Turkish Online Journal of Qualitative Inquiry (TOJQI)
Volume 12, Issue 7, July 2021: 8216 - 8225

Research Article

A Comparative Study on Classification Algorithms Using Different Feature Extraction And Vectorization Techniques For Text

Nadia Anjum[a], Dr Srinivasu Badugu[b]

[a]Assistant Professor, Dept of CSE, Stanley College of Engineering and Technology,
[b]Professor, Dept of CSE, Stanley College of Engineering and Technology

***Corresponding author:** [a]nadiaanjum@stanley.edu.in, [b]drsrinivasu@stanley.edu.in

Abstract

We live in a world where information has a great value and the amount of information available in the text document has risen so that identifying those that are important to us becomes an issue. Because of this data, divided into categories, the user is able to navigate to the information he wants to obtain. Texts are most of the data and here text classification comes to the scene.The aim of this paper is to classify the documents automatically into their classes by comparing different feature extraction and Vectorization techniques. Classification of document requires machine learning (ML) techniques. The ML techniques that we have employed to classify the documents are Support Vector Machine (SVM), Naïve Bayes (NB). The various feature extraction techniques that we have implemented are Stemming and Lemmatization and we note how the algorithms differ in performance when implemented each of the feature extraction technique and vectorization approaches. We used two vectorization techniques, such as vectorization of count vector and vectorization of term frequency inverse document frequency (TF-IDF). The results prove that according to the type of content and metric, the performance of the feature extraction and vectorization methods are contrasting; in some cases are better than the others, and in other cases is the inverse..

Keywords: Machine Learning, Support Vector Machine, Naive Bayes, Stemming, Lemmatization, Vectorization, Text Classification

1. Introduction

Computer-based technologies have transformed the way we live, work, socialize, play, and learn. Recent advancement in various fields has led to the collection of large amount of data. As the amount of data is huge that makes managing / analyzing data complex and challenging.

In a customized knowledge management mission, automated text categorization may play an important role, such as: real-time assignment of files into folder hierarchies; subject recognition to help topic-specific processing operations. Automatic Text Classification helps the text document to be retrieved. Classification of document requires machine learning technique. It is a supervised machine learning task.

The Support Vector Machine, proposed by Vapnik, provides a hyper plane that separates the classes with maximum distance, between two classes of data and has non-linear extensions [1]. It is a supervised classification algorithm which recently used successfully for many tasks of NLP as text classification [2][3]. SVM algorithm represents the text document as a vector where the dimension is the number of distinct keywords. If the document size is large then the dimensions are enormous of the hyperspace in text

classification which causes high computational cost. The feature extraction and reduction can be used to reduce the dimensionality [4].

NB classifiers are statistical classifiers. They can predict class membership probabilities such as the probability that a given tuple belongs to a particular class. Bayesian classification is based on Bayes' theorem. Advantage of using Naïve Bayes algorithm is that it does not require any feature extraction i.e. word to vector conversion which is required for SVM and it calculates the most probable class of each document which guarantees that each is labelled or most probably labelled class.

Assumptions:

1) It assumes that the influence of an attribute value on a given class is unconstrained of the values of the other attributes in the dataset [5]. This assumption has been noted as class conditional independence. It simplifies computations involved and thus considered "naive" [6].

2) It assumes that each every attribute contributes to the final outcome in equal.

Naïve Bayes Classifier has various models:

1) Gaussian Naïve Bayes (continuous data)

2) Multinomial Naïve Bayes (data with discrete features)

3) Bernoulli Naïve Bayes (binary data)

Since, the BBC corpus has distinct features we opted the multinomial naïve bayes algorithm.

2 Literature Survey

Machine Learning gives better understanding of documents classification. Machine learning is concerned with development of algorithms that allows computers to "learn" so as to improve the expected performance. Document classification is a well-established data mining problem and the issue of scientific papers classification is a specialization of this problem posing its own challenges [7].

Thorsten Joachim's et al in [8] conclude that with the massive information in the enterprise, organization of data is needed and using text categorization this problem can be overcome by using SVM avoids catastrophic failures and yields better results. SVM acknowledges particular properties of text like a, high dimensional space b, few irrelevant features c, and sparse instance vector. SVM's ability to generalize well in high dimensional space makes it easier for application in text categorization and increases its robustness.

Vandana Korde et al in [9] emphasizes that text classification process starts from collecting text document and then performing pre-processing which is tokenization , stemming, stop word removal and in indexing we perform term weighing using TF-IDF scheme. Feature selection constructs a vector space which improves accuracy of text classifier.

In [10] the authors looked for patent to patent similarity in which TF-IDF proved to be more powerful in decision making applications. A simple TF-IDF technique is compared with more complicated methods or extensions to TF-IDF.

In[11] the authors implement KNN classification with Stemmed and unstemmed features with train and test ratio as 60/40. After getting results it's clear that KNN works best for less number of features.

3 Proposed System

3.1 Methodology

In the proposed system we use BBC corpus contains 2225 documents over five classes. The different classes are business, entertainment, politics, sports and technology. This dataset contains Business class - 511 documents, Entertainment - 381 documents, Politics - 418 documents, Sports - 512 documents, Technology - 403 documents. The classification algorithms that are being implemented are: Support Vector Machine (SVM) and Naive Bayes Classifier (NBC).

3.2 Architecture of the Proposed System

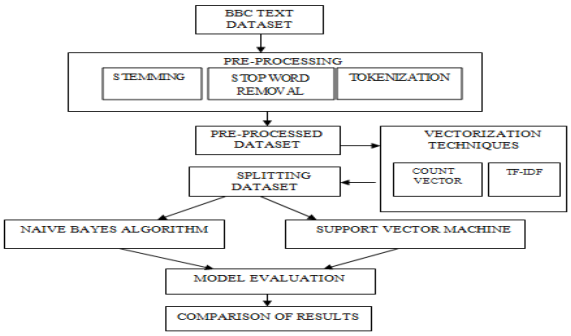

Fig. 1. Architecture of the proposed system

Data Collection

Data is collected from BBC corpus. This data contains 2225 documents over five categories. This dataset contains Business class - 511 documents, Entertainment - 381 document, Politics - 418 document, Sports - 512 document, Technology - 403 document.

Pre-processing

Collected data is pre-processed by tokenization, stop word removal and stemming. Document is tokenized first at paragraph level then by sentence level then to token (word). These tokenized data goes for stop word removal. Stemming is performed on output of stop word removal and this pre-processed data goes for Naïve Bayes algorithm and for SVM word to vector conversion.

As SVM algorithm accepts numerical data, so word to vector conversion is required. This is done by technique One-Hot Encoding technique. Word is converted into vector that is in to number by the presence of word in a document is indicated by '1' and absence of word is indicated by '0'. Thus word to vector conversion list is created.

We applied label encoding for class label. We assigned business as 0 , entertainment as 1, politics as 2, sport as 3 and tech as 4.

Tokenization

It is process of separating/ dividing text or sentences into smaller units called tokens. Tokens can take form of words, subwords or characters. The procedure followed for tokenization is as follows:

Input: c, corpus

Output: t, tokens

For f in c do

tk split f using space as a delimeter

t tk // assign the value of token to t

end for

return t

Stop word removal

Unimportant or unwanted text/data is referred to as stop words in natural language processing. The purpose is simply removing the words from the corpus that occur most commonly or frequently throughout the corpus. The procedure followed for stop word removal is as follows:

Input: t, a list of tokens

e, set of english language stop words

Output: p, pre-processed text

For t_i in t do

For t_k in t_i do

if t_k not found in e then

$p \leftarrow t_k$

end if

end for

end for

return p

Validation

We use five common different measures for the evaluation of the classification quality: Accuracy, Precision, Recall, F-Measure and Support. Accuracy is the proportion of the total number of predictions where correctly calculated. Precision is the ratio of the correctly classified cases to the total number of misclassified cases and correctly classified cases. Recall is the ratio of correctly classified cases to the total number of unclassified cases and correctly classified cases. Also, we used the F-measure to combine the recall and precision which is considered a good indicator of the relationship between them [5].

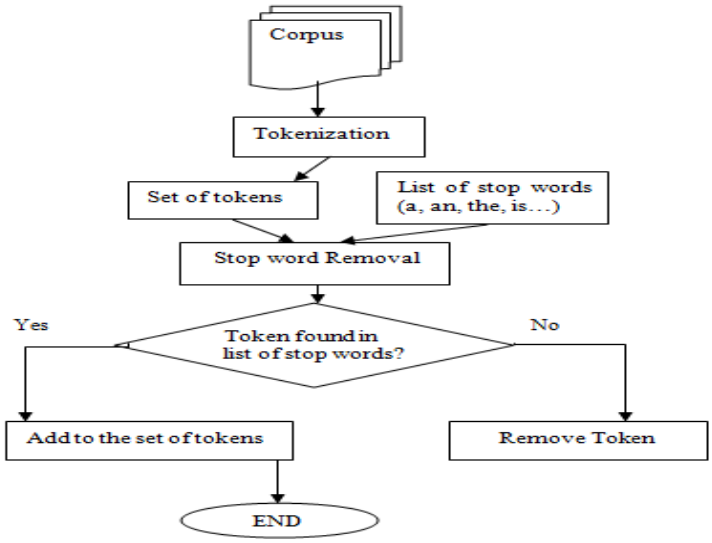

Fig. 2: Flowchart for tokenization and stop word removal

4 Implementation and Result Analysis:

The project is implemented on google colab cloud for python by installing nltk package for stop word removal, stemming etc, With the help of [12] and [13] programs were written in python programming language. The different combinations in which we have implemented the algorithms are as follows:

1) Naive Bayes Classifier with Lemmatization and Count Vector.

2) Naive Bayes Classifier with Lemmatization and TF-IDF vectorization technique.

3) Naive Bayes Classifier with Lemmatization, Count vector and TF-IDF vectorization technique.

4) Support Vector Machine with Lemmatization and Count Vector.

5) Support Vector Machine with Lemmatization and TF-IDF vectorization technique

6) Support Vector Machine with Lemmatization, Count vector and TF-IDF vectorization technique.

5. Results and Visualization

The Naive bayes model when implemented Lemmatization and Count Vector.

Train/Test Ratio: 80/20 Accuracy: 96.4

Confusion Matrix:

Table 1. Confusion matrix for Naive bayes using lemm and count vector

	0	1	2	3	4
0	96	0	3	0	1
1	1	70	5	0	1
2	2	0	91	0	0
3	0	0	0	94	0
4	3	0	0	0	78

Table 2. Performance metrics for Naive bayes using lemm and count vector

Class	Precision	Recall	F1 score	Support
0	94	96	95	100
1	100	91	95	77
2	92	98	95	93
3	100	100	100	94
4	97	96	97	81
Avg	96.6	96.2	96.4	

The Naive Bayes model when implemented with lemmatization and TF-IDF vectorization technique.

Train/Test ratio: 80/20 Accuracy: 95.7

Table 3. Confusion matrix for Naive bayes using lemm and TF-IDF vector

	0	1	2	3	4
0	105	0	0	0	1
1	2	74	1	0	0
2	2	0	79	0	1
3	0	0	0	97	0
4	2	0	1	1	79

Table 4. performance metrics for Naive bayes using lemm and TF-IDFvector

Class	Precision	Recall	F1 score	Support
0	95	99	97	106
1	100	96	98	77
2	98	96	97	82
3	99	100	99	97
4	98	100	99	83
Avg	98	97.2	97.4	

The Naive Bayes model when implemented with lemmatization and TF-IDF vectorization and count vector.

Train/Test ratio: 80/20 Accuracy: 96.85

Table 5. Confusion matrix for Naive bayes using lemm, count vector and TF-IDF vector

	0	1	2	3	4
0	93	0	3	0	3
1	1	82	1	0	3
2	1	0	83	0	1
3	0	0	0	94	0
4	1	0	0	0	79

Table 6. Performance metrics for Naive bayes using lemm, count vector and TF-IDF vector

Class	Precision	Recall	F1 score	Support
0	97	94	95	99
1	100	94	97	87
2	95	98	97	85
3	100	100	100	94
4	92	99	95	80
Avg	96.8	97	96.8	

The SVM model when implemented Lemmatization and Count Vector.

Train/Test Ratio: 80/20 Accuracy: 97.9

Table 7. Confusion matrix for SVM using lemm and count vector

	0	1	2	3	4
0	97	0	3	0	1
1	1	74	1	0	1
2	2	0	91	0	0
3	0	0	0	94	0
4	3	0	0	0	80

Table 8. Performance metrics for SVM using lemm and count vector

Class	Precision	Recall	F1 score	Support
0	96	97	97	100
1	100	96	98	77
2	92	98	97	93
3	100	100	100	94
4	99	98	99	81
Avg	98.2	98	98.2	

Support Vector Machine with Lemmatization and TF-IDF vectorization technique.

Train/Test ratio:80/20 Accuracy: 98.20

Table 9. Confusion matrix for SVM using lemm and TF-IDF vector

	0	1	2	3	4
0	105	0	0	0	1
1	1	75	1	0	0
2	0	0	81	0	1
3	0	0	0	97	0
4	1	2	0	1	79

Table 10. performance metrics for SVM using lemm and TF-IDF vector

Class	Precision	Recall	F1 score	Support
0	98	99	99	106
1	96	97	97	77
2	98	99	98	82
3	99	100	99	97
4	97	95	97	83
Avg	97.6	98	98	

Support Vector Machine with Lemmatization, Count vector and TF-IDF vectorization technique.

Train/Test ratio: 80/20 Accuracy: 97.52

Table 11. Confusion matrix for SVM using lemm, count vector and TF-IDF vector

	0	1	2	3	4
0	93	1	2	0	3
1	0	87	0	0	0
2	2	0	82	0	1
3	0	0	0	94	0
4	0	2	0	0	78

Table 12. performance metrics for SVM using lemm, count vector and TF-IDF vector

Class	Precision	Recall	F1 score	Support
0	98	94	96	99
1	97	100	98	87
2	98	96	97	85
3	100	100	100	94
4	95	97	96	80
Avg	97.6	97.4	97.4	

Performance visualization of Naive Bayes and SVM respectively.

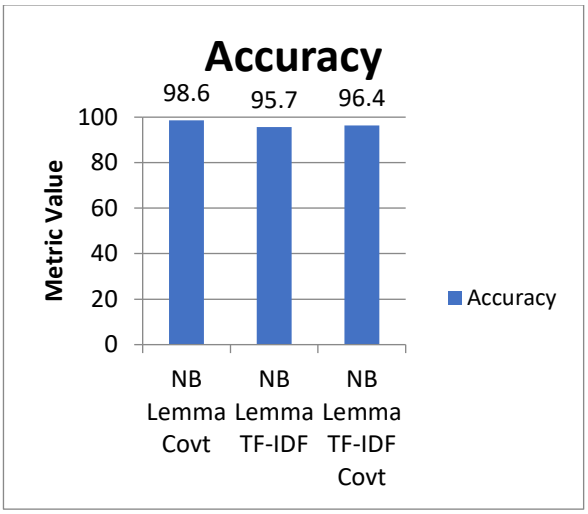

Fig 3- Comparison of Naïve Bayes Accuracy

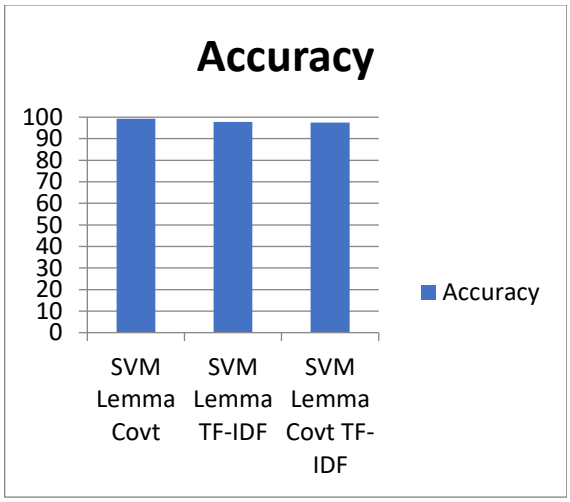

Fig 4- Comparison of SVM Accuracy

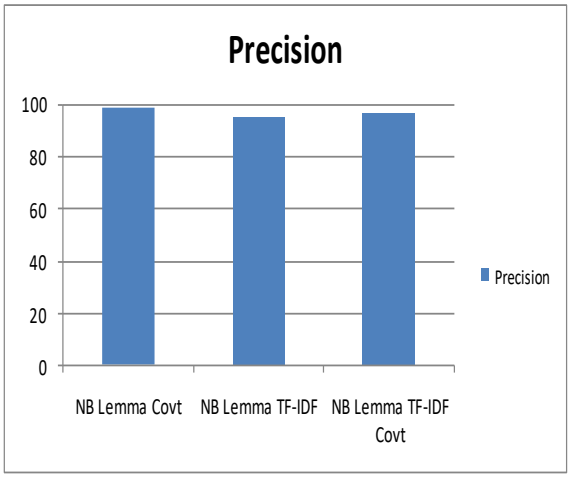

Fig 5- Comparison of Naïve Bayes Precision

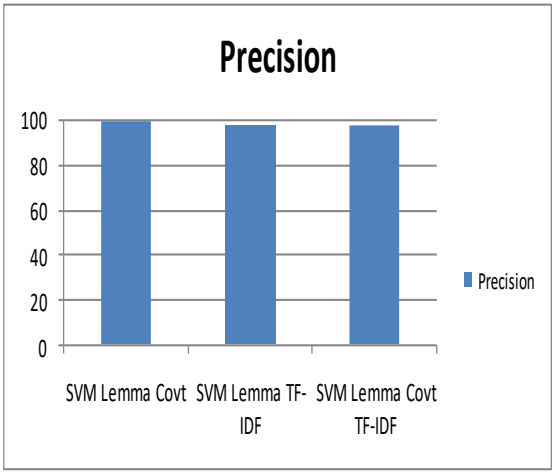

Fig 6- Comparison of SVM Precision

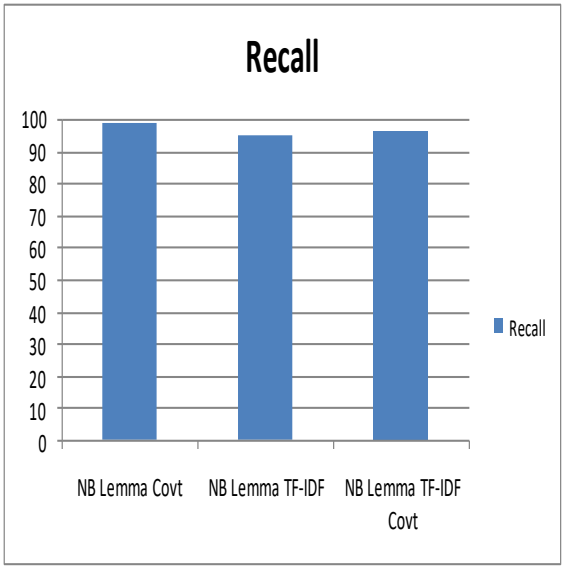

Fig 7– Comparison of Naïve Bayes Recall

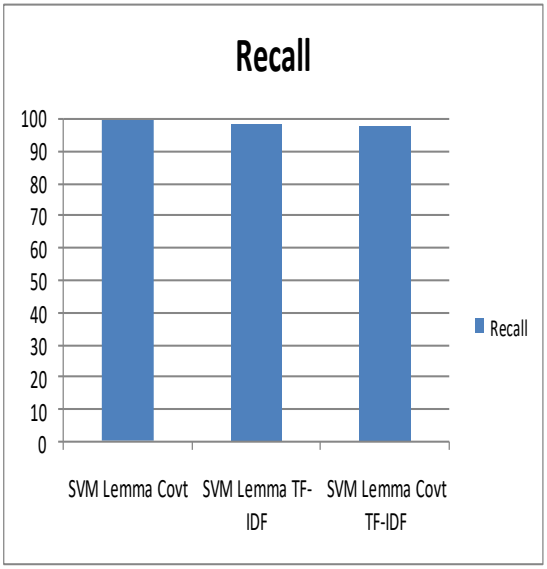

Fig 8- Comparison of SVM Recall

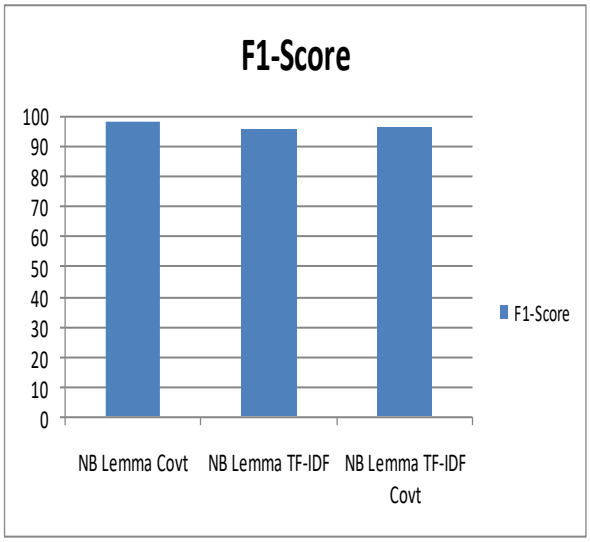

Fig 9- Comparison of Naïve Bayes F1- Score

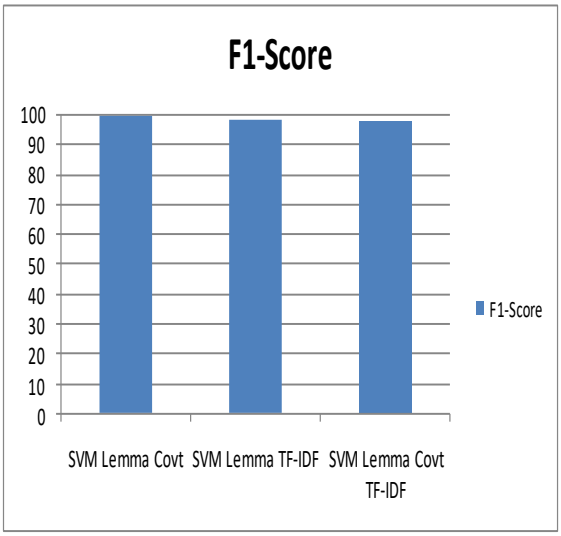

Fig 10- Comparison of SVM F1-Score

The final comparison between the different combinations of algorithms that we have implemented can be tabulated as the following:

Table 13- Performance comparison of algorithms

Algorithm	Accuracy	Precision	Recall	F1-Score
NBC Lemmatization Count Vector	96.4	96.6	96.2	96.4
NBC Lemmatization TF-IDF	97.5	98	97.2	97.4
NBC Lemmatization Count Vector TF-IDF	96.85	96.8	97	96.8
SVM Lemmatization Count Vector	97.9	98.2	98	98.2

SVM Lemmatization TF-IDF	98.2	97.6	98	98
SVM Lemmatization Count Vector TF-IDF	97.52	97.6	97.4	97.4

6. Conclusion and Future Works

In this paper, we implemented Naive Bayes and SVM with different feature extraction technique in combination with count vectorization and TF-IDF technique on the BBC corpus. As we are partitioning documents into training and testing sets with different ratios as 60/40, 70/30, 80/20 the accuracy varies, in general it gave better accuracy for 80 training documents and 20 testing documents division. SVM with Lemmatization and TF-IDF proved to be the most efficient since the highest accuracy achieved is 98.2 when the train and test ratio is 80/20. The other combination of SVM also proved to be accurate than Naive Bayes. In the future, other algorithms like logistic regression or convolution neural networks can be implemented and the impact of different feature extraction techniques and vectorization techniques can be observed. Also in future, comparisons can be made by using different versatile datasets.

References

[1] Xu, Z., Li, P., & Wang, Y. (2012). Text classifier based on an improved SVM decision tree. Physics Procedia, 33, 1986-1991.

[2] Rennie, J. D. (2001). Improving multi-class text classification with naive Bayes (Doctoral dissertation, Massachusetts Institute of Technology).

[3] Hotho, A., Nürnberger, A., & Paaß, G. (2005, May). A brief survey of text mining. In Ldv Forum (Vol. 20, No. 1, pp. 19-62).

[4] Joachims, T. (1998). Text categorization with support vector machines: Learning with many relevant features. Machine learning: ECML-98, 137-142.

[5] Jiawei Han, Micheline Kambar, Jian Pei, “Data Mining Concepts and Techniques” Elsevier Second Edition.

[6] S. Roy and A. Garg, "Predicting academic performance of student using classification techniques," 2017 4th IEEE Uttar Pradesh Section International Conference on Electrical, Computer and Electronics (UPCON), Mathura, 2017, pp. 568-572.

[7] Akritidis, Leonidas & Bozanis, Panayiotis. (2013). A supervised machine learning classification algorithm for research articles. Proceedings of the ACM Symposium on Applied Computing. 115-120. 10.1145/2480362.2480388.

[8] Thorsten Joachims. Text categorization with support vector machines: Learning with many relevant features. In Proceedings of the European Conference on Machine Learning, pages 137–142, 1998

[9] Korde, V.& Mahender, C. (March 2012). Text Classification and Classifiers: A Survey. International Journal of Artificial Intelligence & Applications (IJAIA), Vol. 3, No. 2, pp. 85-99.

[10] Omid Shahmirzadi, Adam Lugowski, Kenneth Younge. “Text Similarity in Vector Space Models: A Comparative Study”, 2018.

[11] Shri, Raj & Gaur, Sanjay & Chowdhary, Prof. (2018). Text Classification using KNN with different Feature Selection Methods.

[12] Python Software Foundation. Python Language Reference, version 3.6. Available at http://www.python.org

[13] G. van Rossum, Python tutorial, Technical Report CS-R9526, Centrum voor Wiskunde en Informatica (CWI), Amsterdam, May 1995.

[14] Akritidis, Leonidas & Bozanis, Panayiotis. (2013). A supervised machine learning classification algorithm for research articles. Proceedings of the ACM Symposium on Applied Computing. 115-120. 10.1145/2480362.2480388.