Fact or Fake
A fake news detection classifier

Aayush Saxena, Harsh Agrawal, Kanchan Bisht, Sai Kiran Mayee Maddi
`asaxena6(200262043), hagrawa2(200257388), kbisht(200203292), smaddi(200257327)`

1 Abstract

With the increased use of social media, we have seen that sharing of information has never been more easier. With that said, sharing incorrect information and making it viral has also become very easy which is causing a lot of problems as this can be used as a powerful influencing tool.

In this project we focus on classifying news articles as Fact or Fake by analyzing the text of the article, its headline and the URL of its source. After some research and exploratory analysis, we compiled a set of features by summarizing the text of the article, finding its sentiment and checking authenticity of the source of the article, which prove to be useful features for the classification process. In addition, we created models using various classification algorithms and tuned them for automatic detection of fake news articles.

2 Background

2.1 Problem Statement

Fake news is a type of yellow journalism or propaganda that consists of deliberate disinformation or hoaxes spread via traditional print and broadcast news media or online social media. Fake news prevailed from 14th century and due to the massive outreach of social media and the online resources, fake news is highly reachable to people. The proliferation of fake news is deliberately done mostly for commercial and political purposes. Some organizations work on creating tools and websites with the content that is similar to the original news and spread hoax information. This appealing misinformation has a great impact on people's opinion and cognizance.

According to the research by Allcott and Gentzkow, social media played a major role during 2016 elections. They stated that, according to the recent evidence 62% of US adults get news from social media and most of it is fake news which is widely shared on Facebook and many of the readers believe it to be true. The fake content is widespread now as online platform has no restrictions on the users for posting content and there are no regulatory authorities to inspect the same, thus, it is difficult to track reliable sources. Considering these instances, there is an increasing need to identify and distinguish between the fact and fake content.

2.2 Literature Survey

In this era of social media, a rapid exchange of information paves the way to research on classifying news as Fake or Fact. This idea is blooming and picking up pace. Different methods are already proposed for solving this issue and many are still under research.

Facebook is trying to analyze and check the accuracy of the posts published in the news feed. They use human intervention to create models and understand spam articles. Based on this model, they are identifying similar spam posts.

"FiB: Stop living a lie", was done as a project in hackathon by students from Princeton, which does a Google search for the text of the post and shows high confidence search results along with an indication of Fact or Fake.

In addition to this, research and projects are being done which apply techniques like querying the web for the text, surveying the audience response, finding the source etc. Others have the notion of finding the linguistic features and setting up a pre-built database to find the truthfulness of the article.

In entirety, we can say that, although researchers are working on identifying features by scanning Twitter, Facebook posts and images, yet it is technically challenging to discover optimally robust algorithm which can solve this problem. We found that it is always not possible to receive the survey response from the audience and it is also very tough to build a database with a set of articles of both the categories(Fact and Fake).

By understanding the above procedures, we decided to build an algorithm which can predict genuineness of the article by analyzing the title and the body. This can be done by trying to capture patterns prevalent in Fact or Fake articles by summarizing the document, finding its sentiment and evaluating the authenticity of the origin of the document.

3 Methods

3.1 Proposed Method

To classify an article as Fact or Fake, our first task was to extract patterns from the text of articles which capture the way a Fact or a Fake article is written.

To convert the text into a feature vector, we utilized Doc2Vec (Section 3.2), which summarizes a document in a fixed length (300) numeric vector. We did the same with the headline of the article and obtained a fixed length feature vector. These features will capture the patterns prevalent in Fake or Fact Articles.

To Augment our feature set, we found the sentiment of the article and the headline by using NLTK's VADER library (Section 3.3) and added the sentiment values to our feature set. This was done as Fake News generally tend to have a lot of negative or positive content.

To expand our feature set further, we compiled a list of Fake and Fact Website URL's and performed a String Matching with our training data URL's (Section 3.4) to generate a score which represents the authenticity of the document origin.

After obtaining the feature-set we applied various classification algorithms (Section 3.5) and fine-tuned them, to classify the documents as Fact or Fake using our feature vector.

3.2 Doc2Vec

- Word2Vec: It is a 2-layered neural network which is used to create word embeddings (which are used to convert words into fixed length vectors). Word2Vec tries to group similar words together by understanding the context in which they had appeared in the past instances. It has 2 variants:
  - Continuous Bag of Words Model: This creates a sliding window around the word and uses the context to predict the target word.
  - Skip Gram Model: This uses a word to predict target context i.e. all surrounding words.

  The neural network tries to correct the weights by backpropagation, whenever the prediction is incorrect. The weights of the neural network after some transformations are used to create the output vectors. It is able to capture relations like King to Queen is same as man to woman.
- Doc2Vec: The Doc2Vec model is a 2-layered neural network which is part of Gensim (an open-source vector space modelling library) which is used to create a numeric representation of a document. Doc2Vec uses word2vec as an underlying model but it also adds an extra input to the neural network, which is a document Id or a unique tag. It also has 2 variants just like Word2Vec:
  - Distributed Bag of Words version of Paragraph Vector (PV-DBOW): This uses the tag to predict the context.[Figure 1]
  - Distributed Memory version of Paragraph Vector (PV-DM): This uses document tag along with context words to predict the target words.[Figure 2]

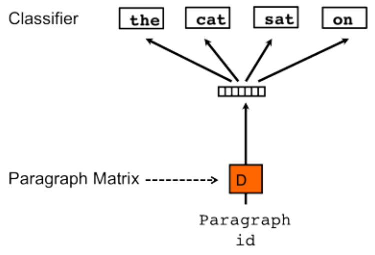


Figure 1: pv-dbow

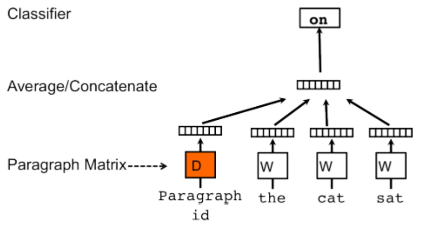


Figure 2: pv-dm

Doc2Vec creates a fixed length vector which represents the essence of the whole document. We are making use of PV-DM model for our project.

Doc2Vec gives the best possible summary of the document. And traditional approaches like BagOf-Words and n-gram cannot be applied to a document.

3.3 Sentiment Analysis

Sentiment Analysis is the process of using Natural Language Processing techniques and Machine Learning models to analyze the emotion or feeling conveyed by the input text. VADER (Valence Aware Dictionary for sEntiment Reasoning) is a model used for text sentiment analysis that is sensitive to both polarity (positive/negative) and intensity (strength) of emotion.

Fake news articles are generally written to bring down someone's reputation or boast about someone to make them popular, so it is natural that they may contain a lot of negative content or a lot of positive content. So sentiment of the headline of the text along with the text of the article can be very useful in determining the authenticity of the article.

Primarily, VADER sentiment analysis relies on a dictionary which maps lexical features to emotion intensities called sentiment scores. The sentiment score of a text can be obtained by summing up the intensity of each word in the text.

Some examples:

- Text: "A really bad, horrible book."
  Sentiment: 'neg': 0.791, 'neu': 0.209, 'pos': 0.0, 'compound': -0.8211
- Text: "VADER is smart, handsome, and funny!"
  Sentiment: 'neg': 0.0, 'neu': 0.248, 'pos': 0.752, 'compound': 0.8439

3.4 URL Matching

In this method, we compare the source URL of the article with a pre-compiled list of fact and fake website URLs. Generally, the fake websites URLs resemble a lot with the URL of some authorized website, to deceive people.

Example legitimate and fake website

Legitimate	***Fake***
cnn.com	CNNews3.com

From the table above, we can see that it is difficult to distinguish between legitimate and fake websites. Through URL matching we generate a similarity score of the training data URL by matching with the list of pre-compiled sources and add it as a feature to our model. This feature improves the performance by a huge amount as a lot of fake news articles come from already identified fake news websites.

3.5 Classification Models

3.5.1 SVM

Support Vector Machines or SVMs are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis.SVM creates a maximum margin hyperplane in a high dimensional space which can be used to separate different classes of data. It uses a fixed number of data points called Support Vectors near the margin of the boundary to create the hyperplane. The equation of the plane is represented as follows:

$$w^T x + w_0 = 0$$

$$w^T x + w_0 >= 0 \quad for\ all\ i,\ such\ that \quad y_i = +1$$

$$w^T x + w_0 >= 0 \quad for\ all\ i,\ such\ that \quad y_i = -1$$

$$Together, \quad y_i(w^T x + w_0) >= 0$$

The points on the margins (support vectors), satisfy the equation:

$$y_i(w^T x + w_0) = 1$$

$$Width\ of\ margin: \quad 2/||w||_{L2}$$

The problem is the maximize this margin as well as satisfy the constraints:

$$y_i(w^T x + w_0) >= 0$$

Instead of solving multiple constraints SVM incorporates the constraints into the optimization by using a Lagrangian function given by the following equation:

$$J(w, w_0, \alpha) = ||w||^2/2 - \sum_{i=1}^{n} \alpha_i \big[y_i(w^T x + w_0) - 1\big], \alpha_i \geq 0$$

Since not all data sets are linearly separable ,the original feature space can be mapped to a high dimension in which the data points are separable, the Kernel trick allows us to do just this with no extra cost.
Radial Basis Kernel/ Gaussian equation:

$$K(x, x^{'}) = exp\big[-1/2||x - x^{'}||^2\big]$$

This is a type of kernel which we can use to map the features to a higher dimensional space.

$$maximize_{\alpha} \sum_i \alpha_i - 1/2 \sum_{i,j} \alpha_i \alpha_j y_i y_j K(x_i, x_j)$$

$$K(x_i, x_j) = \phi(x_i).\phi(x_j)$$

$$\sum_i \alpha_i y_i = 0$$

$$c \geq \alpha_i \geq 0$$

Where ϕ is a feature in high dimensional space, alpha are the Lagrangian coefficients.

We chose SVM because of its above mentioned capabilities and its kernel tricks. Also, because SVM enables classification of linearly unseparable data.

3.5.2 Decision Tree

A decision tree is a decision support tool that uses a tree-like model of decisions and their possible consequences, including chance event outcomes, resource costs, and utility. It is a flowchart-like structure in which each internal node represents a "test" on an attribute (e.g. whether a coin flip comes up heads or tails), each branch represents the outcome of the test, and each leaf node represents a class label (decision taken after computing all attributes). The paths from root to leaf represent classification rules.

We used decision tree because it is useful with categorical attributes, is easy to construct, is able to formulate rules for classification and provides relatively high accuracy.

3.5.3 Naive Bayes

Naive Bayes classifiers are a family of simple "probabilistic classifiers" based on applying Bayes' theorem with strong (naive) independence assumptions between the features.
$P_r(F|W) = P_r(W|F) * P_r(F)/(P_r(W|F) * P_r(F) + Pr(W|T) * P_r(T))$ where,
$P_r(F|W)$ - conditional probability, that a news article is fake given that word W appears in it;
$P_r(W|F)$ - conditional probability of finding word W in fake news articles;
$P_r(F)$ - overall probability that given news article is fake news article;
$P_r(W|T)$ - conditional probability of finding word W in true news articles;
$P_r(T)$ - overall probability that given news article is true news article.

Consider that probabilities $P_r(F|W)$ are known for each word of the news article. Next step is combining the probabilities to get the probability of the fact, that is, whether the given news article is fake or not. The formula for this looks as following:

$$P_1 = P_r(F|W_1)\dots P_r(F|W_n)$$
$$P_2 = (1P_r(F|W_1))\dots(1Pr(F|W_n))$$
$$[P = P_1/(P_1 + P_2)$$

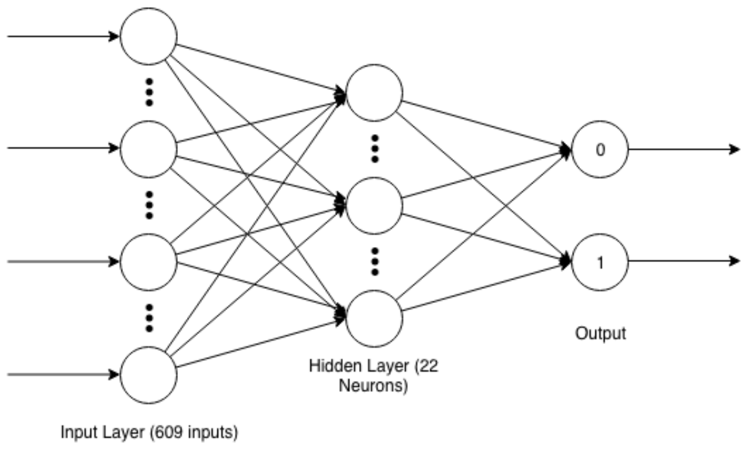
where:
n - total number of words in the news article;
P_1 - product of the probabilities that a news article is fake given that it contains a specific word for all of the words in the news article;
P_2 - same as P_1, but complement probabilities are used instead;
$P_r(F|W_1), P_r(F|W_2)\dots P_r(F|W_n)$ - condition W_1, W_2, W_n respectively appear in it;
P - the overall probability of the fact that given news article is fake.

We have used Naive Bayes since it performs well in case of numerical as well as categorical values and it is particularly useful on large datasets. Despite their naive design and apparently oversimplified assumptions, naive Bayes classifiers have worked quite well in many complex real-world situations.

3.5.4 Deep Learning using Deep Neural Network

A deep neural network (DNN) is an artificial neural network (ANN) with multiple hidden layers between the input and output layers. The model is an assembly of inter-connected nodes and weighted links and the output node sums up each of its input value according to the weights of its links. It then passes this summation through an activation function and reports the output to the next connected node.The neural network adjusts its weights in order to predict the outputs correctly for the given training data.

Keras is an open source neural network library written in Python. It is capable of running on top of TensorFlow, Microsoft Cognitive Toolkit, or Theano. It is designed to enable fast experimentation with deep neural networks and also contains numerous implementations of commonly used neural network building blocks such as layers, objectives, activation functions, optimizers, and a host of tools to make working with image and text data easier.



As our dataset consisted of a large number of attributes (609), the study of relationship amongst them would be pretty complex. Since Neural Networks do not make any prior assumptions, they provide a flexible way to develop a model to fit the data.

4 Experiment

4.1 Data Description

The Data consists of 3988 articles from across the web which are labelled as Fake or Real. A single tuple has URL, Headline, Body and Label. For our experiment we will be using the headline and body of the news article which are textual in nature. Before working with textual data we need to pre-process it, since, it is highly unstructured and needs cleaning.

4.2 Data Pre-processing

We have taken following steps in order to clean our data before performing NLP:

1. Tokenization: It is the process of converting the data into tokens before forming the vectors. An article consisting of paragraphs and sentences, is converted to an array of words. This was done using NLTK's Tokenize library.
2. Lowercase: We converted all the characters to lower case so that the same words can be recognized.
3. Negation Handling: For learning algorithms, it is recommended to use proper structure and follow the rules of context free grammar. All the apostrophes need to be converted into standard lexicons so we can avoid any word disambiguation. Using look-up table of all possible keys we got rid of disambiguates. e.g. "does'nt" becomes "does not".
4. Stop Words: Commonly occuring words do not contribute much to the result as they are present in all the articles and can be removed without affecting our model. Using NLTK's stopwords corpus we removed these from our tokens. e.g. "the","is","at" etc.
5. Punctuations and Empty Strings: Puntuation usually adds noise to the data and dosen't add much value. Punctuations other than (.,?) & empty strings were removed for a better result.
6. Stemming: It is the process of reducing the words to their base form and removing morphological affixes, leaving only the word stem. We use NLTK's PorterStemmer library to stem our words.e.g. "running" -> "run".

4.3 Expermient Setup

In our experiment, we are using dataset acquired from Kaggle. We split the data into two parts using stratified sampling, using 70% of the data to train and validate our model and the rest 30% to test the model. After extracting features form the text, headline and the URL of the articles we formed vectors of size 609 using Doc2Vec library from Gensim,sentiment analysis and URL Matching.Then we use these feature vectors to train various models. Using scikit's GridSearchCV, we have tuned the parameters to increase the accuracy of the model. Some of the open-source libraries used are:

1. NLTK: NLTK is a leading platform for building Python programs to work with human language data. It provides a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning. We have used this library for preprocessing our data. Specifically we have used NLTK's Porter Stemmer for stemming the words, inbuilt stopwords and punctuations set to remove them respectively.
2. Gensim Doc2Vec: Doc2Vec has multiple parameters like:
   (a) vector_size (which denotes the length of the output word vector) - 300
   (b) alpha (initial learning rate) – 0.025
   (c) min_alpha (learning rate will linearly drop to min_alpha as training progresses) - 0.025
   (d) epochs (Number of iterations) - 100
   (e) min_count (ignores all words with frequency less than this) - 5

(f) window (Size of the sliding window) - 10
(g) dm (Defines training Algorithm) - dm=1, 'distributed memory' (PV-DM) is used.

We have build the vocabulary and trained the model using the text of the body of the training data.

3. Scikit: Scikit-learn is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support vector machines, naive bayes, and regression, and is designed to interoperate with the Python numerical and scientific libraries - NumPy and SciPy.

Decision tree and Naive Bayes classifiers are implemented using standard library functions with their default settings.

For the deep neural network classifier, we have used a fully-connected dense layer along with a hidden layer. We varied the number of neurons in the hidden layer to optimize the results. The best accuracy was obtained when the number of neurons in the hidden layer were 22.

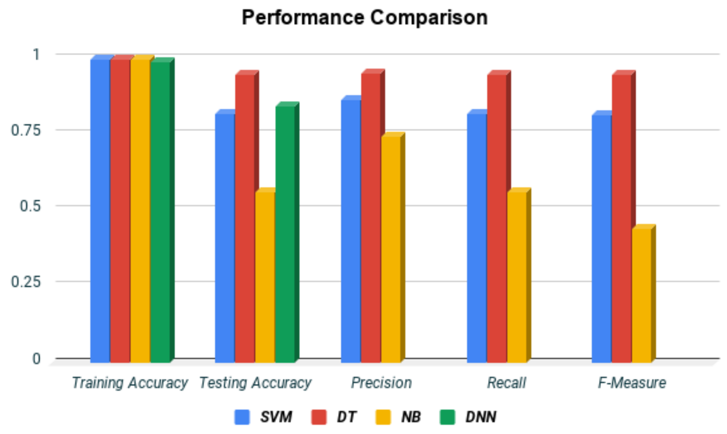
4.4 Results

SVM Parameters: We have used sklearn's Grid Search to train the classifier. The parameters are as follows:

1. C (Cost of misclassifying an instance) – We varied it like C = [0.1, 0.5, 1, 5, 10, 50]
2. Kernel – We tried both the polynomial kernels with degree [-2,3 4] & the RBF kernels with gamma value in [0.1, 0.01, 0.001, 0.0001]
3. Scoring – We chose accuracy as our score metric
4. CV (value of k in k-fold Cross validation) - 5

Deep Neural Network:

1. Input layer: Fully-connected dense (representing all 609 features)
2. Hidden layer: Number of neurons in [10, 12, 15, 22, 25, 27, 30]
3. Output layer: Label - Fake or Real



Detailed Results					
Classifier	Accuracy		Recall	Precision	F-Measure
	Train	Test			
SVM	0.998	0.817	0.817	0.863	0.813
Decision Tree	0.998	0.949	0.949	0.952	0.949
Naive Bayes	0.998	0.563	0.563	0.743	0.439
Deep Learning	0.988	0.842	-	-	-

Note: In Keras as of version 2.0, precision and recall have been removed. We are using version 2.2.4, therefore the corresponding fields are left blank in the above table.

5 Conclusion

After research, analysis and creating models using our data, we found that the Decision tree model performed the best in terms of accuracy and was able to distinguish between Fact or Fake Articles with a testing accuracy of 94%. But this result was because of the feature generated from URL matching, which is highly correlated with our classification label. Since the decision tree would have ignored the other important features, this classifier wont be a very good one if the size of our data set increases and we get fake articles from unknown websites.

So even though Decision tree gave the best accuracy, we feel that the Deep Learning model performed the best. Deep Leaning was able to identify complex patterns through the hidden layer by utilizing all the features and thus performed the best by giving us a testing accuracy of 84%. This model will perform well if we get articles from unknown websites, as it doesn't completely rely on one feature.

We classified our data using SVM and after fine-tuning the hyper parameters using GridSearch, we got a reasonably good testing accuracy of 81%. We found that SVM gave the best results with linear kernel, and it might be due to the fact of the high Correlation of URL Matching Score with the classification label. Although other kernels gave equivalent accuracies and were able to separate the data points in a higher dimensionality using the Kernel trick, yet linear kernel gave the best results.

We tried Naive Bayes classifier as well, which assumes that the attributes are independent, but in the dataset under consideration, the summary vectors might be dependent on each other since the words have a probability of being related to each other. Therefore, Naive Bayes did not perform as well, as the others.

In conclusion, we can say that our classifier works very well using the aforementioned features and is able to correctly identify a Fact or Fake article with a high accuracy using Deep Learning.
Future Scope The current deep neural network model achieves a very good accuracy, but it can be further enhanced by performing a fact check on the text of the document and adding it as a feature to our existing feature set.

References

[1] The current state of fake news: challenges and opportunities

[2] FakeNewsTracker: A Tool for Fake News Collection, Detection, and Visualization

[3] Automatic Detection of Fake News

[4] Sentiment analysis of reviews: Text Pre-processing

[5] Steps for effective text data cleaning (with case study using Python)

[6] Natural Language Toolkit

[7] Wikipedia

[8] CSI: A Hybrid Deep Model for Fake News Detection

[9] Social Media and Fake News in the 2016 Election

[10] VADER Sentiment Analysis

[11] The Role of Decision Trees in Natural Language Processing

[12] Fake news detection using naive Bayes classifier

[13] University of Michigan library research guide for fact and fake sites

[14] Market Watch article on fact and fake news sites

[15] Dataset - https://www.kaggle.com/jruvika/fake-news-detection

Github URL - https://github.com/hrshagrwl/fake-news-classifier