Analysis of Passive Aggressive Algorithm in Fake News Detection

Merzouk Fatima Zohra[1] and Salem Mohammed[2]

University of Mascara, Mascara, Algeria

Abstract. Fake news has emerged as a new and unavoidable issue that can harm individuals and society and it becomes easy to produce and spread this fake news. Attempts are being made to dissolve it using various methods such as Machine learning. In this paper, a detailed study is presented on using the passive-aggressive classifier which is an online machine learning algorithm, to examine texts and distinguish between false and accurate information thus detect fake news. It was tested in several ways with different datasets of diverse sizes and using different feature extraction techniques and gives promising results.

Keywords: Fake News · Fake News Detection · Passive Aggressive Algorithm

1 Introduction

Because of recent advances in computer science, it is now easy to fabricate and diffuse fake news. In some ways, technological advancements have a hand in creating this issue. However, it can also be the solution. Attempts are being made to dissolve it using various methods such as Machine learning.
A machine learning approach can examine the writing styles and linguistic patterns, which can help distinguish between false and accurate information. Machine learning operates by training an algorithm through enormous amounts of data and examining its decision-making effectiveness.
The passive-aggressive classifier is an online learning algorithm that learns with a significant, consistent data flow. It is a linear classifier that ensures that the classification is correct, and that's why it responds aggressively when the prediction is not entirely positive, thus updating the weight vector of prediction and remaining passive when the prediction is positive.[11]
Many papers were published about fake news identification. Because of textual bases, fake news detection is comparable to spam detection, as the research [5] mentioned, and may be dealt similarly while in [5], the author compared the PA (Passive-Aggressive) classifier and the Naive Bayes classifier to LSTM (Long Short Time Memory) in a dataset of 30000 lines using two vectorization approaches, TFIDF, which was employed with only PA (Passive-Aggressive) and naive Bayes, and one hot encoder (bag of words). Passive-aggressive gives an

accuracy of 99%, outperforming both Naive Bayes and LSTM, which both yield a decent percentage.
In the paper [8], the prediction of fake news was by Support Vector Machine and passive-aggressive classifiers. SVM achieved an accuracy of 95 percent when evaluated with a dataset of 6335 lines, whereas Passive-Aggressive generated an accuracy of 92 percent. The paper did discuss how SVM gives more accurate results, especially in small datasets, but takes a long time to train on large ones, which is the opposite of passive-aggressive. Still, it can produce less accurate results.
The author of [4] used a passive-aggressive approach to detect fake news across two separate datasets from Kaggle, 3983 and 25000 tuples, achieving an accuracy of 98.87% and 96.25%. They also evaluated various machine learning algorithms against PA (Passive-Aggressive) classifiers and calculated prediction percentages.
In this paper, a detailed study is presented of applying Passive-Aggressive algorithm based classifier to detect fake news, providing the classifier with different sizes on datasets and examining its behavior with every one of them, as well as, multiple feature extraction techniques and evaluated their performance every time. The passive-aggressive classifier will be compared to other well-known classifiers and assessed the time taken for each one.

2 Passive aggressive algorithm

The authors in [11] developed a category of online learning called Passive-Aggressive(PA) Classifier. The objective is to use a training example to improve the classifier and discard it. The classifier works by being passive to proper classification and aggressive to erroneous classification, resulting in changes.

The passive-aggressive algorithm is a linear classifier that builds up a significant distance between the decision margin and the data points called a buffer zone, defined as a changeable area surrounding the decision boundaries.[3]
Figure 1 shows the buffer zone created by the passive-aggressive algorithm.

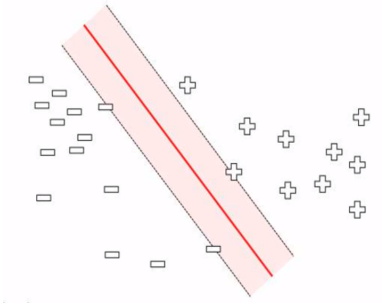

Fig. 1. *The buffer zone of The passive aggressive algorithm [14]*

On round t, the instance given to the algorithm is represented by x_t , a vector in R^n. Furthermore, suppose that x_t corresponds to a distinct label $y_t \in \{+1, -1\}$.

$$\begin{cases} X = \{\overline{x}_0, \overline{x}_1, ..., \overline{x}_t ...\} \; where \; \overline{x}_i \; \in \; \mathbb{R}^n \\ Y = \{y_0, y_1, ..., y_t ...\} \; where \; y_i \; \in \; \mathbb{R}^n \end{cases} \tag{1}$$

The algorithm aims to learn the weight vector $w \in R^n$ that the classification function uses to make predictions. The value $|w.x|$ is the degree of certainty in this prediction.[17]

$$\widetilde{y_t} = \pm \left(w_t.x_t\right) \tag{2}$$

The margin $y_t \left(w_t.x_t\right)$ returns a positive result when the algorithm makes a correct prediction. However, the positive result isn't enough[11]. It is necessary to attain a margin of at least one regularly, or it will face penalization with the loss function , and weights will be updated.

Figure 2 explains the prediction values and where the passive-aggressive algorithm will suffer loss.

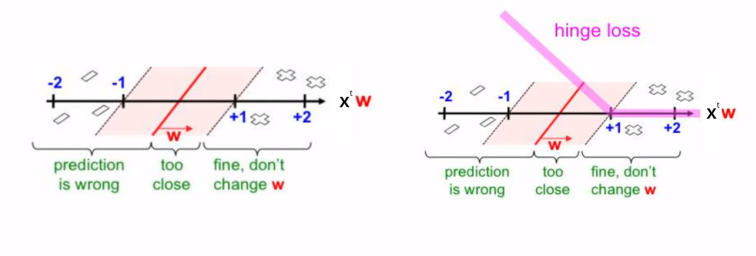

Fig. 2. *The passive aggressive behavior of the algorithm with hinge-loss function [14]*

The hinge-loss function shown below:

$$L(w_t) = max\left(0, 1 - y.\left(w_t.x_t\right)\right) \tag{3}$$

The form $y\left(w.x\right)$ is based on the classification function $(w.x)$, with weight vector w, input x, and classification's objective y. The penalty L is calculated based on the distance to the decision limit and how far from the correct prediction was.

$$w_{t_{+1}} = w_t + \frac{max\left(0, 1 - y.\left(w_t.x_t\right)\right)}{||x_t||^2 \; + \; \frac{1}{2C}} \; y_t.x_t \tag{4}$$

As a result, the value of the loss function is later utilized to update the weight vector. This update indicates that if it had a positive prediction but didn't score high enough, the weight vector will be elevated for those same inputs, so the instances score higher next time. It would be the opposite if the estimate were

negative.[14]
Figure 3 explains how the weight vector is updated.

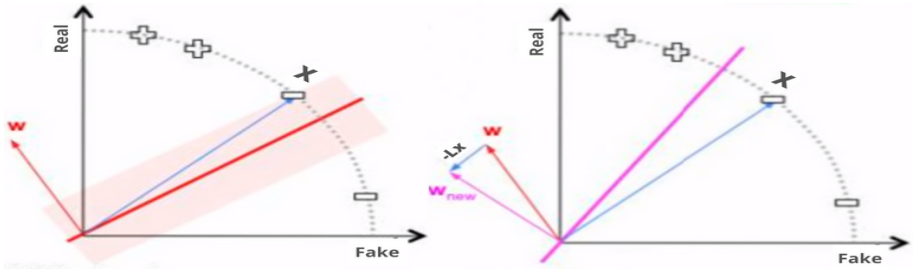

Fig. 3. *Updating the margin w using The hinge-loss function [14]*

And all the steps that were explained earlier become the Algorithm 1.

The algorithm starts by initializing the weight vector. After receiving the inputs, it makes predictions. If the predictions are false, it calculates the loss and updates the weight vector.

Algorithm 1 Passive Aggressive Algorithm

Initialize $w = (0, 0, ...0)$

Monitor Stream :
Receive input $x = \{\overline{x}_0, \overline{x}_1, ..., \overline{x}_t ...\}$ *where* $x \in \mathbb{R}^n$
if $(w.x) > 0$ **then**
  Predict $y = 1$
else $y = 0$
end if
Receive correct value $y \in \{+1, -1\}$
Suffer Loss $l = max\,(0, 1 - y.\,(w.x))$
Update $w = w + (y.l.x)$

3 Passive-Aggressive based fake news detection Approach

Detecting fake news with a Passive-Aggressive is an online learning process that aims to train a model to classify input news, its particularity is that it did not need a big amount of data, it updates the model parameters after each entered data. The proposed approach is depicted in figure 4:

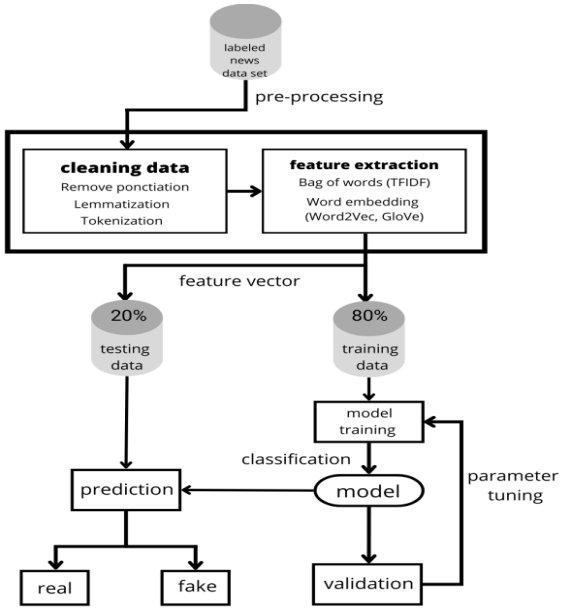

Fig. 4. *The steps of the proposed approach*

3.1 Data Cleaning

After deleting superfluous attributes and missing and incorrect data, then it proceed to remove the punctuation and URL and divide the text into separate words.

3.2 Feature extraction

Organizing complex data in a structured manner is another component of feature engineering. Text, audio, and video are examples of unstructured data forms. To translate words into numerical vectors, designing natural language processing algorithms often begins with the data transformation process. The three primary methods for feature extraction for text TFIDF[1], Word to Vector(Word2Vec) [13], and GloVe [16].

– **Term Frequency - Inverse Document Frequency (TFIDF)** The best way to avoid common words to dominate the document although they may not have as much "informational content" for the model as uncommon ones is to scale the frequency of terms based on how repeatedly they appear in all texts, penalizing widely used words in several publications. The measure Term Frequency (TF) determines how frequently a term appears in a document and the Inverse Document Frequency (IDF) is the criteria for measuring a word's relevance.[1]

– **Word to Vector (Word2Vec)**
Word2Vec is a technique of learning independent word embedding from a text corpus quickly and efficiently. Word Embedding, also known as Word Vector, is a numeric vector input representing words with similar meanings. Each word is mapped to a single vector, then trained in a neural network-like manner. The vectors attempt to capture various aspects of that word in relation to the rest of the text. [13]
Word2Vec excels in grouping related words together and making highly accurate approximations about word meanings based on context defined by nearby words.
– **Global Vectors (GloVe)**
Global Vectors (GloVe) is an unsupervised approach that generates word embedding from a corpus's global word-word co-occurrence matrix. Words are assigned a value based on their proximity to one another, as determined by the window size of the generated vectors. Depending on their value, these vectors can be brought closer or further apart. [16] The GloVe does not rely on local statistics (local context information of words). It utilizes worldwide statistics (word co-occurrence). [12][9]

3.3 Training and Evaluating the model

Randomly shuffling and balancing the dataset to obtain the most reliable results, resulting in reduced dataset size. Then dividing the datasets into the train datasets having 80% of the original dataset and the test dataset having the remaining 20%. For the parameter tuning, the training dataset will be divided into two others, resulting in the training dataset and the validation dataset.
After the parameter tuning, the model will be tested on the remaining 20% and see the accuracy of its prediction.

4 Simulation Results

4.1 Datasets settings

To carry out the experiments, several datasets are considered, Table 1 summarizes the datasets characteristics and contains the abbreviation that will be used to refer to them later.

Name	Label	Rows Number	Year	Abbreviation
Fake News Detection Challenge KDD 2020 [7]	2	5058	2020	KDD-20
Fake News [6]	2	26000	2018	FN-18
Fake and real news dataset [2]	2	42834	2017	FRN-17
Fake News Sample [10]	12	356112	2018	FNS-18

Table 1. *parameter tuning of first dataset*

4.2 Experimentation description

The evaluation of the proposed approach will be through severing tests

- First, trying to find the best parameters for training the proposed Passive-aggressive classifier in the KDD-20, and FN-18 datasets.

- The second experimentation is testing passive-aggressive outcomes with different vectorization methods TFIDF, Word2Vec, and GloVe with the FN-18 and FRN-17 datasets.
- Last, testing the accuracy-time ratio of our proposed passive-aggressive classifier against the classifier eXtreme Gradient Boosting (XGBoost) with the FNS-18 dataset.

4.3 Tuning of the PA parameters

This step is concerned with selecting the passive aggressive classifier parameters. The aggressiveness parameter C is set to 1 and the maximum iteration is 100 iteration by default, but but the goal was to investigate if changing these values would result in a better outcomes.

The aggressiveness parameter C

So the test will evaluate the accuracy with values of C ranging from 0 to 2, increasing the value by 0.1 each time. Different values of C are tested on 40% of each dataset, called the validation dataset.

Table 2 shows the results of parameter tuning of the aggressiveness parameter C in the **KDD-20** dataset.

	C parameter	0.5	1.0	1.5	**1.6**	2.0
KDD-20 dataset	Accuracy	62.91	62.56	63.61	**64.08**	61.1
	Precision	61.96	62.89	62.65	**61.47**	64.42
	Recall	54.87	54.38	59.32	**52.82**	55.34

Table 2. *Results of parameter tuning of the **KDD-20** dataset*

Figure 5 displays the accuracy values during the validation process on the **KDD-20** data-set for C parameter tuning.

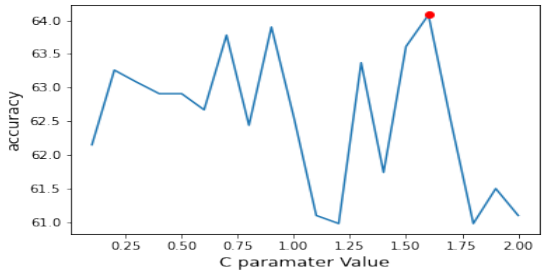

Fig. 5. *C testing on the **KDD-20** data-set*

Table 3 shows the results of parameter tuning of the aggressiveness parameter C in the **FN-18** dataset.

	C parameter	**0.1**	0.5	1.0	1.5	1.6
FN-18 dataset	Accuracy	**91.43**	91.33	90.93	91.13	90.47
	Precision	**91.21**	90.55	89.81	90.4	90.62
	Recall	**92.01**	91.91	91.91	92.92	91.61

Table 3. *Results of parameter tuning of the **FN-18** dataset*

Figure 6 displays the accuracy values during the validation process on the **FN-18** data-set for C parameter tuning.

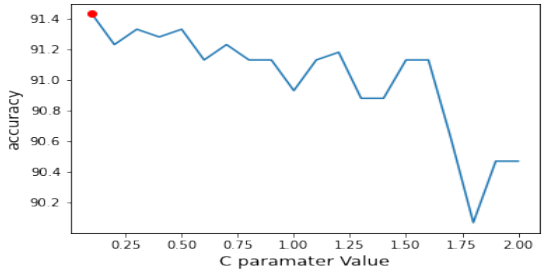

Fig. 6. *C testing on the **FN-18** data-set*

Then the best parameter, 1.6 for the **KDD-20** dataset and 0.1 for the **FN-18** dataset, is applied to the other 40%, the train dataset, and tested on the remaining 20%. The test produced the following outcomes presented in the table 4.

	KDD-20 data-set	FN-18 data-set
Accuracy	62.15	91.94
Precision	60.14	93.89
Recall	63.92	89.74

Table 4. *Results of C after the parameter tuning of the **KDD-20** and **FN-18** datasets*

Figures 7, and 8 represent confusion matrix to each dataset **KDD-20** and **FN-18** datasets after the parameter tuning.

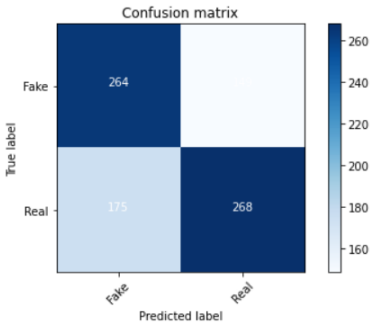

Fig. 7. The confusion matrix of the **KDD-20** dataset after C parameter tuning

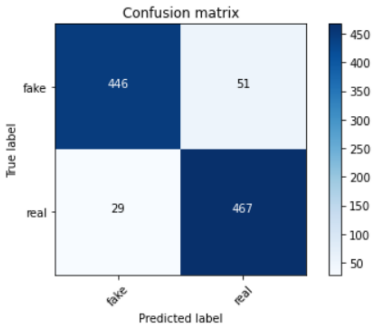

Fig. 8. The confusion matrix of the **FN-18** dataset after C parameter tuning

As shown in the table 4 the accuracy has augmented for the **FN-18** dataset after applying the best parameters comparing to the accuracy shown in table 3 but has decreased in the **FN-18** dataset comparing to the accuracy shown in table 2.

Maximum iteration

The second test will evaluate the accuracy of the **KDD-20** and **FN-18** datasets with the values of maximum iteration increasing by 100 from 100 to 2000. In this evaluation also, different values of max iteration were tested on 40% of each dataset, called the validation dataset. For the **FN-18** data-set, 5000 lines were taken to parameter tuning.

Table 5 shows the results of parameter tuning of Maximum iteration in the **KDD-20** dataset.

	Maximum iteration	100	**200**	500	1000	1500
KDD-20 dataset	Accuracy	63.79	**63.9**	63.02	63.49	63.67
	Precision	64.99	**62.78**	65.44	64.08	67.32
	Recall	56.22	**53.18**	55.96	56.08	56.54

Table 5. *Results of parameter tuning of maximum iteration in the **KDD-20** dataset*

Figure 9 displays the accuracy values during the validation process on the **KDD-20** data-set for maximum iteration parameter tuning.

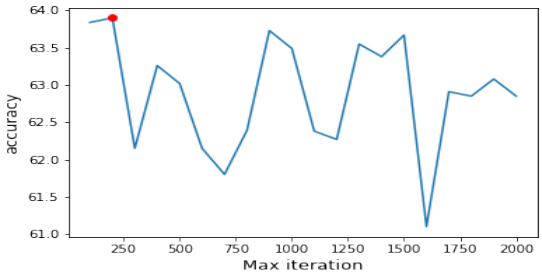

Fig. 9. *Max-iter testing on the* ***KDD-20*** *dataset*

Table 6 shows the results of parameter tuning of Maximum iteration in the **FN-18** dataset.

	Maximum iteration	100	**400**	500	1000	1500
FN-18 dataset	Accuracy	91.19	**91.43**	90.72	91.28	90.62
	Precision	89.85	**90.4**	89.73	89.7	89.81
	Recall	91.70	**91.71**	91.5	92.01	92.01

Table 6. *Results of parameter tuning of maximum iteration in the* ***FN-18*** *dataset*

Figure 10 displays the accuracy values during the validation process on the **FN-18** data-set for maximum iteration parameter tuning.

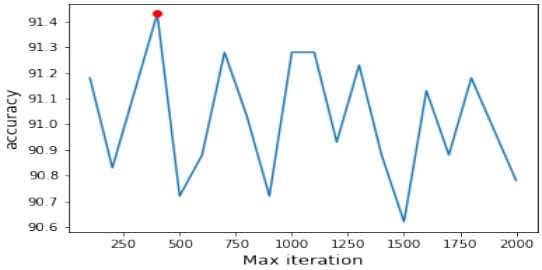

Fig. 10. *Max-iter testing on the* ***FN-18*** *dataset*

Then Applied the best parameter results on the other 40%, the train dataset, and then testing on the remain 20%. The test produced the follow outcomes presented in the table 7.

	KDD-20 dataset	**FN-18** dataset
Accuracy	61.68	90.84
Precision	58.95	93.38
Recall	67.80	87.93

Table 7. *Results of parameter tuning of maximum iteration in the* ***KDD-20*** *and* ***FN-18*** *datasets*

Figures 11 and 12 represent confusion matrix to each dataset **KDD-20** and **FN-18** datasets after the parameter tuning.

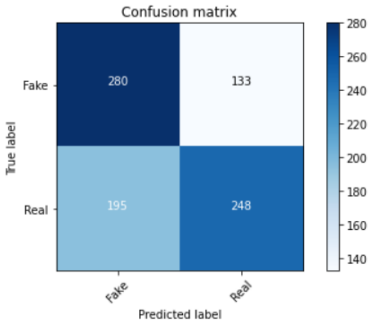

Fig. 11. The confusion matrix of the **KDD-20** dataset after max-iter tuning

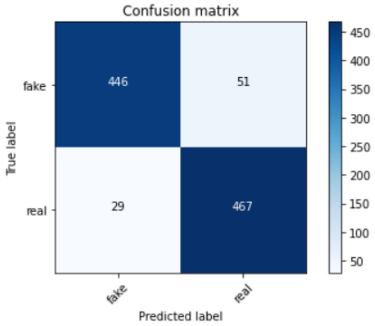

Fig. 12. The confusion matrix of the **FN-18** dataset after max-iter tuning

In the table 7 that the maximum iteration values that gave the best results didn't exceed 500.

The accuracy has decreased for the **KDD-20** and The **FN-18** datasets after applying the best maximum iteration parameters comparing to the accuracy shown in table 5 and table 6.

4.4 Comparison using Different Feature Extraction Techniques

TFIDF was used as a vectorization technique in the last parameter tuning part. In this part,the three different vectorization techniques will be tested on the the **FRN-17** and **FN-18** datasets to see if the vectorization technique impact the accuracy.

When applying the feature extraction technique TFIDF, Word2Vec, and GloVe on the **FN-18** dataset, it gives these results that are shown in the table 8 and each technique used has the confusion matrix corresponding to it shown successively in figures 13, 14, and 15.

	TFIDF	Word2Vec	GloVe
Accuracy	94.6	61.52	48.92
Precision	94.92	57.56	48.87
Recall	94.55	80.81	99.64

Table 8. *The comparison between TFIDF, Word2Vec, and GloVe in the* ***FN-18*** *dataset*

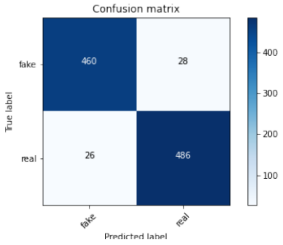

Fig. 13. The TFIDF confusion matrix with the **FN-18** dataset

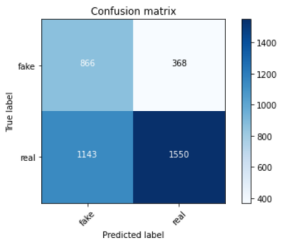

Fig. 14. The Word2Vec confusion matrix with the **FN-18** dataset

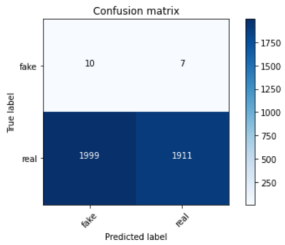

Fig. 15. The GloVe confusion matrix with the **FN-18** dataset

As shown in the table 8 as well as the result in the figures 13, 14, and 15, TFIDF technique gives the highest accuracy in comparison to Word2Vec and GloVe techniques.

For the **FRN-17** dataset applying the feature extraction technique TFIDF, Word2Vec and GloVe gave these results that are shown in the table 9 and each method used has the confusion matrix corresponding to it shown successively in figures 16, 17, and 18.

	TFIDF	Word2Vec	GloVe
Accuracy	98.12	96.0	62.96
Precision	97.25	95.22	89.86
Recall	97.62	96.99	30.55

Table 9. *The comparison between TFIDF, Word2Vec, and GloVe in the* ***FRN-17*** *dataset*

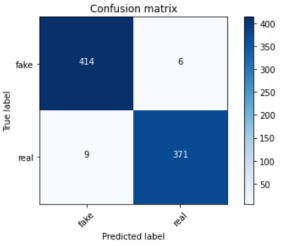

Fig. 16. The TFIDF confusion matrix with the **FRN-17** dataset

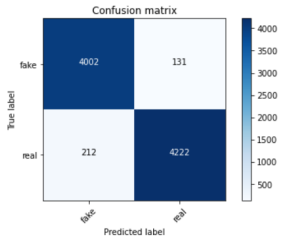

Fig. 17. The Word2Vec confusion matrix with the **FRN-17** dataset

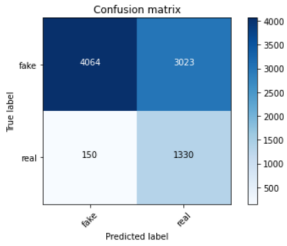

Fig. 18. The GloVe confusion matrix with the **FRN-17** dataset

In the **FRN-17** dataset as well, the TFIDF technique gives the best outcomes. However, Word2Vec also achieved an excellent accuracy in comparison

to the GloVe technique as shown in the table 9 and in the figures 16, 17.

TFIDF technique is the best technique based on the results shown. This is explained by the TFIDF approach, which depends on the frequency of the term appearing in the whole dataset without paying attention to the meaning of the term or the phrase context.

The word2vec learns the local vocabulary of the dataset, making new terms difficult to understand or identify their meaning, thus giving inconsistent results. The GloVe technique needs deep prior training to figure out and be able to identify the global vocabulary successfully, which is what it didn't receive in this research resulting in poor predictions.

4.5 Evaluation with A Large Dataset

The PA will be compared to the XGboost classifier in a large dataset in this test.

eXtreme Gradient Boosting, or xgboost, is an implementation of gradient boosting by [15]. Xgboost is a library that includes an efficient linear model solver and a tree learning algorithm. Due to its performance and speed, xgboost is an excellent option for evaluation. The comparison will be performed with 100000 lines of the **FNS-18** dataset. Datasets are being sliced into smaller datasets of 5000 lines for the machine to handle.

The comparison produced the table 10. The figure 19 represents the accuracy-time ratio for the passive-aggressive and the XGboost classifiers during the test.

	PA	XGboost
Accuracy	84.1	83.2
Precision	86.30	86.78
Recall	80.53	78.49

Table 10. *The passive aggressive and the XGboost classifiers comparison in the **FNS-18** dataset*

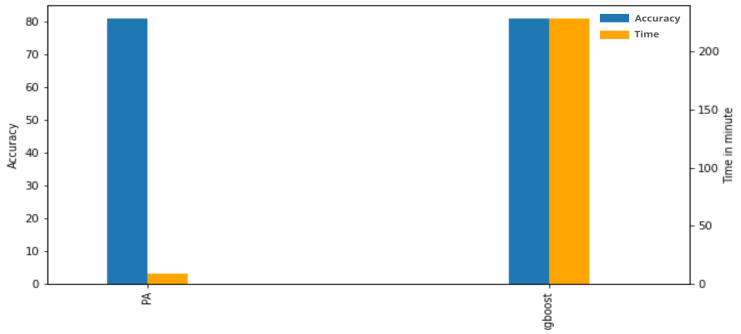

Fig. 19. *The passive aggressive and the XGboost classifiers comparison in the **FNS-18** with 100000 lines*

In the 100000 lines test, though, the passive-aggressive classifier achieved a higher accuracy. As shown in the table 10, the passive-aggressive classifier took less than 20 minutes to acquire these results. In contrast, the Xgboost classifier required more than two hours, as shown in the figure 19.

5 Conclusion

Upon experimenting with the PA classifier, it was tested on several datasets with severe sizes, used different feature extraction techniques, and compared its accuracy and required time an other well-known classifier. These experiments lead to mixed accuracy results with each dataset, technique, and others.

The experiments started by tuning the parameters to get the best results. Yet, it has been concluded that the obtained values of the parameters are different for each dataset and didn't improve the accuracy much compared with the default ones. Therefore, the default values can be used and should not necessarily be modified.

Further, the classifier produced its best result using the TFIDF vectorization technique, achieving up to 98.12% percent accuracy on the **FRN-17** dataset. However, using the same vectorization technique, the accuracy attained was 61.33% percent with the **KDD-20** dataset, so it can be said that the size of the dataset will make a difference when using the passive-aggressive. Also, it is found that the PA classifier can go through an extensive dataset in a reasonable amount of time and is fast compared to other classifiers that match its accuracy.

Regarding the feature extraction technique used, the PA classifier performed well with the TFIDF vectorization technique giving one of the best results, contrary to the word embedding technique GloVe, which can provide better results with further training as it is an unsupervised method.

The long-term focus is to create a global, effective algorithm for detecting false information and news that can use on social media platforms and news websites.

Bibliography

[1] tfidf. What does tf-idf mean?, 2022. http://www.tfidf.com/, visited 17-04-2022.

[2] H Ahmed, I Traore, and S Saad. Detecting opinion spams and fake news using text classification. *Journal of Security and Privacy*, 2018.

[3] Tim Pietz. Online machine learning. *University of Hamburg*, 2018.

[4] Saloni Gupta and Priyanka Meel. Fake news detection using passive-aggressive classifier. *Inventive Communication and Computational Technologies, Proceedings of ICICCT 2020*, pages 155–164, 2021.

[5] Jeffrey Huang. Detecting fake news with machine learning. *Journal of Physics: Conference Series*, 2020.

[6] Kaggle. Fake news, 2018. https://www.kaggle.com/c/fake-news, visited 17-05-2022.

[7] Kaggle. Fake news detection challenge kdd 2020, 2020. https://www.kaggle.com/c/fakenewskdd2020, visited 17-05-2022.

[8] Jasmine Shaikh and Rupali Patil. Fake news detection using machine learning. *IEEE Xplore*, 2020.

[9] Japneet Singh Chawla. What is glove?, 2018. https://medium.com/analytics-vidhya/word-vectorization-using-glove-76919685ee0b, visited 18-04-2022.

[10] Guilherme Pontes. Fake news sample, 2018. https://www.kaggle.com/datasets/pontes/fake-news-sample, visited 17-05-2022.

[11] Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. Online passive-aggressive algorithms. *Department of Computer and Information Science*, 2006.

[12] Great Learning. What is word embedding word2vec glove, 2020. https://www.mygreatlearning.com/blog/word-embedding/#sh4, visited 18-04-2022.

[13] Jason Brownlee. What are word embeddings for text?, 2017. https://machinelearningmastery.com/what-are-word-embeddings/, visited 07-03-2022.

[14] Victor Lavrenko. Text classification: Passive aggressive algorithm, 2014. https://youtu.be/TJU8NfDdqNQ, visited 31-03-2022.

[15] Jerome H Friedman. Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 2001. Accessed 16 May 2022.

[16] Thushan Ganegedara. Intuitive guide to understanding glove embeddings, 2019. https://towardsdatascience.com/light-on-math-ml-intuitive-guide-to-understanding-glove-embeddings-b13b4f19c010, visited 18-04-2022.

[17] Giuseppe Bonaccorso. Ml algorithms addendum: Passive aggressive algorithms, 2017. https://www.bonaccorso.eu/2017/10/06/ml-algorithms-addendum-passive-aggressive-algorithms/, visited 16-02-2022.