Text classification with Naïve Bayes

Yap Chi Yew
School of Computing
Asia Pacific University of Technology and Innovation (APU)
Kuala Lumpur, Malaysia
chiyew1@hotmail.com

Tan Guan Cheng
School of Computing
Asia Pacific University of Technology and Innovation (APU)
Kuala Lumpur, Malaysia
gctan2000@gmail.com

Derric Chong Wei Sen
School of Computing
Asia Pacific University of Technology and Innovation (APU)
Kuala Lumpur, Malaysia
d777.chong@gmail.com

Bee Jian Wei
School of Computing
Asia Pacific University of Technology and Innovation (APU)
Kuala Lumpur, Malaysia
wb04apu@gmail.com

Zailan Arabee Abdul Salam
School of Computing
Asia Pacific University of Technology and Innovation (APU)
Kuala Lumpur, Malaysia
zailan@apu.edu.my

***Abstract*—Naive Bayes is an algorithm which is swift and easy to apply and always used in text classification. In this paper, we anchor in the performance of Naive Bayes text classifiers with a couple of datasets. We have modified the code to measure the accuracy of the algorithm in predicting words. We also use an interactive graph to show the prediction of the algorithm. We found out that Naive Bayes can provide uncomplicated probabilistic predictions which are very smoothly interpretable with only a few tuneable parameters.**

Keywords—Naive Bayes algorithm, text classification, Multinomial Naive Bayes (MNB)

I. INTRODUCTION

With the advancement of technology, the internet has now become a huge database that contains billions of documents and data. The number of documents that are uploaded daily to the internet are also constantly increasing without any sign of stopping soon. Amongst the various types of documents on the internet, some are considered very helpful and impactful to our daily lives. For example, documents such as news articles, journals and research papers may provide us with information that allow us to keep ourselves updated about the various things that are happening all around the world. Some may even help us in terms of work such as completing a research and experiment or something as simple as a school assignment. As such, with these many documents to select from, a text classification system is needed for people to retrieve the desired information within a matter of seconds.

Text categorization, also known as text classification, is the process of assigning texts to a predefined set of categories based on the context of the document. It is mainly used to organize or classify documents on the internet or some dataset in machine learning. Due to the billions of documents being more challenging and hard to manually categories nowadays as it is very time consuming and not effective, a text classifier is considered important and have even been made into a research topic in the technology field to find out the most efficient and accurate classifier. The unique aspect of text classification is that it can be used on almost anything such as categorizing a news article to its topic or classifying languages into its respective origins or even filtering email to detect spam mail etc.

In this paper, our group will be utilising the text classifier along Naive Bayes algorithm to classify a dataset which consists of 18000 news posts into their respective categories. Machine learning will be implemented in this research to separate the dataset into two categories which is the training dataset and the test dataset. The training dataset will be used to train the algorithm while the test dataset will be used for testing on how accurate the documents are categorised into their groups.

This paper is separated into a few sections. Section 1 is about the introduction of this paper. Within section 1, section 1.1 talks about the literature review on the topic which includes similar work or research, methodology and recommendation or conclusion on the literature review. Section 2 is about the materials and methods used in this paper. Section 2.1 talks about the algorithm implementation of this research which is Naive Bayes algorithm. Section 3 is about the result and discussion. Section 3.1 talks about how our group changed a few parameters of the original code to determine what is affected and what has changed while Section 3.2 talks about the results of the code that has been modified. Finally, Section 4 is about our conclusion of this paper based on the results shown in Section 3.

II. LITERATURE REVIEW

There are many researches that have been conducted on text classification or categorisation in the past few years. Some of those projects involved the usage of other algorithms such as Support Vector Machines, K-Nearest Neighbor and logistic regression etc. However, this paper's focus will be on Naive Bayes algorithm.

Wongso et al [1] used a variety of Naive Bayes algorithms to classify a news article in Indonesia language. Text pre-processing techniques such as lemmatization and stop words are introduced to avoid words that are not meaningful. SVD and TF-IDF as a feature selection to test the accuracy is also taken. The combination of TF-IDF and Multinomial Naive Bayes algorithm gave the highest precision of 98.4% as a result when compared to the other combinations and this

shows that the accuracy and the response time can be further improved when multiple algorithms are combined with the appropriate feature selection.

Meanwhile, Murata et al [2] suggested "Japanese Word Sense Disambiguation using the Simple Bayes and Support Vector Machine Methods". The aim of this paper was to estimate the sense of word based on the contents of the documents. The author combined the use of simple Bayes and Support Vector Machine (SVM) to produce a result of 0.786 precision, which is the best among all the other submitted systems. However, when two simple Bayes were used by the author, a precision of 0.793 was produced when the parameters were adjusted.

Tsuruoka and Tsujii [3] proposed a method to improve the precision and recall of the protein name. The machine learning approach using Naive Bayes classifier was implemented after the dictionary approach is completed and the results are further classified into accepted or rejected statements. Contextual feature and Term feature were also used along the Naive Bayes classifier and the result of the research shows a drastic improvement from 48.6% to 74.3%.

Moreover, Dwivedi and Arya [4] provided a study of automatic text classification in information retrieval. Five different algorithms were used in this experiment to discuss the performance of each of the algorithms in information retrieval. A finding that supervised machine learning being able to classify documents faster and consistently was also discovered. In this paper, four datasets were used and SVM was found to have better performance for Life Science and Social Science topics while Naive Bayes was found to have a better performance in terms of computer science and engineering related topics. This shows that the result of each dataset varies depending on the algorithm implemented and it is best for the researcher to understand the characteristics of the selected dataset for better classification.

Finally, Mangal and Goyal [5] implemented Naive Bayes for text classification in Punjabi language. In this paper, an unknown dataset which was not involved in training from ajitjalandhar.com was taken to test the performance of the classifier. The domain of the dataset involved terror attack news, murder news, accidental news, and suicide news, respectively. Analysis of the result shows a 72% accuracy for terror attack news, 64% for murder news, 80% for accidental news and suicide news with 72%. This tells us that Naive Bayes performs relatively well as it is above the average score. However, different news that belongs to different contexts may have an impact on the classifier due to the nature of the news and training on the dataset will also affect the result of the classifier in the end.

III. METHODOLOGY

A) Dataset

The dataset that our group is using will be the 20-newsgroup corpus from the sklearn website. This 20-newsgroup data consists of roughly 20000 newsgroup documents that are distributed into 20 categories. The reason behind the selection of this newsgroup dataset is due to it being a popular dataset for research purposes in this text classification field. The dataset that was used by our group was even further sorted by the author into two training datasets which is 60% training and 40% testing and this document does not consist of any duplications of data, which simplifies the machine learning process. The final dataset after processing consists of 18846 documents that belong to 20 categories. Table I shows the list of 20 newsgroups.

TABLE I. CATEGORIES OF 20 NEWSGROUPS ACCORDING TO SUBJECT MATTERS

Major Category	Minor Category
Computer	comp.graphics
	comp.os.ms-windows.misc
	comp.sys.ibm.pc.hardware
	comp.sys.mac.hardware
	comp.windows.x
Miscellaneous	misc.forsale
Record	rec.autos
	rec.motorcycles
	rec.sport.baseball
	rec.sport.hockey
Religion	alt.atheism
	talk.religion.misc
	soc.religion.christian
Science	sci.crypt
	sci.electronics
	sci.med
	sci.space
Talk	talk.politics.guns
	talk.politics.mideast
	talk.politics.misc

TABLE II. NUMBER OF DOCUMENTS OF 20 NEWSGROUPS ACCORDING TO SUBJECT MATTERS

Topic	# documents in testing	# documents in training	# documents
alt.atheism	319	480	799
comp.graphics	389	584	973
comp.os.ms-windows.misc	394	591	985
comp.sys.ibm.pc.hardware	392	590	982
comp.sys.mac.hardware	385	578	963
comp.windows.x	395	593	988
misc.forsale	390	585	975
re.autos	396	594	990
rec.motorcycles	398	598	996
rec.sport.baseball	397	597	994
rec.sport.hockey	399	600	999
sci.crypt	396	595	991
sci.electronics	393	591	984
sci.med	396	594	990
sci.space	394	593	987
soc.religion.christian	398	599	997
talk.politics.guns	310	465	775
talk.politics.mideast	364	546	910
talk.politics.misc	376	564	940
talk.religion.misc	251	377	628
Total	7532	11314	18846

B) Data pre-processing

Before the classification process is conducted, the dataset needs to go through some data pre-processing process to optimize the data. The first process that our group has done is to convert the content of the dataset into a vector of numbers using TF-IDF. TF-IDF is a measure of how relevant a word is in a collection of documents and its main purpose is for information retrieval. TF-IDF is usually calculated by using two different metrics: term frequency (tf) and inverse document frequency (idf). The calculation of TF-IDF is shows below:

$$TFIDF\ score\ for\ term\ i\ in\ document\ j = TF(i,j) * IDF(i)$$

where

$IDF = Inverse\ Document\ Frequency$

$TF = Term\ Frequency$

$$TF(i,j) = \frac{Term\ i\ frequency\ in\ document\ j}{Total\ words\ in\ document\ j}$$

$$IDF(i) = \log_2\left(\frac{Total\ documents}{documents\ with\ term\ i}\right)$$

and

$t = Term$

$j = Document$

Formula 1: Simplify Calculation of TF-IDF

$$w_{i,j} = tf_{i,j} * \log\left(\frac{N}{df_i}\right)$$

$tf_{i,j} = number\ of\ occurences\ of\ i\ in\ j$

$df_i = number\ of\ documents\ containing\ i$

$N = total\ number\ of\ documents$

Formula 2: Mathematical Calculation of TF-IDF

C) Naive Bayes Classifier

Naive Bayes algorithm is a supervised machine learning algorithm which is based on the original probability calculator, the Bayes Theorem. This algorithm is mainly used to solve classification problems and in context of this paper, our group will use this algorithm to solve a text classification problem based on the dataset chosen. Naive Bayes classifier is a simple and effective classification method that is used in machine learning to help build a prediction or classification model. Since Naive Bayes was based on the Bayes Theorem, it will use the principle of the Bayes Theorem to predict an outcome based on the probability of a given object.

The name of Naive Bayes came from the combination of two words which are Naive and Bayes separately. The term naive is used due to Naive Bayes' naiveness, which assumes that the features of measurements are independent from each other, meaning that each word is measured independently and does not relate to one another. Meanwhile, the word Bayes comes from the Bayes Theorem, which is a calculation of conditional probability. The formula for the Bayes Theorem in context of text classification is shown below:

$$P(c|d) = (P(d|c)*P(c)) / P(d)$$

Formula 3: Bayes Theorem Formula

where "d" is document and "c" is a class. P(d) is the probability of a given document and P(c) is the probability of a class. This formula is used to calculate the probability that a given document belongs to the class "c".

Multinomial Naive Bayes is a variation of the original Naive Bayes algorithm. It is mainly used for document or text classification when the data is distributed in a multinomial form. MNB functions by using the frequency of the words for prediction or classification purposes.

In a nutshell, Naive Bayes algorithm is a good algorithm to perform text classifier. However, it still needs some modifications to improve its accuracy. For instance, we can combine the use of simple Bayes and Support Vector Machine (SVM) to produce a better result. Moreover, we can also apply 10-fold cross validation in the training dataset text classifier to improve the precision of the Naive Bayes algorithm. Thus, the Naive Bayes algorithm is still an advantageous algorithm in text classification.

IV. MATERIALS AND METHODS

A) Algorithm Implementation

This section of coding we are just focusing on the discussion and implementation of the original coding of the algorithm.

```
In [1]: from sklearn.datasets import fetch_20newsgroups

        data = fetch_20newsgroups()
        data.target_names
```

Fig. 1. In[1] Code block

The sparse word count features from the twenty newsgroups will be utilized to show how we are classifying the documents into categories.

First the dataset is downloaded by us from the sklearn website. This bundle of datasets contains 18846 of training data samples in total. Then, we check and see the target names after running this In [1] block of code.

```
Out[1]: ['alt.atheism',
         'comp.graphics',
         'comp.os.ms-windows.misc',
         'comp.sys.ibm.pc.hardware',
         'comp.sys.mac.hardware',
         'comp.windows.x',
         'misc.forsale',
         'rec.autos',
         'rec.motorcycles',
         'rec.sport.baseball',
         'rec.sport.hockey',
         'sci.crypt',
         'sci.electronics',
         'sci.med',
         'sci.space',
         'soc.religion.christian',
         'talk.politics.guns',
         'talk.politics.mideast',
         'talk.politics.misc',
         'talk.religion.misc']
```

Fig. 2. Out[1] Code block

This output block at block Out [1] shows the 20 classes of datasets that can be used for text classification.

```
In [2]: categories = ['talk.religion.misc', 'soc.religion.christian',
                      'sci.space', 'comp.graphics']
        train = fetch_20newsgroups(subset='train', categories=categories)
        test = fetch_20newsgroups(subset='test', categories=categories)
```

Fig. 3. In[2] Code block

This original code at block In [2] has only selected just some of the categories to train and test for simplicity text classification purposes.

```
In [3]: print(train.data[5])
```

Fig. 4. In[3] Code block

Code at block In [3] represents the entry of the data. We can try to print out and check the content of the data sample from the datasets according to the number label of the respective data samples.

```
From: dmcgee@uluhe.soest.hawaii.edu (Don McGee)
Subject: Federal Hearing
Originator: dmcgee@uluhe
Organization: School of Ocean and Earth Science and Technology
Distribution: usa
Lines: 10


Fact or rumor....?  Madalyn Murray O'Hare an atheist who eliminated the
use of the bible reading and prayer in public schools 15 years ago is now
going to appear before the FCC with a petition to stop the reading of the
Gospel on the airways of America.  And she is also campaigning to remove
Christmas programs, songs, etc from the public schools.  If it is true
then mail to Federal Communications Commission 1919 H Street Washington DC
20054 expressing your opposition to her request.  Reference Petition number

2493.
```

Fig. 5. Out[3] Code block

Code at block Out [3] print out the respective training data sample for us to check the content of the data.

```
In [4]: from sklearn.feature_extraction.text import TfidfVectorizer
        from sklearn.naive_bayes import MultinomialNB
        from sklearn.pipeline import make_pipeline

        model = make_pipeline(TfidfVectorizer(), MultinomialNB())
        model.fit(train.data, train.target)
        labels = model.predict(test.data)
```

Fig. 6. In[4] Code block

For now, we are required to convert the content in the datasets of each string into a vector of numbers. We use the TF-IDF vectorizer at code block In [4] to convert the raw documents in the datasets to matrix form of TF-IDF features. Then, we import the Naive Bayes algorithm for multinomial models to do text classification. Next, we import the pipeline constructor to generate a pipeline that can chain itself to the Multinomial Naive Bayes text classifier.

```
In [5]: def predict_category(s, train=train, model=model):
            pred = model.predict([s])predict_category('determining the screen resolution')

            return train.target_names[pred[0]]
```

Fig. 7. In[5] Code block

We implement the predict() method to the pipeline to the code block In [5] after having the algorithm part to distinguish and prompt the category for the strings.

```
In [6]: predict_category('sending a payload to the ISS')
Out[6]: 'sci.space'

In [7]: predict_category('discussing christian')
Out[7]: 'soc.religion.christian'

In [8]: predict_category('determining the screen resolution')
Out[8]: 'comp.graphics'
```

Fig. 8. In[6], In[7], In[8], Out[6], Out[7], Out[8] Code block

Code block In [6], In [7] and In [8] show that the predict_category() function can read the input of the string by the user to do the prediction. The code block Out [6], Out [7] and Out [8] show the prediction results of the respective strings. Therefore, we can explain the output result one by one according to the accuracy of the algorithm classifier.

V. RESULTS AND DISCUSSION

The "sklearn" is a free machine learning library. To show how we can classify the documents into categories, the sparse word count features from 20 newsgroups is used.

```
In [1]: from sklearn.datasets import fetch_20newsgroups

        data = fetch_20newsgroups()
        data.target_names
```

Fig. 9. In[1] Code block

First, we downloaded and imported the 20 newsgroups datasets from sklearn website which contains totals of 18846 training data samples. After running In [1] block of code, we check and ensure the target names are correct.

```
Out[1]: ['alt.atheism',
         'comp.graphics',
         'comp.os.ms-windows.misc',
         'comp.sys.ibm.pc.hardware',
         'comp.sys.mac.hardware',
         'comp.windows.x',
         'misc.forsale',
         'rec.autos',
         'rec.motorcycles',
         'rec.sport.baseball',
         'rec.sport.hockey',
         'sci.crypt',
         'sci.electronics',
         'sci.med',
         'sci.space',
         'soc.religion.christian',
         'talk.politics.guns',
         'talk.politics.mideast',
         'talk.politics.misc',
         'talk.religion.misc']
```

Fig. 10. Out[1] Code block

This shows the output of the In [1] block of code, which contains 20 classes of datasets and can be used for text classification.

```
In [2]: ###select the categories and use the training and testing data set
        categories = ['alt.atheism',
         'comp.graphics',
         'comp.os.ms-windows.misc',
         'comp.sys.ibm.pc.hardware',
         'comp.sys.mac.hardware',
         'comp.windows.x',
         'misc.forsale',
         'rec.autos',
         'rec.motorcycles',
         'rec.sport.baseball',
         'rec.sport.hockey',
         'sci.crypt',
         'sci.electronics',
         'sci.med',
         'sci.space',
         'soc.religion.christian',
         'talk.politics.guns',
         'talk.politics.mideast',
         'talk.politics.misc',
         'talk.religion.misc']
        train = fetch_20newsgroups(subset='train', categories=categories)
        test = fetch_20newsgroups(subset='test', categories=categories)
```

Fig. 11. In[2] Code block

Then, we train and test the text classification algorithm by using the 20 selected categories from the imported datasets. We load only a sub-selection of categories by passing the list of the categories to load to the "fetch_20newsgroups" function.

```
In [3]: ###to print out your train data(a representative entry from the data)
        print(train.data[666])

        From: davewood@bruno.cs.colorado.edu (David Rex Wood)
        Subject: Rockies need some relief
        Nntp-Posting-Host: bruno.cs.colorado.edu
        Organization: University of Colorado, Boulder
        Lines: 13

        Once again, the Rockies bullpen fell apart.  Andy Ashby pitched six (somewhat
        shaky) innings giving up just one run.  Then game the dreaded relief.  Three
        picthers combined to give up 3 runs (one each I believe) in the 7th inning
        and blew the save opportunity.  (Final was 4-2 vs Expos).

        Despite their problems in the pen, I think the Rockies are a team that wont
        be taken lightly.  Going into today's game, the had the league's leading
        hitter and RBI man (Galarraga), two of the leaders in stolen bases (Young
        and Cole) and increasingly strong starting pitching.
        --
        ------------------------------------------------------------------------
        David Rex Wood -- davewood@cs.colorado.edu -- University of Colorado at Boulder
        ------------------------------------------------------------------------
```

Fig. 12. In[3] Code block

In [3] shows the entry of the data. We used the print() function to print out and check the content of the data sample by the respective number sequence of the data sample. In [3] also shows we checked the 666th data sample.

```
In [4]: from sklearn.feature_extraction.text import TfidfVectorize
        from sklearn.naive_bayes import MultinomialNB
        from sklearn.pipeline import make_pipeline

        model = make_pipeline(TfidfVectorizer(), MultinomialNB())
        model.fit(train.data, train.target)
        labels = model.predict(test.data)
```

Fig. 13. In[4] Code block

After that, we converted the content in the datasets of each string into a vector of numbers by using TF-IDF vectorizer which shows at block In [4] then convert the raw documents into matrix form of TF-IDF features.

Naive Bayes algorithm for multinomial models also imported for text classification. Furthermore, we also generate a pipeline that chains itself to the text classifier by using pipeline constructor.

```
In [5]: %matplotlib inline
        import numpy as np
        import matplotlib.pyplot as plt
        import seaborn as sns; sns.set()
```

Fig. 14. In[5] Code block

Python statistical data visualization graph function, Seaborn is imported to our algorithm which shows at In [5] to print the informative statistical graphics.

```
In [6]: from sklearn.metrics import confusion_matrix
        mat = confusion_matrix(test.target, labels)
        sns.heatmap(mat.T, square=True, annot=True, fmt='d', cbar=False,
                    xticklabels=train.target_names, yticklabels=train.target_names)
        plt.xlabel('correct label')
        plt.ylabel('estimated label');
```

Fig. 15. In[6] Code block

We import confusion matrices at code block In [5] to evaluate the accuracy of a classification. This can help us to predict the labels for the test data more accurately. We can evaluate them to learn about the performance of the estimator. Thus, with the imported Seaborn graph and confusion matrices, now we can implement a heatmap graph to show the results of the text classification by the algorithm.

The results of the graph will be showed in the 3.2 Results section

```
In [8]: def predict_category(s, train=train, model=model):
            pred = model.predict([s])
            return train.target_names[pred[0]]
```

Fig. 16. In[8] Code block

We implement the predict() method to the pipeline to the code block In [8] after having the algorithm part to distinguish and prompt the category for the strings.

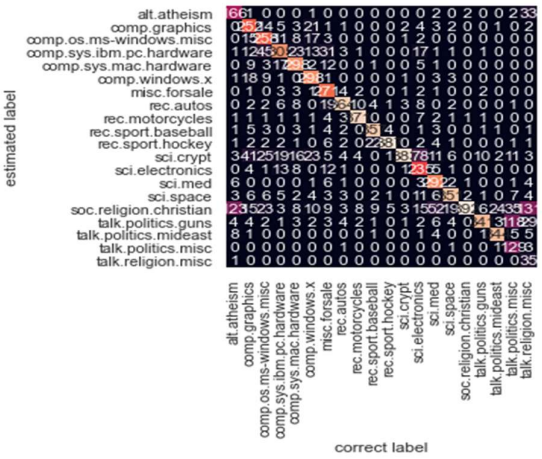

Fig. 17. Heatmap of results of Naïve bayes classifier

Due to the simplicity of Naive Bayes algorithm, it will highlight the highest value of the estimated label against the correct label, which makes the heatmap easier for viewing. The higher value means the accuracy is better. The reason that the numbers are outside the heatmap is because the word categorized in 2 classes creates confusion. So, for those values that are not highlighted, the lower value is more accurate. An example of the confusion the algorithm faced is the comparison between the talk about Christianity and space as the many words crossed over causing the comparison value to be 19. Fortunately, the confusion produced a low value in the comparison.

```
In [7]: from sklearn.metrics import accuracy_score
        print ("Accuracy score is ", accuracy_score(labels, test.targ

        Accuracy score is  0.7738980350504514
```

Fig. 18. In[7] Code Block

To get the algorithm's accuracy, we must import an accuracy scoring tool as shown in code block [7], it will compute the subset accuracy in which the result showed that it has a 77.39% accuracy score.

```
In [9]: ###Now use predict_category(' ') to let to machine to do the text classification

In [10]: predict_category('guns')
Out[10]: 'talk.politics.guns'

In [11]: predict_category('discussing atheism')
Out[11]: 'alt.atheism'

In [12]: predict_category('motor')
Out[12]: 'rec.autos'
```

Fig. 19. In[10], In[11], In[12], Out[10], Out[11], Out[12] Code block

The code block [10], [11] and [12] are category prediction functions which will predict the category of the string input. It then produced the results of the prediction made as shown in code block [10], [11] and [12] . After the algorithm was modified, we were able to classify the text into 20 categories with the accuracy of the algorithm classifier.

VI. CONCLUSION

Naive Bayes algorithm has been selected for our text classification project to classify a dataset with 18846 data samples. We had successfully implemented the algorithm and classified the texts in the datasets. We finished the required data pre-processing process and training process before we implemented the text classification. Machine learning has also been used in this research project to separate the dataset. The result showed that the code had achieved our goals and the accuracy scoring tool showed the accuracy score of our result is 77.39%.

The importance of this research is Naive Bayes can be modified into many other forms. The meaning is necessary modifications can be made to let the Naive Bayes algorithm have a significant improvement on its accuracy. Moreover, other improvements also can be made to increase the measurement accuracy of Naive Bayes algorithm such as we have implemented the confusion matrix into the original algorithm.

In conclusion, we had learnt the technique of implementing text classification with machine learning after we finished this research. We have learnt that text classification is a good tool for tagging your products on e-commerce platforms, analysing the trends of SEO keywords, monitoring the branding and product advertisements on social medias and other remarkable usages. We also gained much knowledge related to Naive Bayes algorithm.

REFERENCE

[1] R. Wongso, F. Ariandy, L. Brandon, C. Trisnajaya, O. Rusli and Rudy (2017). News Article Text Classification in Indonesian Language. Procedia Computer Science, 116, pp.137-143. Available at: https://doi.org/10.1016/j.procs.2017.10.039.

[2] M. Murata, M. Utiyama, K. Uchimoto, Q. Ma and H. Isahara. (2001). Japanese word sense disambiguation using the simple Bayes and support vector machine methods. SENSEVAL '01: The Proceedings of the Second International Workshop on Evaluating Word Sense Disambiguation Systems. [Online]. 1 (1). p. 4. Available from: https://dl.acm.org/doi/10.5555/2387364.2387397.

[3] Y. Tsuruoka and J. Tsujii (2003). Boosting precision and recall of dictionary-based protein name recognition. BioMed '03: Proceedings of the ACL 2003 workshop on Natural language processing in biomedicine - Volume 13. [Online]. 13 (3). p. 8. Available from: https://dl.acm.org/doi/10.3115/1118958.1118964.

[4] S. Kumar Dwivedi and C. Arya (2016). Automatic Text Classification in Information retrieval: A Survey. ICTCS '16: Proceedings of the Second International Conference on Information and Communication Technology for Competitive Strategies. [Online]. p. 6. Available from: https://dl.acm.org/doi/10.1145/2905055.2905191.

[5] S. Bajaj Mangal and V. Goyal (2014). Text News Classification System using Naïve Bayes Classifier. : An International Journal of Engineering Sciences. [Online]. 3 (39). p. 5. Available from: http://ijoes.vidyapublications.com/paper/Vol13/39-Vol13.pdf.