase=True,
                           use_idf=True, smooth_idf=True)
    x = tfidf.fit_transform(df.text)
    y = df['category_id']

    svd = TruncatedSVD(n_components=no_of_features, n_iter=7,
       random_state=42, tol=0.0)
    x_reduced = svd.fit_transform(x)

    x_train, x_test, y_train, y_test = train_test_split(x_reduced,
       y, test_size=0.2, random_state=42)

    return x_train, y_train, x_test, y_test


def generate_tf():
    df = pd.read_csv(input_file)
    label = preprocessing.LabelEncoder()
    df['category_id'] = label.fit_transform(df.category)
    count_vect = CountVectorizer(analyzer='word',
       stop_words='english', max_features=8000, lowercase=True)
    x = count_vect.fit_transform(df.text)
    y = df['category_id']

    x_train, x_test, y_train, y_test = train_test_split(x, y,
       test_size=0.2, random_state=42)

    return x_train, y_train, x_test, y_test


def generate_tf_reduced(factor: int):
    df = pd.read_csv(input_file)
    label = preprocessing.LabelEncoder()
    df['category_id'] = label.fit_transform(df.category)
    count_vect = CountVectorizer(analyzer='word',
       stop_words='english', lowercase=True)
    x = count_vect.fit_transform(df.text)
    y = df['category_id']

    print(x.shape)
    mat = x.tocsc()
    greaterThanOne_cols = np.diff(mat.indptr) > factor
    new_indptr = mat.indptr[np.append(True, greaterThanOne_cols)]
    new_shape = (mat.shape[0],
       np.count_nonzero(greaterThanOne_cols))
    x2 = csc_matrix((mat.data, mat.indices, new_indptr),
       shape=new_shape)
    x2 = x2.tocsr()
```

```
    print(x2.shape)

    x_train, x_test, y_train, y_test = train_test_split(x2, y,
        test_size=0.2, random_state=42)

    return x_train, y_train, x_test, y_test


def generate_tf_svd(no_of_features: int):
    df = pd.read_csv(input_file)
    label = preprocessing.LabelEncoder()
    df['category_id'] = label.fit_transform(df.category)
    count_vect = CountVectorizer(analyzer='word',
        stop_words='english', max_features=8000, lowercase=True)
    x = count_vect.fit_transform(df.text)
    y = df['category_id']

    svd = TruncatedSVD(n_components=no_of_features, n_iter=7,
        random_state=42, tol=0.0)
    x_reduced = svd.fit_transform(x)

    x_train, x_test, y_train, y_test = train_test_split(x_reduced,
        y, test_size=0.2, random_state=42)

    return x_train, y_train, x_test, y_test
```

APPENDIX D: KNN CLASSIFICATION MODEL IMPLEMENTATION

The code for kNN classification model is shown below.

```
from GenTfIdf import generate_tfidf, generate_tf,
    generate_tf_reduced, generate_tfidf_reduced,
    generate_tfidf_svd, \
     generate_tf_svd
from sklearn.metrics import confusion_matrix, accuracy_score,
    classification_report
from sklearn.neighbors import KNeighborsClassifier
import pandas as pd
import numpy as np
import time
import math

pd.set_option('display.width', 320)
np.set_printoptions(linewidth=320)

start_time = time.time()
x_train, y_train, x_test, y_test = generate_tfidf()
# x_train, y_train, x_test, y_test = generate_tfidf_svd(50)
# x_train, y_train, x_test, y_test = generate_tfidf_reduced(50)
# x_train, y_train, x_test, y_test = generate_tf()
# x_train, y_train, x_test, y_test = generate_tf_svd(10)
# x_train, y_train, x_test, y_test = generate_tf_reduced(1000)
print(x_train.shape)

k = int(math.sqrt(x_train.shape[0]))
# Rule of thumb for k is sqrt(sample size)
classifier = KNeighborsClassifier(n_neighbors=k)
classifier.fit(x_train, y_train)

predicted = classifier.predict(x_test)

result = confusion_matrix(y_test, predicted)
print(result)
accuracy = accuracy_score(y_test, predicted)
print("accuracy score: " + accuracy.astype(str))
report = classification_report(y_test, predicted)
print(report)
print('Total time taken: {0:.2f}s'.format(time.time() -
    start_time))
```

APPENDIX E: SVM CLASSIFICATION MODEL IMPLEMENTATION

SVM classification model implementation is shown as follows.

```
from sklearn.metrics import confusion_matrix, accuracy_score,
    classification_report
from sklearn.svm import LinearSVC
import time
from GenTfIdf import generate_tfidf, generate_tf,
    generate_tf_reduced, generate_tfidf_reduced,
    generate_tfidf_svd, \
     generate_tf_svd
import pandas as pd
import numpy as np

pd.set_option('display.width', 320)
np.set_printoptions(linewidth=320)

start_time = time.time()
x_train, y_train, x_test, y_test = generate_tfidf()
# x_train, y_train, x_test, y_test = generate_tfidf_svd(500)
# x_train, y_train, x_test, y_test = generate_tfidf_reduced(500)
# x_train, y_train, x_test, y_test = generate_tf()
# x_train, y_train, x_test, y_test = generate_tf_svd(500)
# x_train, y_train, x_test, y_test = generate_tf_reduced(500)
print(x_train.shape)

clf = LinearSVC(random_state=0, tol=1e-5)
clf.fit(x_train, y_train)

predicted = clf.predict(x_test)
result = confusion_matrix(y_test, predicted)
print(result)
accuracy = accuracy_score(y_test, predicted)
print("accuracy score: " + accuracy.astype(str))
report = classification_report(y_test, predicted)
print(report)
print('Total time taken: {0:.2f}s'.format(time.time() -
    start_time))
```

APPENDIX F: NN CLASSIFICATION MODEL IMPLEMENTATION

NN classification model implementation is shown as follows.

```
from GenTfIdf import generate_tfidf, generate_tf,
    generate_tf_reduced, generate_tfidf_reduced,
    generate_tfidf_svd, \
     generate_tf_svd
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import confusion_matrix, accuracy_score,
    classification_report
import time
import pandas as pd
import numpy as np

pd.set_option('display.width', 320)
np.set_printoptions(linewidth=320)

start_time = time.time()
x_train, y_train, x_test, y_test = generate_tfidf()
# x_train, y_train, x_test, y_test = generate_tfidf_svd(500)
# x_train, y_train, x_test, y_test = generate_tfidf_reduced(500)
# x_train, y_train, x_test, y_test = generate_tf()
# x_train, y_train, x_test, y_test = generate_tf_svd(500)
# x_train, y_train, x_test, y_test = generate_tf_reduced(500)
print(x_train.shape)

# Have to manually interrupt it to produce result
clf = MLPClassifier(tol=1e-3)
clf.fit(x_train, y_train)

predicted = clf.predict(x_test)

result = confusion_matrix(y_test, predicted)
print(result)
accuracy = accuracy_score(y_test, predicted)
print("accuracy score: " + accuracy.astype(str))
report = classification_report(y_test, predicted)
print(report)
print('Total time taken: {0:.2f}s'.format(time.time() -
    start_time))
```