Perspectives in Communication, Embedded-Systems and Signal-Processing (PiCES) – An International Journal
ISSN: 2566-932X, Vol. 5, Issue 6, September 2021

Classifying Fake News Articles Using Machine Learning Techniques

Doddi Srilatha
Computer Science and Engineering, Sreenidhi Institute of Science and Technology, Hyderabad, India

Dustakar Prasanth Rao
Computer Science and Engineering, Sreenidhi Institute of Science and Technology, Hyderabad, India

Chukka Nikhil
Computer Science and Engineering, Sreenidhi Institute of Science and Technology, Hyderabad, India

Janumpally Sai Teja
Computer Science and Engineering, Sreenidhi Institute of Science and Technology, Hyderabad, India

Abstract: ***News is important in daily life as it informs our view to the world and in response, we take actions and make choices in various aspects. Gradually, the tendency towards online news is increasing as it is more concise and is available at the finger tips. There has been large generation of deceptive content worldwide that has an effect on the formation of opinions, making decisions and voting trends. Most of the 'fake news' is initially circulated through social media networks such as Twitter, Facebook, and then makes the way into mainstream media outlets such as Radio and TV. The fake news articles share linguistic features such as heavy use of quoted material. In this use case, the results of fake news detection test and the performance of fake news classifiers is discussed. The aim is to build a new fake news detector using classifiers like Logistic Regression, Decision tree classification, Multinomial Naive Bayes classification.***

Keywords: Classifiers; Confusion Matrix; Count Vectorizers; Machine Learning; Exploratory Data Analysis

I. INTRODUCTION

Many classification algorithms such as Bayesian classification, Logistic Regression, Decision tree classification have been applied for the fake news detection. The hyper-parameters of the classifiers have been selected based on accuracy scores. Hyper-parameter tuning is one of the most promising approaches to improve the performance the classifiers. The Logistic regression is same as linear regression with a sigmoid function attached to its output in order to achieve binary classification. The tradeoff parameter that determines the strength of logistic regression is achieved through hyper-parameter tuning. A variant of Bayes classifier known as Multinomial Naive Bayes classifier is studied since it handles good when the data is discretely distributed. The decision tree classifiers re based on the feature of Information gain and the depth of the decision tree is adjusted based on hyper-parameter tuning. Performance of different classifiers is evaluated against the metrics such as precision, accuracy. It is easy to differentiate between fake news and genuine news from online sources, which is very hectic task in previous studies.

II. DATA PREPROCESSING

Text pre-processing is an important activity in language processing tasks. It transforms text into a digestible form to enhance the working of machine learning algorithms. There are three main components of text preprocessing.

A. Normalization

Normalization aims to keep the data relatively in smaller range. In text pre-processing, Normalization attempts to place all texts on an equal footing. It converts all characters into either lowercase or uppercase.

B. Noise removal

Noise removal is elimination of extra spaces, special characters, numbers from text and also removal of rows with null values.

C. Tokenization

Tokenization is splitting paragraphs into sentences and splitting sentences into words.

D. Stemming

The method of creating morphological variants of basic/root word is known as Stemming. The Stemming algorithm will reduce the words such as “chocolates”, “chocolatey”, “choco” to the base word, “chocolate” and “retrieval”, “retrieved”, “retrieves” to the base “retrieve”.

Compared to stemming, lemmatization puts the terms into the context. It therefore links words to one word of

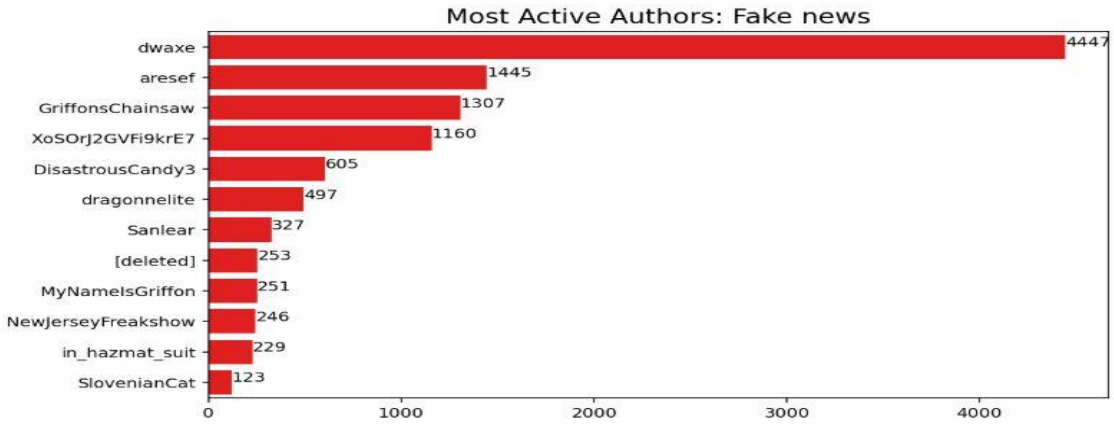

Fig 1. Most Active Authors of Fake News

similar meaning. Lemmatization is generally preferred to Stemming, since lemmatization analyses words in a morphological way.

III. EXPLORATORY DATA ANALYSIS (EDA)

Analyzing the data sets to identify and summarize the important correlated features usually with visual methods is known as exploratory data analysis. EDA consists of finding most active authors or publishers, most referenced domains, most frequent unigrams and bigrams.

A unigram is one and bigram is a sequence of two words. A statistical model is a distribution of probability over word sequences. Unigrams are helpful to differentiate between related words and phrases. Gappy bigrams are word pairs which allow gaps.

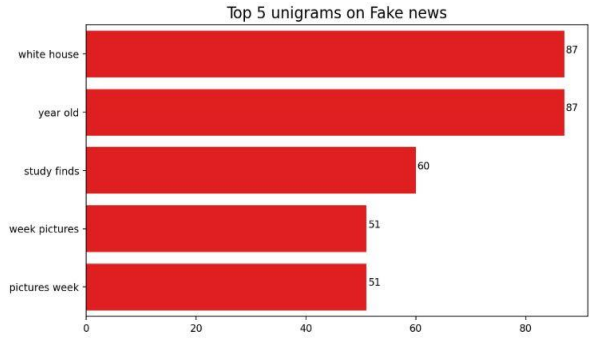

Fig 2. Top 5 Unigrams of Fake News

IV. HYPER-PARAMETER TUNING

The parameters of the model(classifier) that are to be adjusted according to the data are known as Hyper-parameters. GridSearch cross validation technique is one of the useful techniques in evaluating the hyper-parameters. The process of adjusting these hyper-parameters is known as hyper-parameter tuning.

The hyper-parameters differ from model to model. For example, Ngram range for vectorizer, alpha for Naïve Bayes, maximum depth for Decision tree classifier.

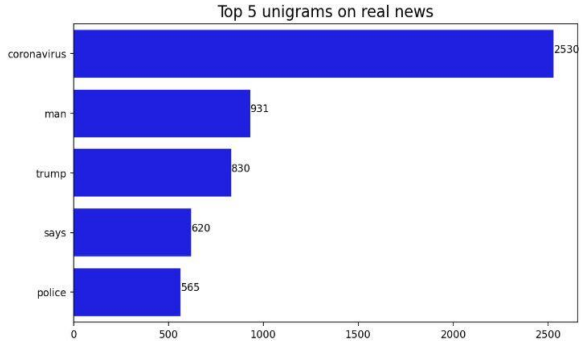

Fig 3. Top 5 Unigrams of Real News

V. FEATURE EXTRACTION

To delete words/terms is known as tokenization and for that the text must be parsed. Then the terms are represented as integer or floating values that are to be fed as input to the machine learning algorithm. This is known as Vectorization (or extraction) of functions. Algorithms take numeral vectors as input only. An easy and efficient model for doing this function is called the BagofWords model in machine learning. The classifier/model is not bothered in the words about the order detail and concentrates on the occurrences of the words in a text. A unique number is assigned to each title. Any text with the length of the vocabulary of the recognized terms can be represented as a fixed-length vector. Some of them are

A. CountVectorizer

The CountVectorizer helps tokenize a set of text documents and create recognized word vocabulary. It's also to use the language to encrypt new documents. You can use it as follows:

1) Creates a CountVectorizer class instance.
2) To acquire a vocabulary from one or more texts, call the function fit( ).
3) To encode each as a vector, call the function transform( ) on one or more documents.

The encoded vector has the length that of the entire vocabulary, and the number of times each word occurs in the text is an integer count. We call it as sparse vector as it contains lot of zeros.

B. *TFIDFVectorizer*

One problem with simple counts is common terms such as "a", "an", "the" etc. Appear several times, and their large numbers in the encoded vectors will not be very important. An alternate solution to this is TF-IDF. It is an abbreviation of "Term Frequency–Inverse Document Frequency".

Term Frequency: This sums up the presence of a word in a text.

Frequency of Inverse Document: This downscales the terms that occur many times across the document.

The TF-IDF Vectorizer will tokenize documents, studies the vocabulary and reverse the weighting of document frequencies and allow you to encrypt new documents. CountVectorizer can also be used with TF-IDF Transformer to measure the frequencies of the inverse documents and start encoding documents. This model aims to highlight more interesting words. The scores are standardized to values between 0 and 1 and like most machine-learning algorithms, the encoded text vectors can be used directly.

VI. CLASSIFIERS

The process of categorizing texts into organized groups is known as text classification or text tagging. Some of the classifiers that are used here are

1. Logistic Regression
2. Decision Tree
3. Naïve Bayes

A. *Logistic Regression*

The Logistic Regression is like the Linear Regression. In Linear Regression ,the input values(x) are used linearly to determine the value of the output(y) using the weights or coefficient values. A main distinction from linear regression is that the modeled output value is not a numerical value but a binary value (0 or 1).

$$y = \frac{e^{b_0+b_1x}}{1+e^{b_0+b_1x}} \quad (1)$$

Here b_0 is the bias or intercept term where y is the desired output and b_1 is the coefficient for the input value (x). – a column in your input data which has a coefficient b associated with it (a constant real value) calculating from your training data.

The corresponding coefficients (Beta values b) of the logistic regression have to be measured from your training data. It is understood using estimates of maximum likelihood.

The model is shown as,

$$P(X) = \frac{e^{b_0+b_1x}}{1+e^{b_0+b_1x}} \quad (2)$$

The threshold for the probability is fitted based on training data.

Output=0/1

Hypothesis: $Z = b_0X + b_1X$ (3)

h(x)=sigmoid(Z) (4)

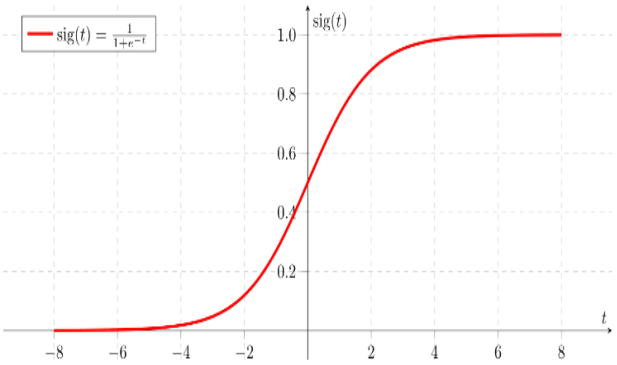

Fig 4. Sigmoid Function

SNO	Testcase Name	Hyper-parameters	Accuracy
1	CountVectorizer & Logistic Regression	ngram_range=(1,1) stop_words=None C=0.01	77.24%
2	CountVectorizer & Logistic Regression	ngram_range=(1,2) stop_words=None C=0.01	78.22%
3	CountVectorizer & Logistic Regression	ngram_range=(1,1) stop_words=english C=1	85.23%
4	CountVectorizer & Logistic Regression	ngram_range=(1,2) stop_words=english C=1	85.37%
5	TfidfVectorize& Logistic Regression	max_df=0.75 min_df=2 ngram_range=(1,1) C=0.5	82.89%
6	TfidfVectorize& Logistic Regression	max_df=0.75 min_df=2 ngram_range=(1,2) C=0.5	83.32%
7	TfidfVectorize& Logistic Regression	max_df=0.75 min_df=2 ngram_range=(1,1) C=1	84.39%
8	TfidfVectorize& Logistic Regression	max_df=0.75 min_df=2 ngram_range=(1,2) C=1	84.7%

Table 1. Performance of Logistic Regression with Count Vectorizer

B. *Decision Tree Classification*

Continuous splitting is known as the grouping of Decision tree according to certain parameter.

Decision Tree includes:

1. Nodes: Checking condition for a given attribute value.
2. Edges / branch: Edges match the result of a check and bind to the following node or leaf.
3. Leaf nodes: These are terminal nodes which forecast the result (represent class distribution).

Heuristic partitioning is used to construct the Decision Tree and the process is called recursive partitioning. This method is also described as dividing and conquering as it divides the data into sub-sets, that are instead divided recursively into smaller sub-sets, and so on and so forth until the cycle halts whenever the algorithm decides that the data inside the sub-sets is homogeneous enough or that any other stop requirement has been satisfied.

Algorithm

1. Decision tree algorithm begins with the root of the tree and breaks the data on the feature resulting in the greatest Information Gain (IG) (decrease in uncertainties towards final choice).
2. We can then perform this splitting process at every child node in an iterative cycle, until the leaves are not further divisible. It implies that the samples at each node of the leaf are all of the same class.
3. In implementation, we should set a cap on tree depth to prevent overfitting. We rely on pureness somewhat here since the final leaves may still be unclean.

SNO	Testcase Name	Hyper-parameters	Accuracy
1	TfidfVectorizer &DecisionTree Classifier	max_depth=22 ngram_range=(1,2)	74.36%
2	TfidfVectorizer &DecisionTree Classifier	max_depth=22 ngram_range=(1,3)	74.48%
3	TfidfVectorizer &DecisionTree Classifier	max_depth=25 ngram_range=(1,2)	74.31%
4	TfidfVectorizer &DecisionTree Classifier	max_depth=25 ngram_range=(1,3)	75%

Table 2. Performance of Decision tree classification with Tfidf Vectorizer

C. *Naïve Bayes Classifier*

Naive Bayes classifiers are the set of Bayes' Theorem-based classification algorithms. It is based on a universal definition, i.e. a pair of every characteristic to be categorized is independent of one other.

The basic principle of Naive Bayes is that every feature contributes equally and independently to the outcome.

Bayes' Theorem considers the probability of an occurrence happening given the likelihood of some other occurrence already taking place. Bayes' theorem is defined as following, mathematically:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \quad (5)$$

where A and B are events and P(B) not equal to 0.

Equation (5) explains, given that event B has happened, we can find the likelihood of event A occurring. Here, evidence is event B, and inference is event A.

P(A) is the priori of event A (previous probability, i.e. likelihood of occurrence before evidence is seen). The evidence is an undefined instance attribute value (here, it is case B).

P(A|B) is a posteriori probability of event B, i.e. probability of the event after evidence has been shown.

D. *Multinomial Naive Bayes*

Multinomial Naive Bayes is used when, in essence, data is discrete. The features / predictors the classifier uses are the frequency of the terms that appear in the text.

E.g.: Number of occurrences.

The mean of every word for a given class is determined to determine our likelihood.

The average listed for word i and class j is:

$$P(i \mid j) = \frac{word_{i\,j}}{word_j} \quad (6)$$

Since there are 0 terms, however, Laplace Smoothing is done with a low α:

$$P(i \mid j) = \frac{word_{i\,j} + \alpha}{word_j + |V| + 1} \quad (7)$$

where V is the array of all vocabulary terms and α=0.001.

Combination of P probability distribution with proportion of documents corresponding to each class.

For term/word i, class j at f frequency is:

$$\Pr(j)\ \alpha\ \pi_j \prod_{i=1}^{|V|} \Pr(i \mid j)^{f_i} \quad (8)$$

We'll use the number of logs to avoid underflow:

$$\Pr(j)\ \alpha\ log\left(\pi_j \prod_{i=1}^{|V|} \Pr(i \mid j)^{f_i}\right) \quad (9)$$

$$\Pr(j) = log\pi_j + \sum_{i=1}^{|V|} f_i \log\left(\Pr(i \mid j)\right) \quad (10)$$

One concern is that when a word appears again, the risk of it occurring again increases. To smooth out this, we take the frequency log:

$$\Pr(j) = log\pi_j + \sum_{i=1}^{|V|} \log(1 + f_i) \log(\Pr(i \mid j)) \qquad (11)$$

Also, we'll apply an Inverse Document Frequency (IDF) weight to every word to take stop words into account:

$$Sum = \sum_{n=1}^{N} doc_n$$

$$t_i = \log\left(\frac{Sum}{doc_i}\right)$$

$$\Pr(j) = log\pi_j + \sum_{i=1}^{|V|} f_i \log(t_i \ \Pr(i \mid j)) \qquad (12)$$

SNO	Testcase Name	Hyper-parameters	Accuracy
1	CountVectorizer &MultinomialNB	ngram_range=(1,2) alpha=0.36	88.36%
2	CountVectorizer &MultinomialNB	ngram_range=(1,3) alpha=0.36	88.68%
3	CountVectorizer &MultinomialNB	ngram_range=(1,2) alpha=0.6	88.54%
4	CountVectorizer &MultinomialNB	ngram_range=(1,3) alpha=0.6	88.68%
5	TfidfVectorizer& MultinomialNB	max_df=0.75 min_df=2 ngram_range=(1,2) alpha=0.1	86.58%
6	TfidfVectorizer& MultinomialNB	ngram_range=(1,3) alpha=0.1	86.22%
7	TfidfVectorizer& MultinomialNB	ngram_range=(1,2) alpha=1	86.64%
8	TfidfVectorizer& MultinomialNB	ngram_range=(1,3) alpha=1	86.3%

Table 3. Performance of Multinomial NB with count vectorizer

VII. PERFORMANCE EVALUATION

The machine learning model has to evaluated against certain metrics like Accuracy, Precision, Recall, F-score. Accuracy depicts how much the classifier is right to the max. Precision says of all the positive groups which are predicted; how many were actually positive. Recall also known as sensitivity indicates right classifier awareness. F-score helps to compare two models of low recall and high precision or vice versa. For classification models, a construction matrix is required for the calculation of metrics.

A. *Confusion Matrix*

Confusion Matrix, also known as an error matrix, is a table that is frequently used to define the performance of a classification model (or "classifier") on a collection of test data that knows the exact true values for. It enables the output of an algorithm to be visualized. It is a description of the results of prediction on a classification problem. The number of false and true predictions is articulated and divided by each class with count values. It also concentrates on the type of errors made by the classifier.

It is a table of predicted and actual values.

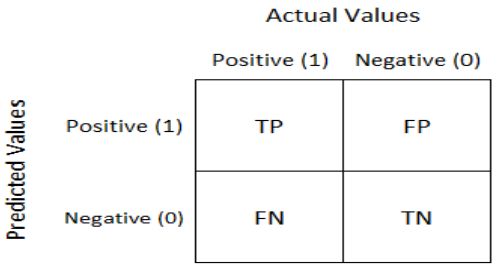

Fig 5. Confusion Matrix

For measuring Recall, Speed, Specificity and Accuracy, Confusion matrix is extremely useful.

True Positive (TP): Positive expected by the classifier, and that is true.

True Negative (TN): Negative expected by the classifier, and that is true.

False Positive (FP): Positive expected by the classifier, and false.

False Negative (FN): Negative expected by the classifier, and it is false.

B. *Accuracy*

How much is the classifier right overall?

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \qquad (13)$$

VIII. CONCLUSION

The work presented in this topic suggests that Multinomial Naive Bayes classifier is better in performing natural language processing tasks. Naive Bayes also has other variants but Multinomial is chosen because it operates better on discrete nature of data for example, number of occurrences if a term in the text. The work done in this paper is also encouraging, as it explains a fairly successful level of machine learning classification with only one extraction feature for huge fake news materials. Ultimately, further study and analysis is due to begin to define and create additional fake news classification grammars which will produce a more detailed classification strategy for both fake news and official statements.

REFERENCES

[1] Classifying Fake News Articles Using Natural Language Processing to Identify In-Article Attribution as a Supervised Learning Estimator.

[2] Tolles, Juliana; Meurer, William J (2016). "Logistic Regression Relating Patient Characteristics to Outcomes".

[3] Quinlan, J. R. (1986). "Induction of decision trees" . Machine Learning.

[4] Breiman, Leo; Friedman, J. H.; Olshen, R. A.; Stone, C. J. (1984). Classification and regression trees.

[5] I. Rish, "An empirical study of the naive bayes classifier," in IJCAI 2001 workshop on empirical methods in artificial intelligence, pp. 41–46, 2001.

[6] How to Encode Text Data for Machine Learning with scikit-learn, Machine Learning Mastery, [Online]. Available: https://machinelearningmastery.com/prepare-text-data-machine-learning-scikit-learn\, Accessed 19th June 2020

[7] What is Text Classification?, Monkey Learn, [Online]. Available: https://monkeylearn.com/what-is-text-classification/, Accessed 19th June 2020

[8] Naïve Bayes Classifier, Towards Data Science, [Online]. Available: https://towardsdatascience.com/naive-bayes-classifier-81d512f50a7c, Accessed 19th June 2020

[9] Logistic Regression for Machine Learning, Machine Learning Mastery, [Online]. Available: https://machinelearningmastery.com/logistic-regression-for-machine-learning/, Accessed 19th June 2020

[10] Logistic Regression — Detailed Overview, Towards Data Science, [Online]. Available: https://towardsdatascience.com/logistic-regression-detailed-overview-46c4da4303bc, Accessed 19th June 2020

[11] Decision Tree Classification, Towards Data Science, [Online]. Available: https://towardsdatascience.com/decision-tree-classification-de64fc4d5aac, Accessed 19th June 2020

[12] K-Fold Cross Validation, Data Driven Investor, [Online]. Available: https://medium.com/datadriveninvestor/k-fold-cross-validation-6b8518070833, Accessed 19th June 2020

[13] Confusion Matrix in Machine Learning, Geeks for Geeks, [Online]. Available: https://www.geeksforgeeks.org/confusion-matrix-machine-learning/, Accessed 19th June 2020

[14] Understanding Confusion Matrix, Towards Data Science, [Online]. Available: https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62, Accessed 19th June, 2020