COVID-19 Fake News Detection Using Joint Doc2Vec and Text Features with PCA

Hector Mejia[1,2][0000−0002−6635−3189], Carlos Chipantiza[1][0000−0002−0832−1840], Jose Llumiquinga[1][0000−0002−0506−9862], and Isidro R. Amaro[1][0000−0003−2402−910X]

[1] Yachay Tech
School of Mathematical and Computational Sciences, San Miguel de Urcuqui-100119. Imbabura - Ecuador
[2] Nodel - Data Science Department, Guayaquil, Guayas - Ecuador

Abstract. With the current pandemic, it is imperative to stay up to date with the news and many sources contribute to this purpose. However, there is also misinformation and fake news that spreads within society. In this work, a machine learning approach to detect fake news related to COVID-19 is developed. Specifically, Doc2Vec language model is used to transform text documents into vector representations, and handcrafted features like document length, the proportion of personal pronouns, and punctuation are included as complimentary features as well. Then, Principal Component Analysis (PCA) is performed on the original feature vectors to reduce dimensionality. Both the original and reduced data are fed to various machine learning models and finally compared in terms of accuracy, precision, recall, and execution time. The results indicate that the reduced set of features had minimal accuracy impact, with 4% reduction. However, the execution times are greatly reduced in most cases, specifically at testing time, indicating that dimensionality reduction can be useful on projects already in production that would need model inference on large volumes of documents to detect fake news.

Keywords: principal components, text classification, fake news.

1 Introduction

As a result of the COVID-19 pandemic, multiple news articles started to emerge from multiple sources, and with it, misinformation, speculation, and fake news were disseminated as well. Since the internet plays a very important role in the propagation of information, news with false information can influence people either in their perception of the facts, generate social or economic problems, and manipulate public opinion, and change political outcomes [8]. With this current problem, people around the world have been influenced many times by the dissemination of information about the origin, spread, and treatment of this disease, as they have been left in a situation of helplessness or uncertainty about their future. Therefore, it is important to control the creation and dissemination of fake news content.

The most accurate way of detecting fake news is manual detection but it is a very slow process, making it unfeasible to tag large amounts of news. Nowadays with the widespread popularity of scientific computing and artificial intelligence, an scalable model for automatic detection of fake news using machine learning and feature engineering can be made. However, the text classification task needs vector representations from the textual documents, as well as discriminatory features, and with increasing vector dimensionality, training and inference times become slow and unpractical for real-time processing.

In this work, a method to detect COVID-19 related fake news is developed using feature extraction, normalization, dimensionality reduction, and classification. For the feature extraction component, a language model called Doc2Vec is employed to create vector representations from text documents. Moreover, a set of handcrafted features, derived from the lengths of the documents and punctuation, is concatenated to the previous document vectors to aid in the classification. These resulting feature vectors are normalized by subtracting the mean vector to each instance and dividing by the standard deviation.

Furthermore, Principal Component Analysis is employed to reduce the dimensionality of the features, while minimizing performance reduction. The purpose of adding this procedure is to reduce inference time when using classification models in production environments.

Then, a set of the most common machine learning algorithms are trained and tested on the original features, as well as the reduced components on two separate experiments to classify truthful and fake news. Specifically, the models employed are: Stochastic Gradient Descent optimized linear model, Support Vector Classifier, Random Forest, AdaBoost, and K-Nearest Neighbors.

Finally, training and testing performance metrics, as well as time, are compared between the models that were developed using the original features and the models built with the reduced principal components.

2 Previous Work

The pandemic of COVID-19 presented multiple opportunities in research across many areas, and sciences related to digital information are no exception. Many works explored the task of fake news detection related to this current world problem using machine learning approaches.

Some authors developed frameworks that employed traditional feature engineering and machine learning models. For instance, the work of Felber [4] utilized linguistic features like n-grams, emotional tone, and punctuation and machine learning models like support vector machines (SVM), linear regression (LR), random forest (RF), naive Bayes (NB) and multi-layer perceptron to classify fake and truthful reviews. This work achieved accuracies ranging 90.79% to 95.70% with a dataset of 10700 news related to COVID-19 for the Constraint@AAAI2021 challenge [12].

Moreover, Shushkevich and Cardiff [17] proposed the use of an ensemble model consisting of bidirectional long short-term memory, support vector ma-

chine, logistic regression, Naive Bayes, and a combination of logistic regression and Naive Bayes. For the feature engineering phase, they used term frequency-inverse document frequency (TF-IDF) to transform text to vector representation and consequently feed the model for training and testing. This work achieved an F1-score of 0.94 using the same dataset of the Constraint@AAAI2021 challenge.

Other authors harnessed the state of the art in natural language processing (NLP), using transformers in their frameworks. Transformers are deep learning-based language models that are pretrained on large corpora of text and then fine-tuned for the desired task, like text classification.

The work of Bang *et al.* [2] explored two methodologies for fake news detection. The first is fine-tuning ALBERT-base [9], BERT-base, BERT-large [3], RoBERTa-base, and RoBERTa-large [10] with different loss functions, and the second is data cleansing using an influence score, inspired by the work of Kobayashi *et al.* [7]. Finally, these two approaches achieved weighted f1-scores of 98.13% using the dataset for the Constraint@AAAI2021 challenge and 38.18% weighted-F1 on the Tweets-19 dataset [1].

Furthermore, Vijjali *et al.* [18] proposed a two-stage transformer model to detect fake news related to COVID-19. The first stage retrieves the most relevant facts concerning user claims about particular COVID-19 news, while the second verify the level of truth in those claims by computing the textual entailment between the claim and the facts retrieved from a manually curated dataset. This dataset is based on a publicly available information source consisting of more than 5000 COVID-19 false claims and verified explanations. As for the results, this work achieved 85.5% accuracy on the aforementioned dataset using BERT+ALBERT as the transformer models for the two-stage detector.

3 Materials and Methods

In this section, the employed dataset, as well as the steps taken to develop the fake news classification model are described in detail.

3.1 Dataset Description

The Dataset used in this work is called COVID19-FNIR and was downloaded from the IEEE DatePort. It contains two sets of text news, collected between February 2020 and June 2020, that are related to the COVID-19 global pandemic, one set corresponding to verified truthful news, and the other is a compilation of fake news. This data was gathered from sources in multiple regions: India, the United States of America and various countries of Europe. Moreover, it should be added that this data does not contain special characters or information that is not considered non-vital [15].

The dataset contained a total of 7588 examples with each class consisting of exactly half of that total. Fig. 1 depicts the distribution of classes.

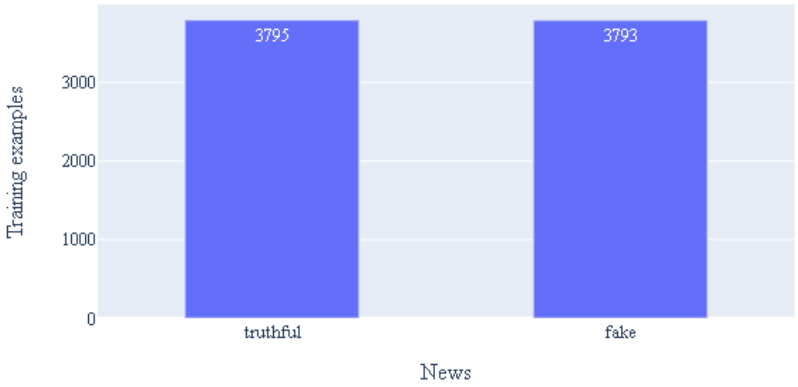

Fig. 1. Distribution of classes

.

3.2 Data Processing

To develop a model to detect COVID-19 related fake news, textual data has to be preprocessed. In this work, three stages of data processing that will allow us to perform training of various machine learning models are defined. These stages are dataset consolidation, feature extraction, and feature normalization.

The original dataset consisted of two files. One file contained all verified truthful news collected from many sources, while the other contained fake news that was manually labeled by the author of the dataset. Furthermore, these files were loaded and concatenated. In addition, an integer label was assigned to each example: 1 to denote "fake news" and 0 to denote "truthful news".

After the dataset is consolidated and tagged a feature extraction is performed on the textual information. Since machine learning models can only use numerical data to learn patterns, a transformation has to be carried on the text corpus to vector representations.

Doc2Vec is a neural network that was designed for such a purpose. It is trained for language modeling, i.e predicting words given previous words that are used as context. Given a sequence of training words and paragraph ids $w_1, w_2, w_3, ..., w_T$, where T is the total number of words in the vocabulary plus the number of paragraphs, the objective is to maximize the average log probability:

$$\frac{1}{T}\sum_{t=k}^{T-k} logp\left(w_t|w_{t-k}, ..., w_{t+k}\right)$$

In this framework, every word is mapped to a unique vector, represented by a column in a matrix W. The column is indexed by the position of the word in the vocabulary. Moreover, these vectors are averaged or concatenated and used

as features for the prediction of the next word. This prediction is done by adding a softmax classification layer. Therefore:

$$p\left(w_t|w_{t-k},...,w_{t+k}\right) = \frac{e^{y_{w_t}}}{\sum_i e^{y_i}}$$

Then each y_i is computed as:

$$y = b + Uh\left(w_{t-k},...,w_{t+k};W\right),$$

where U, b are the softmax parameters and h is the concatenation or average of each word/paragraph vector. Once the language model is trained, the softmax layer was removed and retain only the embeddings of each each document.

In addition to the document vectors, manual features were computed from the text, as seen in the work of Rayana *et al.* [13]. Specifically, the length of each document was calculated in words L, the ratio of exclamation sentences RES, the ratio of first personal pronouns $PP1$ and the percentage of capital letters PC. Finally, these features were concatenated to each document vector to produce the final feature vectors.

After the features are computed, feature normalization is performed since the resulting document embeddings present different scales across its dimensions. This is done by simply subtracting the mean u and dividing by the standard deviation s:

$$z = \frac{x - u}{s}$$

These feature vectors have numbers in each dimension, in a range from 0 to 1. Furthermore, a correlation matrix was performed, as depicted in Figure 2, in this data for further dimensionality reduction techniques.

The aforementioned procedure was implemented using python 3.8 with the help of scikit-learn and gensim libraries for feature extraction and matplotlib for plotting. The implementation details can be seen in Algorithm 1.

3.3 Principal Component Analysis

One of the main objectives when analyzing high-dimensional data is to obtain a representation in a reduced space that demonstrates the main patterns of similarity between individuals. This problem can be addressed by using multivariate techniques such as the Principal Component Analysis (PCA) [6, 14].

In other words, principal component analysis reduces a large number of multivariate variables into a relatively small number of linear combinations of these that can be used to account for much of the variability in the data. The central idea of PCA is to reduce the dimensionality of a data set with minimal information loss. [14].

To decide how many components to work with, considered at least 80% [14] which is possible with 10 components (82%).

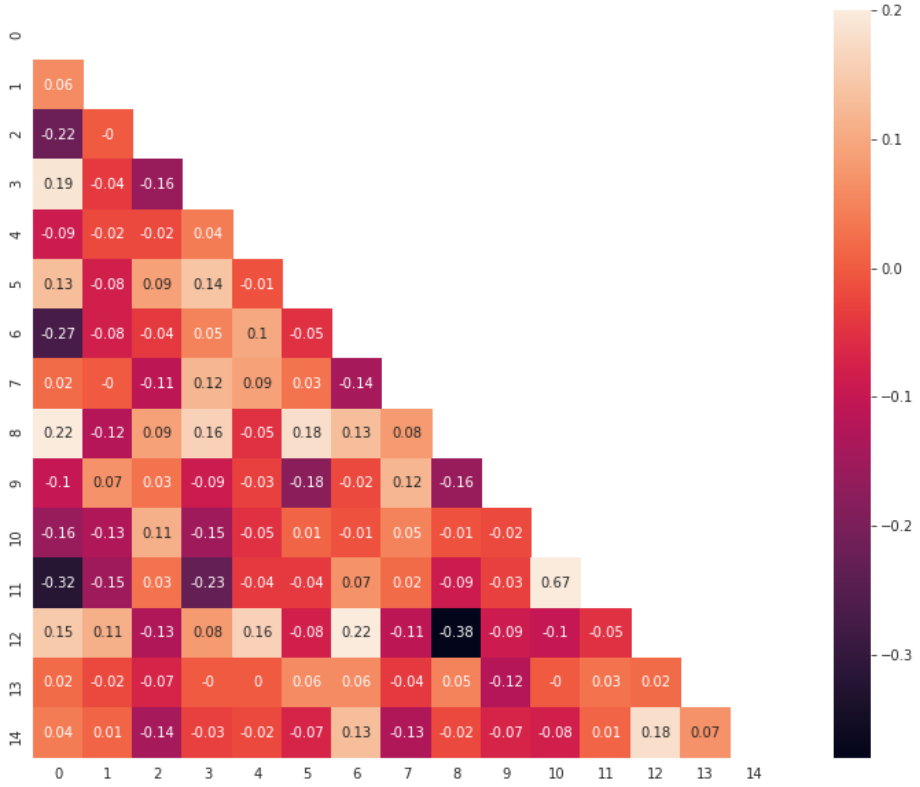

Fig. 2. Correlation matrix from the computed features.

3.4 Machine Learning Training and Testing

At this stage, five different supervised machine learning algorithms are trained and tested for fake news classification on two different settings. The first setting uses the original set of features, while the second uses the ten first principal components. The procedure was implemented using python 3.8 with the help of scikit-learn to train various machine learning models. The implementation details can be seen in Algorithm 2, the same algorithm was used with the total features and with the features after PCA. Moreover, the machine learning algorithms are briefly described below:

SGD Classifier: Stochastic Gradient Descent (SGD) Classifier is a classification algorithm that works with regularized linear models and gradient descent learning (SGD), hence the reason for its name. In this algorithm the error gradient is estimated each sample at a time and the model is updated over time with a learning rate [11].

Algorithm 1: Feature extraction implementation

Data: Truthful news directory d_T, Fake news directory d_F
Result: Scaled train and test features datasets S_{train}, S_{test}, train and test principal components PC_{train}, PC_{test}

$D_T, L_T \leftarrow load\_data\_and\_labels(d_T)$;
$D_F, L_F \leftarrow load\_data\_and\_labels(d_F)$;
$D \leftarrow D_T \cup DF$;
$L \leftarrow L_T \cup LF$;
$D_{train}, L_{train}, D_{test}, L_{test} \leftarrow train\_test\_split(D, L, test\_size = 20\%)$;
$M_E \leftarrow train\_Doc2Vec\_LM(D_{train})$;
$E_{train}, E_{test} \leftarrow encode\_documents(D_{train}, D_{test}, M_E)$;
$f_{1train}, f_{1test} \leftarrow get\_documents\_length(D_{train}, D_{test})$;
$f_{2train}, f_{2test} \leftarrow get\_exclamation\_ratio(D_{train}, D_{test})$;
$f_{3train}, f_{3test} \leftarrow get\_caps\_percentage(D_{train}, D_{test})$;
$f_{4train}, f_{4test} \leftarrow get\_1stpersonalpronoun\_ratio(D_{train}, D_{test})$;
$f_{train} \leftarrow concatenate(f_{1train}, f_{2train}, f_{3train}, f_{4train})$;
$f_{test} \leftarrow concatenate(f_{1test}, f_{2test}, f_{3test}, f_{4test})$;
$F_{train} \leftarrow concatenate(f_{train}, E_{train})$;
$F_{test} \leftarrow concatenate(f_{test}, E_{test})$;
$M_S \leftarrow fit\_standard\_scaler(F_{train})$;
$S_{train}, S_{test} \leftarrow scale\_features(F_{train}, F_{test}, M_S)$;
$M_{PC} = fit\_PCA(S_{train})$;
$PC_{train}, PC_{test} =$
$get\_principal\_components(S_{train}, S_{test}, M_{PC}, min_variance = 80\%)$;

Support Vector Machine Classifier (SVM): Builds a model that finds the best separation between classes and assigns new examples to one or the other category. It does this by separating the classes into two spaces by means of a separation hyperplane that is defined as a vector between the two classes [5].

Random Forest Classifier: Uses the fundamentals of decision trees, except that the Random forest uses many decision trees that it builds at training time and makes predictions by averaging the predictions of each component tree. It generally has a much higher prediction accuracy than a single decision tree and works well with predetermined parameters [11].

AdaBoost Classifier: This algorithm is part of boosting algorithms. Boosting is an approach that relies on creating an accurate prediction rule by combining many relatively weak and inaccurate rules, also the boosting algorithm can track the model that failed in accurate prediction. AdaBoost stands for Adaptive Boosting and uses 1-level decision trees as weak rules that are sequentially added to the ensemble, the importance of the tree in the final classification is calculated, then the weights are updated so that the next decision tree takes into account the errors made by the previous decision tree [16].

K Neighbors Classifier: Implements the nearest neighbor algorithm where the input is the number of k nearest training examples in the feature space.

This algorithm searches the observations closest to the observations it is trying to predict and classifies a point based on the surrounding points[11].

After training, the predictions are extracted using the fitted models and the test sets from both experimental settings. These predictions will be compared against true labels, unseen by the models, to compute various metrics to assess performance. These metrics are precision, recall, f1-score, and accuracy.

Precision is the ability of the classifier not to label as positive an observation that is negative and is given by:

$$\text{Precision } = \frac{tp}{tp + fp},$$

where *tp* are the True positives and *fp* are the False Positives.

Recall is the ability of the classifier to find the positive samples and is defined by:

$$\text{Recall } = \frac{tp}{tp + fn},$$

where *tp* are the True positives and *fn* are the False Negatives.

The f1-score is a weighted harmonic measure of precision and recall, where at 1 it reaches its best value and at 0 it reaches its worst value. It is given by:

$$F_1 = 2\frac{\text{Precision } * \text{ Recall}}{\text{Precision } + \text{ Recall}}$$

Finally, the accuracy measures the ratio of correct predictions against all the instances of a test set and it is formulated as:

$$Acc = \frac{tp + tn}{tp + tn + fp + fn}$$

Algorithm 2: Machine Learning Models

Data: Train and Test features datasets F_{train}, F_{test}
Result: Metrics
$modelset \longleftarrow \{SGD, SVM, RF, AdaBoost, KNN\}$;
for $model \in modelset$ **do**
  $model\_training(x_{train}, y_{train})$;
  $TrainingTime \longleftarrow get\_execution\_time$;
  $y_{preds} \longleftarrow model\_predict(x_{test})$;
  $ClassificationTime \longleftarrow get\_execution\_time$;
  $get\_classification\_report(y_{test}, y_{preds})$;
end

4 Results and Discussion

Table 1 shows the run times for training and testing. In general, it can be seen that the training and prediction times of the models are shorter after having performed the PCA analysis, except for two models which are the Support Vector Machines and the Random Forest Classifier. It would be an important topic for future work to treat these two models in more detail, under the influence of of PCA analysis, to examine why the execution times increased after dimensionality reduction.

Table 1. Performance based on Time.

	Before PCA		After PCA	
Machine Learning Method	Train Time	Test Time	Train Time	Test Time
SGDClassifier	0.0608	0.0081	0.0379	0.00097
Support Vector Machines	0.0607	0.1146	0.9964	0.065
Random Forest Classifier	0.0622	0.0431	1.8470	0.0371
AdaBoost Classifier	1.5708	0.0411	1.3162	0.0336
KNeighbors Classifier	1.6498	0.0548	0.0132	0.0963

Tables 2 and 3 show the precision, recall, and F1-score for the two classes of the COVID-19 news dataset, i.e. the labels 0 for truthful and 1 for fake news. Comparing these two tables containing performance metrics before and after PCA shows that, in general, the dimensionality reduction produced by PCA has a minimal influence over performance. It can be observed that the precision, recall, and F1-score metrics after PCA indicate that most of the information needed to correctly predict news about COVID-19 is still present.

Table 2. Performance Parameters before PCA.

	Before PCA					
	Precision		Recall		F1-score	
Machine Learning Method	0	1	0	1	0	1
SGDClassifier	0.85	0.84	0.83	0.86	0.84	0.85
Support Vector Machines	0.84	0.89	0.89	0.82	0.86	0.85
Random Forest Classifier	0.86	0.84	0.83	0.86	0.85	0.85
AdaBoost Classifier	0.81	0.86	0.87	0.79	0.84	0.83
KNeighbors Classifier	0.84	0.86	0.86	0.83	0.85	0.84

In terms of accuracy, the models using the full dimensionality of the features have a higher number of correctly classified elements than the models using the

Table 3. Performance Parameters after PCA.

	After PCA					
	Precision		Recall		F1-score	
Machine Learning Method	0	1	0	1	0	1
SGDClassifier	0.74	0.85	0.88	0.68	0.80	0.76
Support Vector Machines	0.79	0.85	0.86	0.77	0.82	0.81
Random Forest Classifier	0.78	0.83	0.84	0.77	0.81	0.80
AdaBoost Classifier	0.77	0.84	0.86	0.74	0.81	0.78
KNeighbors Classifier	0.77	0.84	0.86	0.75	0.81	0.79

features after a PCA, as shown in Figure 3, but the difference is still very minimal, which means that using the same models, but with a lower computational cost can still rely on the reduced features for classification tasks.

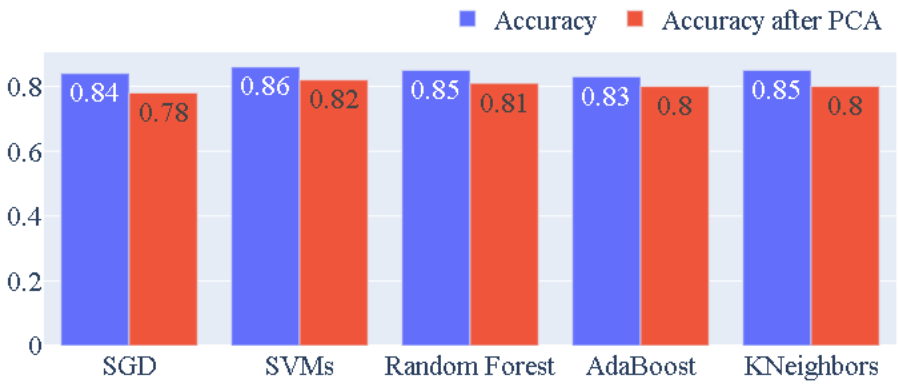

Fig. 3. Performance based on Accuracy

5 Conclusion

To detect fake news related to COVID-19, using the COVID19-FNIR Dataset, the effectiveness of the Principal Component Analysis (PCA) procedure was analyzed against several Machine Learning models, widely known in the field for classification tasks, both before and after applying dimensionality reduction. When compared to the performance evaluation using the original set of features, in terms of accuracy, the PCA analysis only resulted in a reduction of 4%, maintaining high performance for every machine learning model. Furthermore, The experiments also show that machine learning methods together with Principal Component Analysis are more efficient, in terms of COVID-19-related news classification than only using the original features, since the training and evaluation

times of the models were greatly reduced while maintaining high-performance metrics. This reduction in time would be greater, when the training corpus becomes much larger, or when is used in production environments for inference. In addition, the Doc2Vec model with PCA for dimensionality reduction allows keeping a large amount of feature information, maintaining good accuracy, and reducing training and test execution time for classification tasks. This is useful since detecting false news in near-real-time helps to mitigate many social problems, problems that have to do with the spread of false information about COVID-19. Finally, this work contributes to the research about fake news and helps to identify fake news related to COVID-19, for future work an attempt will be made to explore other dimensionality reduction techniques that allow better performance, in combination with machine learning methods.

References

1. Alam, F., Dalvi, F., Shaar, S., Durrani, N., Mubarak, H., Nikolov, A., Martino, G.D.S., Abdelali, A., Sajjad, H., Darwish, K., Nakov, P.: Fighting the covid-19 infodemic in social media: A holistic perspective and a call to arms (2021)
2. Bang, Y., Ishii, E., Cahyawijaya, S., Ji, Z., Fung, P.: Model Generalization on COVID-19 Fake News Detection (jan 2021), http://arxiv.org/abs/2101.03841
3. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding (oct 2018), http://arxiv.org/abs/1810.04805
4. Felber, T.: Constraint 2021: Machine Learning Models for COVID-19 Fake News Detection Shared Task (jan 2021), http://arxiv.org/abs/2101.03717
5. Fletcher, T.: Support vector machines explained (01 2009)
6. Jolliffe, I.: Methods of Multivariate Analysis. Springer (2002), https://www.springer.com/gp/book/9780387954424
7. Kobayashi, S., Yokoi, S., Suzuki, J., Inui, K.: Efficient Estimation of Influence of a Training Instance. In: Proceedings of SustaiNLP: Workshop on Simple and Efficient Natural Language Processing. pp. 41–47. Association for Computational Linguistics, Stroudsburg, PA, USA (2020). https://doi.org/10.18653/v1/2020.sustainlp-1.6, https://www.aclweb.org/anthology/2020.sustainlp-1.6
8. Koirala, A.: Covid-19 fake news classification with deep learning (10 2020). https://doi.org/10.13140/RG.2.2.26509.56805
9. Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., Soricut, R.: ALBERT: A Lite BERT for Self-supervised Learning of Language Representations (sep 2019), http://arxiv.org/abs/1909.11942
10. Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., Stoyanov, V.: RoBERTa: A Robustly Optimized BERT Pretraining Approach (jul 2019), http://arxiv.org/abs/1907.11692
11. Paper, D.: Hands-on Scikit-Learn for Machine Learning Applications: Data Science Fundamentals with Python. Apress (2019), https://books.google.com.ec/books?id=kqy-DwAAQBAJ
12. Patwa, P., Sharma, S., Pykl, S., Guptha, V., Kumari, G., Akhtar, M.S., Ekbal, A., Das, A., Chakraborty, T.: Fighting an Infodemic: COVID-19 Fake News Dataset. pp. 21–29 (nov 2021). https://doi.org/10.1007/978-3-030-73696-5_3, http://arxiv.org/abs/2011.03327 http://dx.doi.org/10.1007/978-3-030-73696-5_3 https://link.springer.com/10.1007/978-3-030-73696-5_3

13. Rayana, S., Akoglu, L.: Collective Opinion Spam Detection. In: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 985–994. ACM, New York, NY, USA (aug 2015). https://doi.org/10.1145/2783258.2783370, https://dl.acm.org/doi/10.1145/2783258.2783370
14. Rencher, A.C.: Methods of Multivariate Analysis (2002). https://doi.org/10.1002/0471271357
15. Saenz, Julio A. y Kalathur Gopal, S.R.y.S.D.: Covid-19 fake news infodemic research dataset (covid19-fnir dataset). https://doi.org/10.21227 / b5bt-5244, https://dx.doi.org/10.21227/b5bt-5244
16. Schapire, R.E.: Explaining AdaBoost, pp. 37–52. Springer Berlin Heidelberg, Berlin, Heidelberg (2013). https://doi.org/10.1007/978-3-642-41136-6_5, https://doi.org/10.1007/978-3-642-41136-6_5
17. Shushkevich, E., Cardiff, J.: TUDublin team at Constraint@AAAI2021 – COVID19 Fake News Detection (jan 2021), http://arxiv.org/abs/2101.05701
18. Vijjali, R., Potluri, P., Kumar, S., Teki, S.: Two Stage Transformer Model for COVID-19 Fake News Detection and Fact Checking (nov 2020), http://arxiv.org/abs/2011.13253