A Tool for Fake News Detection

Bashar Al Asaad
West University of Timişoara
Timişoara, Romania
bashar.alas3ad@hotmail.com

Mădălina Eraşcu
West University of Timisoara
Institute e-Austria Timişoara
Timişoara, Romania
madalina.erascu@e-uvt.ro

***Abstract*—The word post-truth was considered by Oxford Dictionaries *Word of the Year 2016*. The word is an adjective relating to or denoting circumstances in which objective facts are less influential in shaping public opinion than appeals to emotion and personal belief. This leads to misinformation and problems in society. Hence, it is important to make effort to detect these facts and prevent them from spreading.**

In this paper we propose machine learning techniques, in particular *supervised learning*, for fake news detection. More precisely, we used a dataset of fake and real news to train a machine learning model using `Scikit-learn` library in `Python`. We extracted features from the dataset using text representation models like *Bag-of-Words*, *Term Frequency-Inverse Document Frequency (TF-IDF)* and *Bi-gram frequency*. We tested two classification approaches, namely *probabilistic classification* and *linear classification* on the title and the content, checking if it is *clickbait/nonclickbait*, respectively *fake/real*.

The outcome of our experiments was that the linear classification works the best with the TF-IDF model in the process of content classification. The Bi-gram frequency model gave the lowest accuracy for title classification in comparison with Bag-of-Words and TF-IDF.

***Index Terms*—fake news, Bag-of-Words, TF-IDF, Bi-gram, clickbait**

I. Introduction

Today, in the age of technology, and while we spend most of our time online, we receive hundreds of information from random sources. We are digital citizens so we have the duty of fighting fake news spreading and controlling our life.

In January 2018, the European Commission established a high-level group of experts to advise on policy initiatives to fight fake news and disinformation spread online. The outcome of this group was a report (March 2018) designed "to review best practices in the light of fundamental principles, and suitable responses stemming from such principles". Among the recommendations of the group was to "invest in research and innovation actions to improve technologies for online media services".

In this paper, we present our preliminary experiments on applying machine learning techniques for fake news detection. In particular, we studied and developed methods and tools for detecting fake news, also, proposing a methodology for that purpose and implementing an algorithm which allows reporting, respectively detecting fake news articles.

We used the machine learning library `Scikit-learn` (http://scikit-learn.org) in Python since it has built-in methods that implement different classification approaches. We have used probabilistic (*Naive Bayes*) and linear (*Support Vector Machine*). As text representation models, we used *Bag-of-Words*, *Term Frequency-Inverse Document Frequency (TF-IDF)* and *Bi-gram frequency*. By combining these approaches, we built a fake news detection tool. It has a minimal user interface allowing the user to enter a link to any news article he would like to verify. The entered link is parsed and analyzed. The analysis is made based on article title, date of publication, author name and content.

The experimental results with the tool developed are promising giving an accuracy score greater than 0.8 for content and title classification. The linear classification model works the best with the TF-IDF model in the process of content classification (0.94). The lowest accuracy score was obtained by the probabilistic classifier combined with TF-IDF. Both linear and probabilistic classification gave the same accuracy score (0.95) for title classification, while the Bi-gram frequency model gave the lowest accuracy for title classification in comparison with Bag-of-Words and TF-IDF.

We recommend the users of our tool not to take the results of a fake news detection as a ground truth but to use also the filter of their critical thinking in order to decide the nature of the article.

II. Related Work

The term *fake news* emerged during the last two years. However, fake news and misleading information were present in all time periods. Most of works which discuss the detection of fake news and biased information are relevantly modern. Some of them are based on studying the credibility of a news source regardless of the news content. This process is not a good way because a news source could be classified as untrusted and at the same time it could publish a true fact. An example in this sense is http://www.fakenewsai.com. The authors of this website have used artificial intelligence algorithms (neural networks) to detect if a given website is a credible news source or not. Another project that discusses fake news problem based on news sources and not on the article content is http://bsdetector.tech. This website offers an extension to be installed on Internet browsers and which verifies any website the user will access. Then it gives a notification if the website is classified as unreliable. The websites

This work was supported by a grant of the Romanian National Authority for Scientific Research and Innovation, CNCS/CCCDI - UEFISCDI, project number PN-III-P2-2.1-PED-2016-0550, within PNCDI III.

978-1-7281-0625-0/18/$31.00 ©2018 IEEE
DOI 10.1109/SYNASC.2018.00064

classification was made based on http://www.opensources.co which contains a list of online sources classified using tags like fake, satire, conspiracy, rumor, junk science, hate, clickbait, unreliable, political, reliable. The list was drawn up by analyzing a group of news websites and organizing them under tags (categories). The category of every website was specified by getting background information about the owners of the website, its repetition and its recent behaviour. Unlike these approaches, we will use the content of an article to decide if it is fake or real.

An approach for detecting fake news based on article content is [1], where the authors present an algorithm that uses data mining to detect fake news. The idea is that the verification of a news article depends on many factors, like: 1) the publisher, 2) the content, 3) the time of posting on social media websites, and 4) the number of engagements between different users and the article. First, the algorithm extracts the text features (the characteristics of the content and the publisher). Then it will perform linguistic and visual studies on the extracted features (source, headline, body text, image, video). The linguistic study includes studying the lexical features and the syntactic features; lexical features like different writing systems and sensational headlines; syntactic features such as sentences characteristics and words frequencies. The visual study includes the images and videos specifications analysis. The extracted information will be used to construct a machine learning model to classify the articles as fake or real. The constructed machine learning model will also use external sources to learn from, for example readers feedback. In contrast, in our tool, we used a static dataset where no feedback will be used for training the model. We performed studies only on the text features without taking the visual features into consideration. We studied articles from news websites, and not only social media websites.

Paper [2] discusses the detection of fake news based on clickbait titles. The system studies the relation between the article title and its body. To achieve this, they used a dataset provided by http://www.fakenewschallenge.org. This dataset has headlines and articles verified by the Fake News Challenge Stage 1 (FNC-I) algorithms (https://github.com/FakeNewsChallenge/fnc-1) and classified as unrelated or related. The algorithm verifies the similarity between a headline and content using CoreNLP Lemmatizer (https://stanfordnlp.github.io/CoreNLP/simple.html). The similarity score is calculated based on the n-gram technique, by studying the matches between the headline and the content. They also used the frequency of appearance and the inverse document frequency of a term (https://nlp.stanford.edu/IR-book/html/htmledition/inverse-document-frequency-1.html) to calculate the matching score. When a word is mentioned in the headline and frequently used in the body of the article, then the result will be related. Different to this, we do not check if the title of an article is related to its content.

A similar approach to detect the similarity between headline and article content using the same dataset offered by the fake news challenge website is [3]. The authors present a system of classifiers consisting of two stacked[1] layers. The first layer consists from five independent classifiers (slave classifiers) developed using Natural Language Processing (NLP) modules. The second layer has one master classifier which will use the output of the weak classifiers (slave predictions) as an input. The master classifier will use the predictions of slave classifiers to give a final prediction as an output. Classification techniques used are multi-layer perceptron and ReLU activation function.

It is clear that NLP techniques are the most used methods in fake news detection. These methods help the machine to identify the fake sources and articles. In the project https://github.com/aldengolab/fake-news-detection, the authors presented a NLP-based application which helps detecting misleading sources and headlines; this project applies the NLP algorithms on the input text to make a decision. They present a comparison between the efficacy of models using three different sets of features. The first features set was about applying TF-IDF using Bi-gram frequency; the second features set was studying the syntactical structure normalized frequency (PCFGs); the third one was a union between the first two features sets. They used the classification models from the machine learning library `Scikit-learn` to implement the algorithm. Differently to this work, we also performed title verification using cosine similarity and parsing algorithm, also detecting if the title is clickbait/nonclickbait.

Other strategy to detect fake news and alternative facts like is *crowdsourcing*[2]. In [4], based on social argumentations online, the authors present a prototype system to verify the credibility of a news article. They developed a graph-theoretic framework by using substantial discussion basics. This argumentation graph is filled with information from different Internet users (especially social networks users) through a web-based application.

Another strategy for fake news detection is presented in [5]. The authors show that satirical news could be used as a guide in distinguishing between fake and real news. They present an automated tool which can indicate deceptive information fast and efficiently by analyzing the satire news text characteristics using Support Vector Machines models and 10-fold cross validation to train and evaluate a machine learning model. The main difference to our approach is that they focus on detecting satire articles.

III. Preliminaries

Machine learning (ML) intends to solve the problem of *constructing computer programs that improve by experience*. Applications of ML include virtual personal assistants (Siri, Alexa, Google Now), traffic predictions, online fraud detection, email spam and malware filtering.

There are several types of ML algorithms, depending on how they answer to the questions: 1) *How does the computer*

[1] Stacking in machine learning is an approach to combine classifiers.

[2] Crowdsourcing is a specific sourcing model in which individuals or organizations use contributions from Internet users to obtain needed services or ideas.

know whether it is improving or not?, or 2) *How does it know how to improve?*, namely:

- *Supervised learning*: a training set of examples with the correct responses (targets, class variables or labels) is provided and, based on this training set, the algorithm tries to respond correctly to all possible inputs.
- *Unsupervised learning*: correct responses are not provided, but instead the algorithm tries to identify similarities between the inputs, so inputs that have something in common are categorised together.
- *Reinforcement learning*: the algorithm is told when the answer is wrong, but it is not told how to correct it.
- *Evolutionary learning*: biological evolution is seen as a learning process so organisms adapt to improve their survival rates and chance of having offspring in their environment.

For our problem we have used *supervised learning*. In *supervised learning*, we have a set of data called *the training data* that consists of a set of input data that has target data (labels), which is the answer that the algorithm should produce, attached. The machine learning process consists of the following steps:

1) *Data collection and preparation,* which includes searching for training data, cleaning it and prepare it to the process of features extracting. This step is important to get good results.
2) *Feature selection,* which consists of identifying the features that are most useful for the problem under examination.
3) *Algorithm choice,* given the dataset, after selecting the features, we should choose the suitable algorithm to extract these features from the dataset (classification).
4) *Parameter and model selection,* which means choosing the machine learning model and setting its parameters to guarantee the best performance with the extracted features.
5) Given the dataset, algorithm, and parameters, *training* uses computational resources in order to build a model of the data in order to predict the outputs on new data.
6) Before the model is used, it needs to be *tested* and *evaluated* for accuracy on data that it was not trained on.

A. Classifiers

When we want to solve a problem using machine learning we have more than one approach to find the solution. The used approach differs based on the problem type. When the proposed problem is about determining to which label a given observation belongs to, then we can use *classification methods* to find the solution.

Classification methods are based on classifying the observations in different categories depending on different factors. A dataset with a number of observations is provided for the classifier to help it finding the category of a new given observation. The classifier uses the dataset to train and create patterns to recognize every observation to which category it suits. This method belongs to the supervised learning approach. Any algorithm that performs the classification method and classifies a given feature into a category is called a *classifier*.

In the following, we present classification approaches useful for text classification problems.

a) Naive Bayes Classifier: Naive Bayes classifiers are probabilistic classifiers[3] which apply Bayes theorem in their decision rule assuming strong independence between features. Naive Bayes classifiers are useful for text classification problems, because they can be trained precisely in the supervised learning case.

Given a class variable (label) y and a vector of features $x = (x_1, x_2, ..., x_n)$, where n is the number of features, the probability of y depending on the vector of features x can be calculated using the formula:

$$P(y|x_1, ..., x_n) = \frac{P(y)\prod_{i=1}^{n} P(x_i|y)}{P(x_1, ..., x_n)}$$

Since the value of $P(x_1, ..., x_n)$ is the same for all class variables (labels), we can say that the result of a prediction probability r is the maximum probability of the calculated probabilities for all class variables:

$$r = max(P(y)\prod_{i=1}^{n} P(x_i|y))$$

b) Linear Support Vector Machines Classifier.: Linear classifiers perform classification by using the value of a linear combination between the features of a given observation. In general, linear classification works good for problems with large number of features. Given x the features vector obtained from the input data and w the feature vector obtained from a labelled dataset train data, the output score y will be:

$$y = f(\vec{w}\vec{x}) = f(\sum_j w_j x_j),$$

where f is a function that uses the value $\sum_j w_j x_j$ to obtain the final output score.

In binary classification problems where the classifier has to classify the given observations in two groups (e.g. true or false), the linear classification approach can be seen as a process of splitting a high-dimensional space into two parts by a *hyperplane* (decision boundary). Every part represents a class variable (label). When classifying an observation, the result of f will map the observation to one of the two categories, based on the value obtained from $\sum_j w_j x_j$.

One of the classifiers that uses the linear approach is *linear support vector machine*. It represents the given dataset as points in linear space, separated in categories. For example, assume we have an animal and we want to know if it is a giraffe or an elephant. We are given the features *length of neck* and *head size*, and a dataset of giraffes and elephants

[3] Probabilistic classifier is a classifier that can predict the category of a given observation, based on the observation's probability distribution over the set of categories.

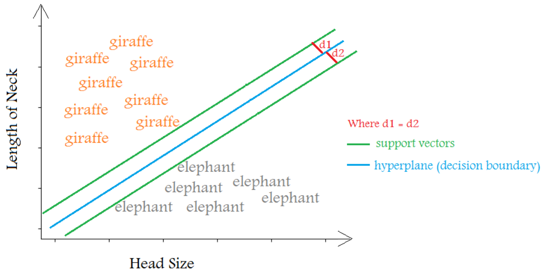

Fig. 1. Two Dimensional Linear Representation

properties. The linear representation of this problem will look like in Fig. 1. In this example we had a classification problem with two features, hence the 2D representation as in Fig. 1. Also, it is a binary classification, so we had a 1D single decision boundary. The number of features will determine the dimension of the representation space and the dimension of the decision boundary. The class variables number will determine how many decision boundary the representation will have. Given *length of neck* and *head size* features of an animal (giraffe or elephant), we will know if the animal is a giraffe or an elephant by representing these feature in the 2D dimensional space we see in Fig. 1. If the given features result a point above the boundary decision then the animal is a giraffe, and if they result a point under the boundary decision then the animal is an elephant. The boundary decision line is specified based on the support vectors. *Support vectors* are made to guarantee the maximum margin between data points from every class. This means the boundary decision should keep an equal distance from the closest data point of each class. In our example, d_1 represents the distance between the boundary decision and the closest giraffe point and d_2 represents the distance between the boundary decision and the closest elephant point.

B. Feature Extraction Models for Text

This type of models are used in natural language processing domain to obtain features from the input text data. The extracted features are mainly used to train a machine learning classifier. Their main purpose is to characterize the input text and make it suitable to be processed by the machine learning classifier.

The input text is represented as a matrix, so-called *term-document matrix*. The number of rows represents the number of all input documents (every document is represented in a row). The number of columns represents the number of all features extracted from all input text documents. The features could be single words or n-grams. Hence, every document is represented as a vector of specific identifiers (count of word frequencies, weight of a word depending on its frequency in a document, n-gram frequencies, etc.).

In the following, we present three text representation models: 1) Bag-of-Words 2) N-gram 3) Term Frequency-Inverse Document Frequency

a) Bag-of-Words model: This model analyzes the text from all input documents and converts it in a Bag-of-Words form. For example, for more than one text (set of text documents), we can have one bag of words which will contain all distinct words from all texts in one bag (well-structured container). It ignores the order of the words and used grammars. It is also known as one-gram model, that is the count of occurrences of a term (single word) in a document. In this case, every element in the matrix will represent the frequency count for a term (column) in a document (row).

b) N-gram model: In this class of models, it is possible to use the either *count of word frequencies* or *weight of a term in a document* to characterize the input text. The main difference is the features form. We can consider Bag-of-Words model as a special case of n-gram models ($n = 1$). When $n = 1$ means that every feature will consist of a single word (token). When $n > 1$ then the model will group n words into a single feature. Then the frequency occurrences or the weight of this feature will be calculated. In n-gram, the models keep the order of the words from the original text. On the other hand, in Bag-of-Words model the features indexes are stored in the matrix randomly.

c) Term Frequency-Inverse Document Frequency model: Term Frequency-Inverse Document Frequency (TF-IDF) uses the frequency count of a term in a document in addition to its frequency count in the whole set of documents for the characterization of the input text. These frequencies are used to calculate a value which represents the importance of a term (word, n-gram) mentioned in a document. In other words, it represents every term (feature) by its weight in a document. This model is typically used because the normal term frequency is not always a satisfiable way to characterize text documents, especially when the set of documents has large number of documents, since we have insignificant terms with high frequency.

In this model, the weight of a term in a document is calculated by finding the frequency of a term in a document then finding the frequency of the same term in the whole set of documents had as an input. This is because the frequency of a term in a document is not enough to determine the importance of a term for a document. For example, the common words like “this” and “a” will have a high number of frequencies. In the normal count frequency model, these words will be considered features with high effect. But in this model it will have ineffective weight because these words will have high frequency count all over the set of documents and not only in the studied document. This will result in a low weight for these words in all documents, so the features corresponding to frequent words will be considered not important in comparison with the features corresponding to less often words.

C. Cosine Similarity Approach

Cosine Similarity is a method for measuring the similarity between two vectors in the vector space. It represents the cosine value of the angle between the two vectors. For example, it is 0 if the vectors are orthogonal, and 1 if the angle between the vectors is 0. The cosine similarity method does not take into consideration the magnitude of the vectors, it depends on their orientation to judge the similarity between them.

Cosine similarity uses the dot product of the two vectors and the product of their magnitude to calculate the similarity score [6].

$$cos(\theta) = \frac{A \cdot B}{||A|| \cdot ||B||} = \frac{\sum_{i=1}^{n} A_i.B_i}{\sqrt{\sum_{i=1}^{n} A_i^2.} \cdot \sqrt{\sum_{i=1}^{n} B_i^2}}$$

The method can be used with any text representation model presented previously. The smaller the value we obtain, the more similar the titles are.

IV. Problem Statement

Lots of factors influence the process of verifying and analyzing news articles, which makes it difficult to achieve a 100% accuracy. So we organised the factors we will be analyzing into: 1) source and the author, 2) title, 3) publication date, 4) content.

These factors combined together form the article full content. We observed that it was not enough to detect the content solely and ignoring the title, author and date. Every factor has its role in determining the credibility of a news article. For example, a news article can have a real content, but an exaggerated title to attract users to click on it. This requires title clickbait detection. Also, reposting old news as new one is common, so the date of publication is important too.

The analysis can be performed with computer science tools. At this aim, we implemented an algorithm (Algorithm 1) that takes as an input a link to a news article and, as output, it displays *details* about the entered article. These details are:

1) for *source and the author*: the news source type will be verified if it is credible or not, then a classification tag will be printed to the user; the author will be extracted and showed to the user; in case the author is not mentioned, the domain name of the website is considered the author.
2) for *title*: if the title is clickbait or nonclickbait.
3) for *publication date*: the most similar real news title that happened in the respective date.
4) for *content*: if the content is real or fake.

V. Our Approach

In this section we present algorithms for fake news detection. They are based on parsing and machine learning techniques. The main algorithm is Algorithm 1 which for every input link to an article displays information about the content, title, date and author by calling other subalgorithms. All these subalgorithms are self-explanatory. We will briefly describe the main ideas behind.

a) Parsing: Web page parsing is the process of extracting information from a web page. Web parsing is based on the HTML source code of the web page. Through parsing, we were able to extract the required information for validation from the given web page, like title, content, date of publication, and author name (if exists).

Algorithm 1 Fake News Detector

```
input: Web link to a news article.
output: (1) Author: name/website. (2) Title: clickbait/nonclickbait. (3) Date: the most similar news title in the respective publication date. (4) Content: fake/real.
Step 1. Verify if the introduced link is trusted or not using http://www.opensources.co lists.
Step 2. If the introduced link is classified as trusted then go to Step 3. Else, print a classification tag (fake, bias, etc.).
Step 3. Parse the HTML source code of the introduced link using Algorithm 2 and extract the following information from the web page: 1) Author of the article. 2) Publication date of the article. 3) Title of the article. 4) Content of the article.
Step 4.
    • Analyze the author using Algorithm 2; if the author name is missing then consider the website which published the article as an author.
    • Analyze the title, respectively content, using Algorithm 3; verify if the extracted title is clickbait/nonclickbait, respectively if the content is fake or real using machine learning and print clickbait/nonclickbait or fake/real accordingly.
    • Analyze the date using Algorithm 4; use the extracted publication date to check the news titles that actually happened in the respective date.
```

Algorithm 2 Web Link Parsing

```
input: Web Link to a news article.
output: (1) The author of the article. (2) The publication date of the article. (3) The title of the article. (4) The content of the article.
Step 1. Open the web link and get the HTML source code.
Step 2. Extract the title of the article.
Step 3. Extract the content of the article.
Step 4. Extract the publication date of the article.
Step 5. Extract the author of the article. If exists, then print the author. Else, print the website domain name.
```

b) Machine Learning: We used machine learning models to verify the article content and title. Using machine learning, we were able to: 1) check if the title is clickbait or not; 2) decide if the article content is fake or real.

In comparison with parsing, machine learning was used to solve classification problems. On the other hand, parsing was used for extracting data.

As it is mentioned in Section III, there are different types of machine learning algorithms; here we used supervised learning. Our algorithm takes a dataset as input; the output of the algorithm will depend on the input dataset. If the input is fake/real news dataset, then the output displays if the article is fake/real. On the other hand, if the input is clickbait/nonclickbait titles dataset, the output displays if the title is clickbait/nonclickbait.

The machine learning process in our application consists of following steps:

1) In the case of content detection, we used the fake and real news dataset from https://github.com/GeorgeMcIntire/fake_real_news_dataset. In the case of title detection, we used the dataset mentioned in the paper [7]. These datasets are clean, labelled and ready to be used for features extraction.
2) We checked if words (tokens) in the articles and titles have a significant impact on whether the content was fake or real, and the title is a clickbait or not.
3) We chose the following text representation models: a) Bag-of-Words model; b) Term Frequency-Inverse Document Frequency model; c) Term frequency Bi-gram model.
4) We implemented these three text representation models with two main classification approaches: linear and probabilistic.
5) We evaluated the classifiers using test data from the imported datasets. The test data were not used in the classifiers training process.

c) Cosine Similarity: We used this approach to verify the similarity between the news title we have and a list of news titles. The list of news titles is extracted using the API https://www.newsapi.org. The list of titles is based on the date of publication we extracted from the news article web page. In other words, we have a list of titles for all events that happened at the same day the input article was published.

To perform the cosine similarity algorithm on the titles, we represented our given title and all the titles in the list as TF-IDF vectors. This means that every title we have has been represented as a vector of weights values for each word (token) in it. Now, we can perform the cosine similarity algorithm between our input title and every other title in the list. The highest similarity score will be recorded and its associated title will be showed to the user.

Algorithm 3 Analyze Article Content/Title

```
input: The article content/title
output: fake/real or clickbait/nonclickbait
Step 1. Read the dataset with fake and real news or with clickbait/nonclickbait titles and split it into train and test sets.
Step 2. Build the text representation model (Bag-of-Words, Term Frequency-Inverse Document Frequency, Bi-gram) from the train and test data.
Step 3. Fit the train data to machine learning classifiers: (1) Naive Bayes (Probabilistic classifier), (2) Linear support vector machine (Linear classifier).
Step 4. Predict the label (fake/real) of the article content or the label (clickbait/nonclickbait) of the article title using the machine learning classifiers.
Step 5. Use the test data from Step 1 to calculate the accuracy score for the machine learning classifiers.
```

Algorithm 4 Analyze Publication Date

```
input: The article publication date.
output: News title with highest score similarity.
Step 1. If the date respects the ISO 8601 date format (yyyy-mm-dd or yyyy-mm-ddThh:mm:ss) then go to Step 3. Else go to Step 2.
Step 2. Transform the extracted date to the ISO 8601 date format.
Step 3. Get the list of news titles that happened in the respective date by sending a web request to https://www.newsapi.org.
Step 4. Build text representation model (Term Frequency-Inverse Document Frequency) from the extracted title and the list of received titles from newsapi.
Step 5. Use the Cosine Similarity approach to find the title which is the most similar to the extracted title.
Step 6. Print the extracted title, the other similar title and the similarity score.
```

VI. Implementation Details

We implemented our application using Python (https://www.python.org) since it supports a large number of efficient packages helping to deal with any type of data (images, text, audio, etc.) and to achieve any target he wants (machine learning, deep learning, web development, etc.). To implement our application we used the following libraries: (1) `Scikit-learn` (http://scikit-learn.org), (2) `Pandas` (https://pandas.pydata.org), (3) `Beautiful Soup 4` (https://www.crummy.com/software/BeautifulSoup/bs4/doc/), (4) `PyQT5` (https://www.riverbankcomputing.com/software/pyqt/intro), (used for implementing the application's graphical user interface); and two external APIs: (1) Google Cloud Natural Language Processing API (https://cloud.google.com/natural-language/), (2) News API (https://newsapi.org).

A. Parsing

In order to obtain the useful information from a given web link, our algorithm uses the Beautiful Soup 4 library for parsing the HTML source code and obtaining the title of the article and the content.

For extracting the publication date we did not use the Beautiful Soup 4 library, because there is no common HTML tag to represent the publication date. Hence we transferred the encoded HTML bytes obtained by the Beautiful Soup 4 object to string.Now, we can search for the date of publication by searching for any text that matches a date format pattern. We created a regular expression to extract all texts that match any possible date format.

For extracting the author name, since there was no specific and common HTML tag used to represent the author, we had to search for the author name in the HTML source code after we have converted it to string. To do that, we used the Google Cloud Natural Language Processing API to extract the entities of type *Person* from the whole HTML source.To get the author

	BoW	TF-IDF	Bi-gram
MN	0.883	0.845	0.903
LSVC	0.883	0.941	0.868

TABLE I
CONTENT DETECTION ACCURACY SCORES

name, we had to search for the word *Author* in the source. The Google API returned all persons names from the given text, and this will be considered the author names. If the HTML source has no mention of the *Author* then the website domain name will be considered the author of the news article.

B. Machine Learning

1) Analyzing Content and Title: The verification of content and title is similar the only difference being the used dataset.

We started by reading the dataset using the `Pandas` library. Our datasets (for the content and title) are stored in two separated files of type csv. We have two data sets:

- fake_or_real_news.csv has a 6335 news articles, 3164 being fake, and 3171 being real. Every news article in the dataset is labelled as fake or real.
- log_32k.csv has 32000 titles, 15999 being clickbait, and 16001 being non-clickbait. Every title in the dataset is labelled as clickbait or nonclickbait.

After we have read the datasets, we used `Scikit-learn` library for all the steps involved in the supervised learning algorithm for fake news detection.

2) Analyzing Date of Publication: For this analysis we used https://newsapi.org API and cosine similarity method from `Scikit-learn`.

VII. EXPERIMENTAL RESULTS

In this section we discuss the obtained results by combining each classifier with every text representation model. To compute the prediction accuracy of a classifier, we used the `metrics` class from `Scikit-learn` library. Then we represented these results as a ROC Curve. The accuracy is calculated based on the test data we had when we split the dataset to train data and test data. The items labels in the test data were removed and stored in another separate variable. We pass the test data to the classifier to see what predictions it results. Then we compare the classifier predictions with the labels we have removed from the test data. The accuracy score is the percentage of the true predictions.

We have the following abbreviations to represent the results of the classifiers in the tables of accuracy and ROC curves: 1) Bag-of-Words: BoW 2) Bi-gram: bigram 3) Term Frequency-Inverse Document Frequency: TF-IDF 4) Multinomial Naive Bayes: MN 5) Linear Support Vector Classifier: LSVC

A. Content Detection Results

We used a dataset with 6335 news articles and performed random split on this dataset into two parts: *training data* and *testing data*. Training data consists of 66.6% of the data and testing data consists of 33.3% of the data.

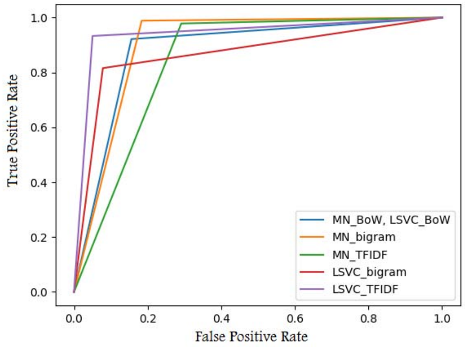

Fig. 2. ROC Curve Representation for Content Detection

	BoW	TF-IDF	Bigram
MN	0.957	0.956	0.849
LSVC	0.947	0.956	0.845

TABLE II
TITLE DETECTION ACCURACY SCORES

From Table I, we observe that the LSVC classifier performed well with the TF-IDF model; at the same time the MN gave the worst result with TF-IDF. This is because the MN usually requires integer feature counts to predict with a higher accuracy. Also, the results show that representing documents through weighted terms vectors (TF-IDF vectors) is more efficient in the case of linear classification. This is because the linear classifiers do not necessary require integer feature counts.

Both classifiers gave the same accuracy score with the BoW model. On the other hand, the MN classifier gave a better score accuracy in case of Bi-gram model. This depends on the Bi-gram term-document matrix, which has pairs of tokens frequencies as features. This means that using the frequencies of pairs to calculate the probabilities gives better results than mapping these frequencies in a linear space; the reason is that having a pair of tokens with a high frequency can indicate a bigger probability for one of the labels than the other.

B. Title Detection Results

From Table II, we observe that the accuracy scores from both classifiers with the models BoW and TF-IDF are the highest. This is because the dataset we used has a large number of titles, hence a large dataset means large number of features. At the same time, titles are short documents which makes it easier to indicate to which class variables they belong to. This is because in short documents, as a title, there is a small number of features. So the frequencies of a word will have more impact on the classification process, in comparison with large documents which have large number of features where the prediction process will be less accurate. On the other hand, accuracy scores obtained using Bi-gram model were the lowest with both classifiers. This is because the Bi-gram feature extraction strategy gives relatively low results in case

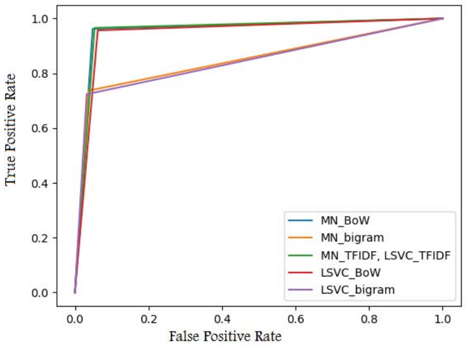

Fig. 3. ROC Curve representation for title detection

of short documents. This is because a large number of Bi-gram frequencies will be 0 or 1, so it will be harder to the classifier to create a clear classification pattern.

The dataset we used to detect if a title is Clickbait/nonClickbait had 32000 news titles. We performed randomly split on this dataset into two parts. Training data consists of 66.6% of the data and testing data consists of 33.3% of the data.

VIII. Conclusion and Future Work

The problems of *fake news* and *disinformation* play an important role on nowadays life. This is because the advanced level of technology and communication methods we have enabled information spreading among people without any verification. This is a reason why researchers started searching for solutions to stop fake news and disinformation from spreading easily. However, it is well known that controlling the flow of information online is impossible.

In this paper, we performed an attempt to verify the news articles credibility depending on their characteristics. At this aim, we implemented an algorithm combining several classification methods with text models. It performed well, and the accuracy results were relatively satisfying.

As future work, we plan to better study the combination between the feature extraction methods and the classifiers as we will be able to choose the text representation model that performs best with the classifier. Moreover, to achieve a higher accuracy, we will have to implement a more sophisticated algorithm which may use data mining technologies with big data, because creating a big dataset including more types of news articles with more class variables (labels) will help raising the accuracy score.

References

[1] K. Shu, A. Sliva, S. Wang, J. Tang, and H. Liu, “Fake news detection on social media: A data mining perspective,” *SIGKDD Explor. Newsl.*, vol. 19, pp. 22–36, Sept. 2017.

[2] P. Bourgonje, J. M. Schneider, and G. Rehm, “From clickbait to fake news detection: An approach based on detecting the stance of headlines to articles,” in *Proceedings of Natural Language Processing meets Journalism* (O. Popescu and C. Strapparava, eds.), Association for Computational Linguistics, 2017.

[3] J. Thorne, M. Chen, G. Myrianthous, J. Pu, X. Wang, and A. Vlachos, “Fake news stance detection using stacked ensemble of classifiers,” in *Proceedings of the 2017 Workshop: Natural Language Processing meets Journalism, NLPmJ@EMNLP, Copenhagen, Denmark, September 7, 2017*, pp. 80–83, 2017.

[4] R. J. Sethi, “Crowdsourcing the verification of fake news and alternative facts,” in *Proceedings of the 28th ACM Conference on Hypertext and Social Media*, HT ’17, (New York, NY, USA), pp. 315–316, ACM, 2017.

[5] V. Rubin, N. Conroy, Y. Chen, and S. Cornwell, “Fake news or truth? using satirical cues to detect potentially misleading news,” in *Proceedings of the Second Workshop on Computational Approaches to Deception Detection*, (San Diego, California), pp. 7–17, Association for Computational Linguistics, June 2016.

[6] A. Singhal, “Modern information retrieval: a brief overview,” *BULLETIN OF THE IEEE COMPUTER SOCIETY TECHNICAL COMMITTEE ON DATA ENGINEERING*, vol. 24, p. 2001, 2001.

[7] A. Chakraborty, B. Paranjape, S. Kakarla, and N. Ganguly, “Stop clickbait: Detecting and preventing clickbaits in online news media,” in *Advances in Social Networks Analysis and Mining (ASONAM), 2016 IEEE/ACM International Conference on*, pp. 9–16, IEEE, 2016.