Stemming Text-based Web Page Classification using Machine Learning Algorithms: A Comparison

Ansari Razali[1], Salwani Mohd Daud [2]
Faezehsadat Shahidi[4]
Department of Informatics FTIR
Universiti Teknologi Malaysia
Kuala Lumpur, Malaysia

Nor Azan Mat Zin[3]
Faculty of Information Science & Technology
Universiti Kebangsaan Malaysia
Bangi, Selangor
Malaysia

***Abstract*—The research aim is to determine the effect of word-stemming in web pages classification using different machine learning classifiers, namely Naïve Bayes (NB), k-Nearest Neighbour (k-NN), Support Vector Machine (SVM) and Multilayer Perceptron (MP). Each classifiers' performance is evaluated in term of accuracy and processing time. This research uses BBC dataset that has five predefined categories. The result demonstrates that classifiers' performance is better without word stemming, whereby all classifiers show higher classification accuracy, with the highest accuracy produced by NB and SVM at 97% for F1 score, while NB takes shorter training time than SVM. With word stemming, the effect on training and classification time is negligible, except on Multilayer Perceptron in which word stemming has effectively reduced the training time.**

***Keywords*—Web page classification; stemming; machine learning; Naïve Bayes; k-NN; SVM; multilayer perceptron**

I. Introduction

The fast-growing number of websites in the World Wide Web (WWW) necessitates efficient methodologies to locate information from millions of web pages. Internet has become a huge repository of information and thereby web page documents need to be categorized to facilitate the indexing, searching and web pages retrieval by the search engine [1]. An automation of web pages classification can be achieved by using machine learning. Supervised machine learning algorithms are used for problem that has label and predefined categories. Features that will be the input to the machine learning algorithms is gathered through web data mining; a process of extracting patterns from web pages data [2], which comprises web pages content, hyperlinks or user logs usage.

For this article, only the web pages contents, specifically texts are used as the features–images and audios are discarded. This allows web pages classification to be carried out similar to plain text document classification; words in the web pages are vectorized and become the features that train the classifiers. Pre-processing procedures such as stop word removal and word stemming are commonly conducted before running a machine-learning algorithm to reduce classifiers' processing time by reducing the features in the document. However, word stemmer is known to produce errors to the resulting stemmed words and this may affect classifiers' classification accuracy, which is measured by its precision and recall value [3].

The BBC dataset used in this research consists of news articles that are predefined and labeled based on five categories. Machine learning algorithms are used to extract and learn prominent features that defines each category so that future articles can be classified automatically. In webpages searches, the speed of classification process is an important factor that affects user experience. While this is important, faster processing should not be justified on the expense of classification accuracy. Thereby, this research evaluates word stemming procedure on the classification speed and accuracy, using different machine learning algorithms.

II. Web Page Classification

Web page classification, or also called web page categorization, is defined as a task to determine the category of a web page. In a formal definition, let C = {c1,…ck} as predefined categories, D = {d1,…dk} as web pages and A = D x C as a decision matrix (Table I).

whereby each entry a_{ij}, $(1 <i<N, 1<j<K)$ indicates whether web page di is in category c_j. Each $a_{ij} \epsilon \{0,1\}$; 1 is when a web page di fits category c_j, 0 when it is not in c_j. A web page can be fitted in one category, multiple categories or none of the categories. The objective of web page classification is to estimate the unknown assignment function f: D x C → {0,1} by means of a learned function f': D x C → {0,1}, which is either a classifier, model or hypothesis, such that f' coincides with f to maximum extent. The learned function f' is derived from performing machine learning over a training data which consists of web pages that are labeled with their assigned categories. The trained function f' will then be used to classify unseen data of web pages to its categories [23].

TABLE. I. Decision Matrix

Web Pages	Categories				
	C_1	…	C_j	…	C_k
d_1	a_{11}	…	a_{1j}	…	a_{1k}
…	…	…	…	…	…
d_i	a_{i1}	…	a_{ij}	…	a_{ik}
…	…	…	…	…	…
d_n	a_{n1}	…	a_{nj}	…	a_{nk}

Web classification is almost similar to text classification, but with additional steps because of special characteristics in web pages:

- Web pages are semi-structured documents commonly written in HTML that has information enclosed between tags.
- Web pages have topological information about the link graph which shows hyperlinks information with the linked web pages.

In web page classification, there are multitude of potential inputs that can be used by classifier, such as URL of the web page, HTML tags frequency, the content of the tags and so on.

III. Related Works

Previous works on these algorithms use Naïve Bayes to classify 4,887 website homepage contents into 10 categories yielding 89% accuracy [4], k-NN shows higher accuracy as compared to Naïve Bayes for text and document classification despite showing low performance in terms of its fully dependency on every sample in the training set [5]-[6], SVM performed better than Naïve Bayes in classifying health and non-health related websites [3], Naïve Bayes trumps k-NN and SVM when classification is carried out to predict users' personality based on Twitter texts [5], and automatic web page categorization on educational based corpus is conducted using seven classifiers, with high accuracy classifiers demonstrated by Linear SVM, Logistic Regression, Multinomial Naïve Bayes, and Multilayer Perceptron. Decision Tree is the worst performed while k-NN is moderate.

An approach was used by extracting information from both web pages contents and links structure as inputs to SVM and neural network [7]. An improved k-NN classifier uses new feature weighting and new distance weighted voting scheme [8], and an improvement is suggested on the k-NN to adopt density-based approach to manage unevenly distributed dataset. The distance between k-NN and test data are adjusted based on their difference of density [9].

The effect of word stemming to the performance of text classification is arguable. A performed system should acquire high number of relevant documents (high recall) and only a few non-relevant documents (high precision). An evaluation of Porter stemming based on information retrieval from a corpus of 400 MEDLINE (Medical Literature, Analysis and Retrieval System Online) shows improvement of precision and recall as compared to information retrieval without using stemmer [10].

Researchers [11] argue that stemming has little impact on the performance of text classification. Schofield *et al* [12] have conducted experimental procedure to validate the outcome of various stemmers on different type of text corpus. The study concludes that generally stemmer yield no meaningful improvement in likelihood and coherence and can even degrade topic stability. The researcher claims that Porter stemmer for instance just reduce the possible unigrams that can be generated and does not appear to improve the model quality. Statistical approach of stemming does not need to have built in set of morphological rules as in rule-based approach; it learns the rule by training on a well-formed corpus. Thus, it overcomes Porter's error. Nonetheless, statistical approach has shortcomings such as dependency on corpus size and quality, higher execution time and high storage use [13]-[14]. In this report, we examine the effect of stemming to classifiers performance.

Previous works recorded extensive discussions on web pages classification using various types of machine learning algorithms. Many literatures however focus solely on the classification accuracy and does not include the processing time in the results. Additionally, even less literature records the difference in the accuracy and processing time with and without word stemming.

IV. Experiments

A. Dataset

This study uses dataset that originates from BBC News website articles gathered by [15]. It consists of 2,225 documents that corresponds to articles on five topical areas published on the BBC News website from year 2004 to 2005. The articles are labeled based on the topics, namely 'business', 'politics', 'entertainment', 'sport' and 'tech'. The BBC dataset consists of 2,225 documents and is split randomly into training and test dataset with the ratio of 80:20, which is a common ratio used for this purpose [16]. After splitting, there are 1,780 documents in training and 445 in test dataset. The frequency distribution for the dataset before and after splitting is shown by Table II.

This dataset comes in the form of raw text files and separated into five different folders based on their respective topics. Data in these text files need to be collated in a spreadsheet to enable further analysis and processes. Python codes are used to combine all the data and subsequently exported into a comma-separated values (.csv) spreadsheet file. A column named as 'newstype' is created to indicate the news category while column 'news' stores the news article.

B. Pre-Processing

The number of rows of a matrix corresponds to the number of words in a document collection. There can be hundreds of thousands of different words in the document collection. Pre-processing is an effort to reduce these words, which is the input feature for the machine learning classifiers, by cleaning up the document, selecting and extracting feature word and perform word stemming. These processes can improve computational efficiency and classification effectiveness.

1) Cleaning up document: Text documents are broken down into individual words through tokenization. Tokenization is commonly followed with other pre-processing steps such as removing stop words, punctuation, special characters and word stemming. These individual words will be selected and extracted to become features that represent respective categories or labels. These words or features serve as inputs to machine learning classifiers. After pre-processing, there are a total of 389,548 features extracted from the whole document collection. The process of cleaning text is important to remove unnecessary and non-important elements of sentences as show in Fig. 1.

TABLE. II. FREQUENCY DISTRIBUTION

News type	Number of documents (Before splitting)	Number of documents (After splitting)
Business	510	399
Entertainment	386	312
Politics	417	331
Sport	511	403
Tech	401	335

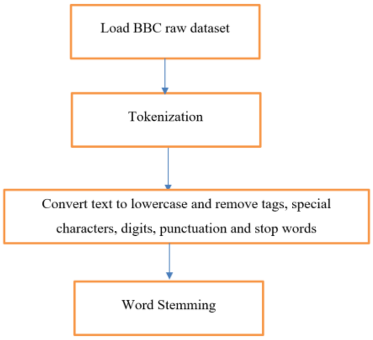
Load BBC raw dataset
Tokenization
Convert text to lowercase and remove tags, special characters, digits, punctuation and stop words
Word Stemming

Fig. 1. Cleaning up Process of Text Document.

2) *Feature extraction and selection:* Feature selection and extraction is a process to reduce noise terms that are not related to the categories of training document. To reduce these noise terms, feature selection is first conducted to extract index terms (features) that will become the predictor to evaluate and assign unseen documents as belonging to the appropriate category. Since feature selection reduces noise terms, in effect it reduces vector dimensions and thus enable classifiers to produce faster results. Training documents of similar categories are represented with the same term vectors, thus they can be closely located in term vector space.

For the first step in text categorization, we need to transform the documents consisting of strings of characters into a representation that is suitable for the learning algorithms and the classification tasks. And the most commonly used document representation is Vector Space Model (VSM), that is, each document is represented by a vector of words. A word-by-document matrix A is used for a collection of documents, with each entry represents the occurrence of a word in a document, that is, A=(a_ij), where (a_ij) is the weight of word i in document j.

The weight value of each term can be computed by different weighted schemes namely Boolean value, Term Frequency (TF), Inverse Document Frequency (IDF), Term Frequency and Inverse Document Frequency (TFIDF) [17]. The simplest approach of determining the weight is Boolean weighting, which sets the weight (a_ij) to 1 if the word occurs in the document and 0 otherwise. TF weighted scheme counts the words that are most frequently occurring as shown by Fig. 2 which summarize the word count frequency from each of the five categories in the BBC dataset.

3) *Word stemming:* Stemming is a feature term reduction technique that is used by removing suffixes such as 'ed', 'ing' and 'ily'. It reduces complexity and enable more efficient information retrieval especially in data mining applications. Nonetheless, stemming may create non-real words as the stemmer does not check on grammatical rules during the stemming process [18]. Lemmatization is an alternative that checks on canonical forms of the words, but it is computationally expensive and thus takes up more processing time [19]. Porter stemmer is one of the most widely used stemmer. Other types of stemmer include Lovins, Lancaster and Porter2, which is also referred as Snowball [12]. Porter Stemmer algorithm as shown in Fig. 2 is commonly used in text classification. It is based on steps by which each step removes a type of suffix by using substitution rules. Non-real words such as 'studi' is a grouping stemmed words that resulted from words that comes from a similar root namely 'studied, studies, study, studying' [12],[20],[21].

Feature selection and extraction through applying TF-IDF and Porter Stemming are able to reduce dimensionality by trimming down the number of features from 389,548 to 233,123 features.

C. *Classification*

There are 1,780 documents in training and 445 in test dataset. The training dataset is used to train the machine learning classifiers whereby the classifier learns the characteristics of the dataset and the relations between these documents and their predefined categories. Once the classifier is fully trained, the test dataset is fed into the classifier as the input and it will output the predicted categories for each document. We compare few machine algorithms for the classification stage, i.e. Naïve Bayes, k-NN, Support Vector Machine (SVM), and Multilayer perceptron.

1) *Naïve bayes:* Naïve Bayes uses vector analysis method that is based on the concept of conditional probability or Bayes theorem to measure documents relevancy. The probability of class a to be the category for document b is given by:

$$P(a|b) = \frac{P(a)P(b|a)}{p(b)} \qquad (1)$$

From the training dataset, the classifier calculates the probability value of each feature term belonging to certain category. It is based on the fraction of time a term appears among all terms in documents of a category. The sum of probabilities for each category of each term occurring in the document is calculated that enables the classification of a new document. This classifier is called 'naïve' because each term is assumed to occur independently from each other [6].

There are several types of Naïve Bayes, among them are Multinomial, Boolean and Bernoulli. For text classification, Multinomial Naïve Bayes is mostly used due to its computational efficiency and relatively good predictive performance. It uses multinomial distribution with the classification features consisting of the number of word occurrences or the weight of the word [22]. In this project, the weight of the word is used which is obtained from TF-IDF.

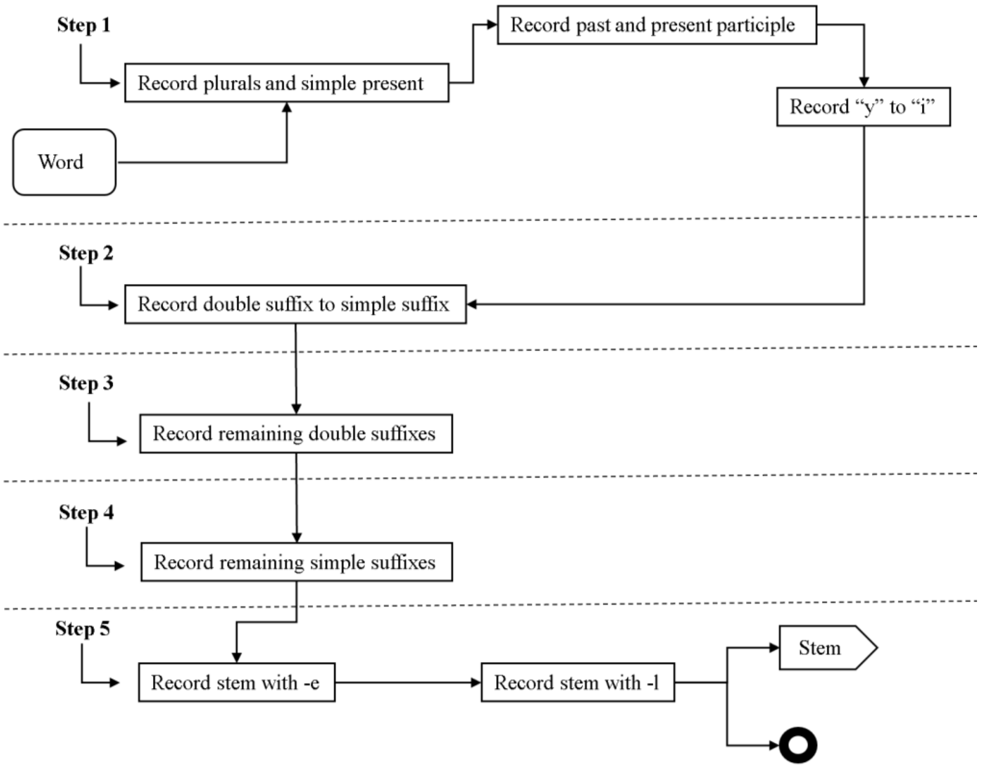
Step 1
Record plurals and simple present
Record past and present participle
Record "y" to "i"
Word
Step 2
Record double suffix to simple suffix
Step 3
Record remaining double suffixes
Step 4
Record remaining simple suffixes
Step 5
Record stem with -e
Record stem with -l
Stem

Fig. 2. Five Steps of Porter Stemmer [10].

2) k-Nearest Neighbor (k-NN): is a non-linear lazy learning classifier that delays learning process until a new document appear to be classified. It compares the new document directly with the training documents and computes their similarity score by measuring the distance between the documents by using Euclidean distance or cosine similarity. Euclidean distance is used in this project as it the most widely used distance metrics in k-NN classification. K-NN use the similarity score to rank the document's neighbors among the training document vectors. The k-nearest neighbors are used to predict the category of the new document [23].

3) Support Vector Machine (SVM): Support Vector Machine (SVM) is a powerful technique for classification. The state-of-the-art in text classification usually applies machine learning techniques such as SVM [24]. However, SVM is not suitable for large datasets or text corpora, because the training complexity of SVM is highly dependent on the input size [25]. Comparing the processing time, SVM takes longer time than NB and k-NN during classifier training but is faster than k-NN during classification. SVM is a universal learner. It is a linear learner in its basic form, but can be configured to learn polynomial classifiers, radial basic function (RBF) networks and three-layer sigmoid neural nets by applying appropriate kernel function. In a classification procedure carried out on big text corpora, [25] concluded that RBF and Sigmoid kernels need higher time to build model and requires additional parameters as compared to linear SVM. It is difficult to determine its parameterization with imbalanced data.

4) Multilayer Perceptron Multilayer Perceptron (MLP): also known as Artificial Neural Networks (ANN), is a multilayer, feed-forward neural network that contains nodes at the input layer, hidden layer and output layer. Having these multi-layers allows MLP to learn non-linear functions. Neurons in the hidden layer and output layers have biases that acts as weight. The purpose of learning is to assign the right weights to these edges that minimize the cost function. By entering vectors, these weights can determine the output vector.

MLP trains using backpropagation error method. Backpropagation error is a supervised learning method that computes the error at the output and used by gradient descent optimization to adjust the edge weights by calculating the gradient of the loss function. This adjustment is repeated iteratively, and iteration ends when the output error is below the established standard.

The problem with training MLP is to minimize error function E which is defined by the sum of square differences over all data in the training set. A simplified equation for error function E for an MLP with weights n is given as:

$$\min_{w \in \mathbb{R}^n} E(W) \qquad (2)$$

with w∈R^n is column weight vector with components w_1, ω_2,… ω_n. There are various approaches to improve the efficiency of error minimization process, and one of the common methods used for text classification is quasi-Newton, which uses second order derivative related information [26].

Activation functions is an important element in ANN. Its purpose is to convert input signal of a node into non-linear property before channeling the signal to output signal, which then will be the input to the next layer in the stack. Non-linearity enables ANN in modelling complicated, high dimensional and not linearly separable big dataset. There are various types of activation functions, among mostly used are sigmoid, tanh and Rectified Linear Units (ReLU). Activation functions using sigmoid and tanh is less suitable for learning because its small derivatives can lead to vanishing gradient; when the neuron's activation saturates at either tail of 0 or 1, the gradient at these regions is almost zero. This will cause very slow or no learning during backpropagation as the weights are updated with small values. In this respect, ReLU function is less susceptible to this vanishing gradient issue because it has an identity derivative in the positive region [27]. Any negative elements are set to '0'; with no exponentials, multiplication nor division operations. Its gradient computation is simple and, in this way ReLU can speeds up neural networks training.

Although the number of hidden layers and nodes are an important determinant in ANN performance and processing time, there is no standard method on their selection. One of the method applicable is try and error approach [28]. Hidden layer size is arbitrarily selected, and the outcome is observed.

V. Results and Discussions

Table III summarize the results obtained from the experiments done for word stemming and without word stemming. Dataset is classified using Multinomial Naïve Bayes with the best classification outcome is achieved by using parameter Laplace smoothing and without word stemming; F1 score of 0.97, training time of 0.02 seconds and classification time of 0.78 seconds.

The best classification outcome for k-NN is achieved by using parameter k = 31 and without word stemming; F1 score of 0.96, training time of 0.007 seconds and classification time of 0.91 seconds.

Linear SVM without word stemming provides the best score with F1 of 0.97, training time of 11.37 seconds and classification time of 3.14 seconds. RBF and Sigmoid by far performed worse than linear SVM.

ANN classification is carried out using ReLU as an activation function. Generally, word stemming results in lower F1 score but reduces the training time. ANN with 3 layer and each layer containing 1,000 nodes shows the best F1 but long training time of 716.95 seconds. A more balanced 2-layer ANN with 50 nodes each takes only 15 seconds of training time and 0.73 seconds of classification time. This is taken as the best classification parameter and outcome for ANN. Another observation is ANN with three hidden layers performs no better than with one hidden layer. It does however introduce complexity and extends the training time.

Based on the Table IV and Fig. 3, in terms of classifiers' performance in classification, generally all of the classifiers perform at a high F1 score. The difference is marginally very low between the classifiers. Naïve Bayes and SVM each score 0.97 while k-NN and Multilayer Perceptron each get 0.96. All of the classifiers performed better with higher F1 score without word stemming. The effect of stemming on the training and classification time (as shown in Fig. 4) is negligible on all classifiers, except for Multilayer Perceptron. Stemming effectively reduced training time in Multilayer Perceptron modelling phase.

TABLE. III. Most Common Words in Each Category

Category	Politics	Tech	Business	Entertainment	Sport
	Party	People	Company	Film	Win
	Labour	Game	Firm	Award	Game
	Government	Mobile	Market	Star	Play
	Election	Technology	Rise	Music	Time
	People	Phone	Sale	Win	Player
	Blair	Service	Bank	Band	England
	Minister	User	Share	Actor	Match
	Tory	Firm	Economy	Director	Team
	Plan	Music	Price	Oscar	Final
	Brown	Software	Growth	Album	Club

TABLE. IV. Classifier's Best Performance Comparison

Classifier	Stemming	Precision	Recall	F1	Training time (seconds)	Classification time (seconds)
Naïve Bayes	No	0.97	0.97	0.97	0.02	0.78
k-NN	No	0.96	0.95	0.96	0.007	0.91
SVM	No	0.98	0.97	0.97	11.37	3.14
Multilayer Perceptron	No	0.97	0.96	0.96	18.17	0.73
Naïve Bayes	Yes	0.94	0.95	0.95	0.02	0.70
k-NN	Yes	0.92	0.92	0.92	0.007	0.98
SVM	Yes	0.95	0.95	0.95	10.23	2.3
Multilayer Perceptron	Yes	0.94	0.93	0.94	15.00	0.60

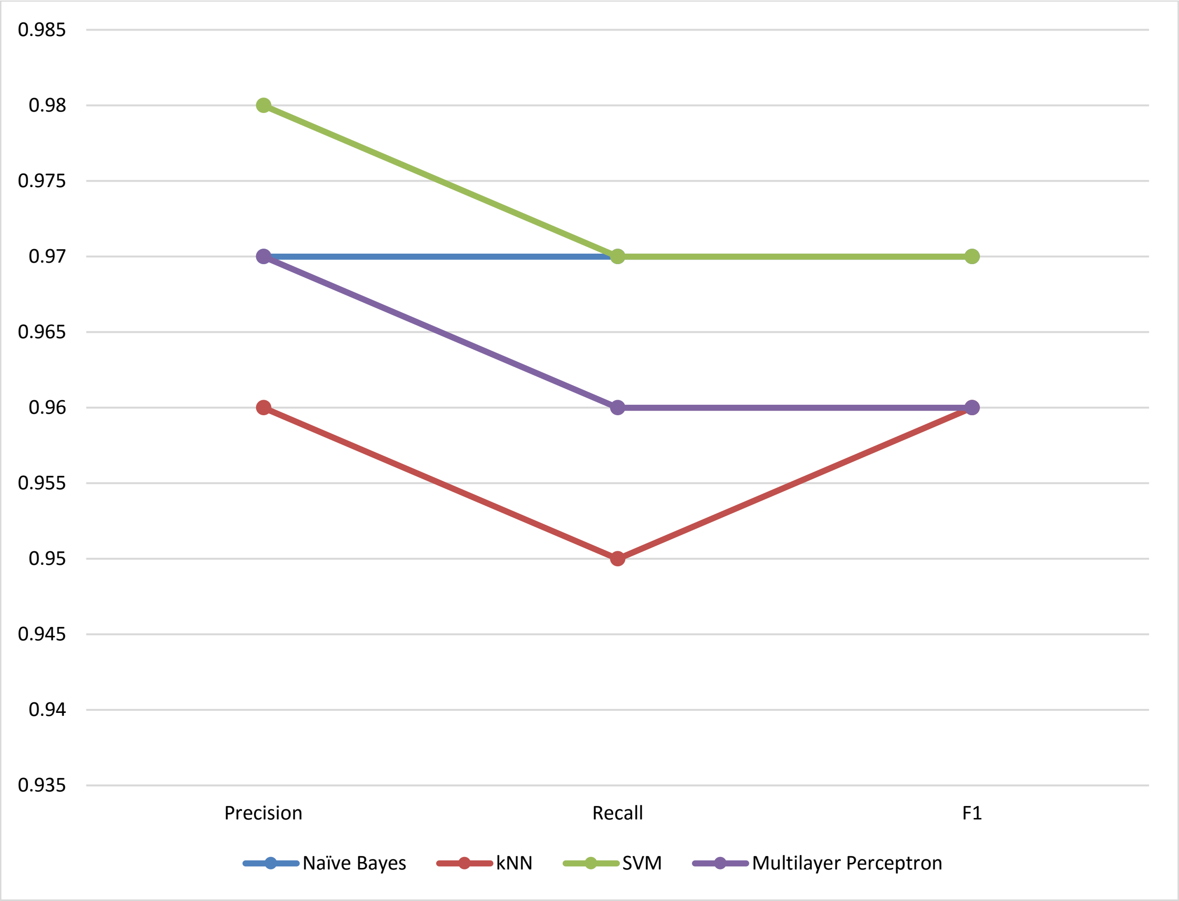

Fig. 3. Classifier Accuracy Comparison.

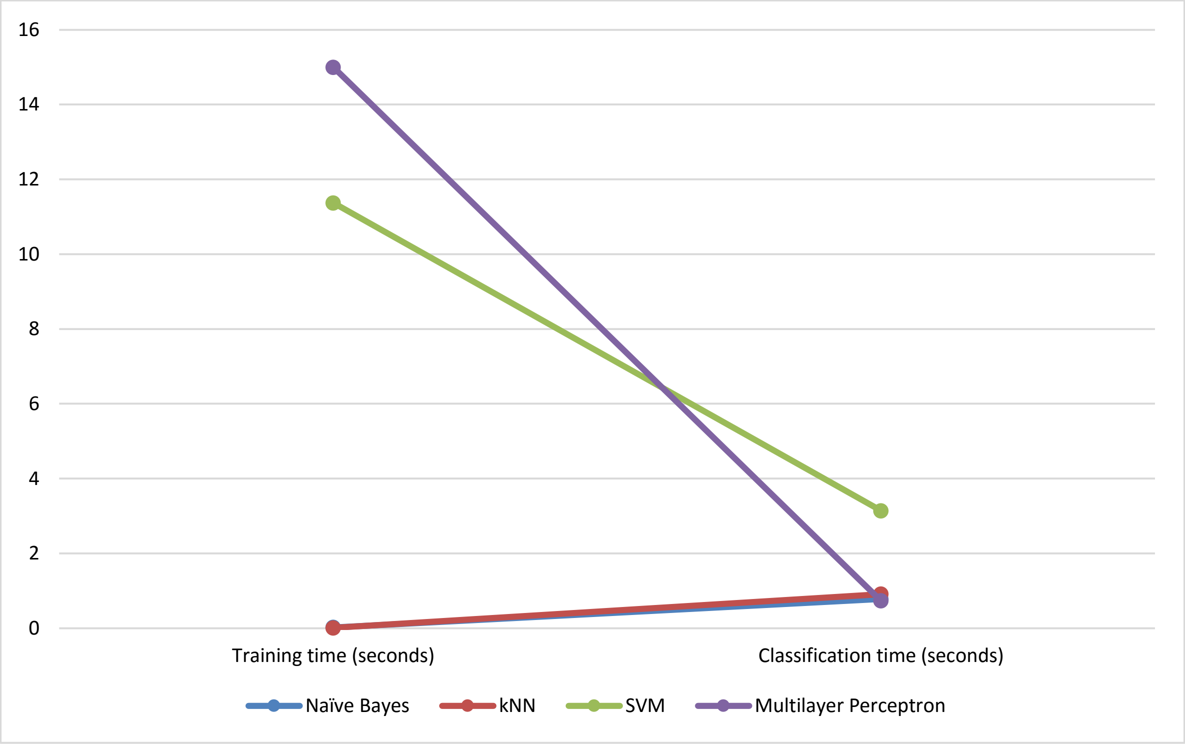

Fig. 4. Classifier Training and Classification Time Comparison.

VI. Conclusion

All of the classifiers produce respectable accuracy with very low marginal differences, which is not decisive. A larger dataset with larger number of categories may present higher complexity and dimensionality to the classifiers and perhaps with such challenges there will be a distinct best performer. This study exclusively uses Porter stemmer to perform word stemming. Future works may attempt the use of other word stemmer or lemmatization. Lemmatization may extend the processing time as it checks on canonical form of words. Nonetheless, it is interesting to observe on whether a correct grammar and real words produced from lemmatization will increase, instead of reducing, classifiers' accuracy as observed with Porter stemmer in this study.

Acknowledgment

The authors would like to express greatest appreciation to Ministry of Education, Malaysia and Universiti Teknologi Malaysia (UTM) for the financial support (grant number: Q.K130000.2538.19H12). Last but not least, we also appreciate Advanced Informatics at Razak Faculty of Technology and Informatics (FTIR), UTM for their support in conducting this research.

References

[1] A. Videira and N. Goncalves, "Automatic Web Page Classification Using Visual Content." in 10th Int. Conf. on Web Information Systems and Technologies, WEBIST 2014, Barcelona, Spain, April 2014, pp.279-294.

[2] M. Allahyari, S. Safaei, S. Pouriyeh, E.D. Trippe, K. Kochut, M. Assefi, and J.B. Gutierrez, "A Brief Survey of Text Mining: Classification, Clustering and Extraction Techniques", J. ArXiv, vol. abs/1707.02919, August 2017.

[3] A. Siddiqui, M. Adnan, R. A. Siddiqui and T. Mubeen, "A comparative study of web pages classification methods applied to health consumer web pages," in 2nd Int. Con. on Computing Technology and Information Management, ICCTIM 2015, Johor, Malaysia, 2015, pp. 43-48.

[4] A.S. Patil and B.V. Pawar, "Automated Classification of Web Sites using Naive Bayesian Algorithm," in Int. MultiConference of Engineers and Computer Scientists, IMECS 2012, Hong Kong, March 2012, vol. 1, pp. 14-16.

[5] A. Moldagulova and R. Sulaiman, "Using KNN Algorithm for Classification of Textual Documents," in 8th Int. Conf. on Information Technology, ICIT 2017, Amman, Jordan, May 2017, pp. 665-671.

[6] V. Bijalwan, V. Kumar, P. Kumari, and J. Pascual, "KNN based Machine Learning Approach for Text and Document Mining", Int. J. of Database Theory and Application, vol. 7(1), pp. 61–70, 2014.

[7] M. Chau and H. Chen, "A Machine Learning Approach to Web Page Filtering Using Content and Structure Analysis," Decision Support Systems, vol. 44, pp. 482–494, 2008.

[8] J. Alamelu Mangai, Satej Milind Wagle, and V. Santhosh Kumar, "A Novel Web Page Classification Model using an Improved k Nearest Neighbor Algorithm" in 3rd Int. Conf. on Intelligent Computational Systems, ICICS 2013, Singapore, April 2013, pp. 50-53.

[9] K. Shi, L. Li, H. Liu, J. He, N. Zhang and W. Song, "An improved KNN text classification algorithm based on density," in IEEE Int. Conf. on Cloud Computing and Intelligence Systems, Beijing, 2011, pp. 113-117.

[10] W. Abdessalem, "A New Stemmer to Improve Information Retrieval," Int. J. Network Security and Its Applications (IJNSA), vol. 5, pp. 143-154, Jul 2013.

[11] M. Toman, R. Tesar, and K. Jezek, "Influence of Word Normalization on Text Classification," in InSciT, Oct 2006, pp. 354-358.

[12] A. Schofield and D. Mimno, "Comparing Apples to Apple: The Effects of Stemmers on Topic Models," Transactions of the Association for Computational Linguistics, vol. 4, pp. 287-300, 2016.

[13] X. Jinxi, W.C. Bruce, "Corpus-based Stemming Using Co-occurrence of Word Variants," ACM Transactions on Information Systems, vol. 16(1), pp. 61-81, 1998.

[14] D. Patel, M. Patel, and Y. Dangar, "A Survey of Different Stemming Algorithm," Int. J. of Advance Engineering and Research Development, vol. 2(6), pp. 50-53, 2015.

[15] D. Greene and P. Cunningham, "Practical Solutions to the Problem of Diagonal Dominance in Kernel Document Clustering," in 23rd Int. Conf. on Machine Learning, ACM, Jun 2006, pp. 377-384.

[16] E. Setiani and W. Ce, "Text Classification Services Using Naïve Bayes for Bahasa Indonesia," in Int. Conf. on Information Management and Technology, ICIMTech, Jakarta, Indonesia, 2018, pp. 361-366.

[17] Singh, K. Nareshkumar, Devi, H. Mamata, Mahanta, A. Kakoti, "Document Representation Techniques and Their Effect on Document Clustering and Classification: A Review," Int. J. Advanced Research in Computer Science, vol. 8(5), pp. 1781-1784, 2017.

[18] Q.S. Mahdi, K.H. Qadir, and R.L. Falih, "Web Page Classification by Using Neural Networks," Zanco J. Pure Applied Sciences, vol. 23, 2011.

[19] S. Raschka, "Naïve Bayes and Text Classification I – Introduction and Theory." J. ArXiv, 2014.

[20] C. Naik and V. Kothari, "Document Classification using Neural Networks Based on Words," Int. J. Advanced Research in Computer Science, vol. 6(2), pp. 183-188, 2015.

[21] B.L. Devi and A. Sankar, "Feature Selection for Web Page Classification Using Swarm Optimization," Int. J. of Computer and Information Engineering, vol. 9(1), pp. 340-346, 2015.

[22] B.Y. Pratama and R. Sarno, "Personality Classification Based on Twitter Text Using Naïve Bayes, KNN and SVM," in Int. Conf. on Data and Software Engineering, Yogyakarta, Indonesia, 2015, pp. 170-174.

[23] B. Choi and Z. Yao, "Web Page Classification," Foundations and Advances in Data Mining, Springer, 2005.

[24] J.H. Wang and H.Y. Wang, "Incremental Neural Network Construction for Text Classification," in Int. Symposium on Computer, Consumer and Control, Taichung, Taiwan, 2014, pp. 970-973.

[25] R. Romero, E. L Iglesias, and L. Borrajo, "A Linear-RBF Multikernel SVM to Classify Big Text Corpora," BioMed Research Int. 2015.

[26] I.E. Livieris, "Improving the Classificaiton Efficiency of an ANN Utilizing a New Training Methodology," Informatics, vol. 6, p.1, 2019.

[27] S. Eger, P. Youssef, I. Gurevych, "Is it Time to Swish? Comparing Deep Learning Activation Functions Across NLP Tasks," in Con. on Empirical Methods in Natural Language Processing, Brussels, Belgium, 2018, pp. 4415-4424.

[28] F.S. Panchal, and M. Panchal, "Review on Methods of Selecting Number of Hidden Nodes in Artificial Neural Network," Int. J. Computer Science and Mobile Computing, IJCSMC, November 2014, vol. 3, pp. 455 – 464.