Evaluation of Machine Learning Methods for Fake News Detection

Dimitrios Papakostas, George Stavropoulos, and Dimitrios Katsaros

Abstract In a cyber-connected world, fake information appears to be more enticing or interesting to the audience because of their limited attention spans and the plethora of content choices. Taking this into account, fake news detection/classification is definitely becoming of paramount importance in order to avoid the so-called reality vertigo, preclude misinformation and protect actual reality. This chapter presents a comprehensive performance evaluation of eight machine learning algorithms who perform fake news detection/classification based on regression, support vector machines, neural networks, decision trees and Bayes theorem. In every case, our study reaffirms that performance is governed by the nature of data, nevertheless, it sheds light and draws safe generic conclusions with respect to the dimensionality that each algorithm should have, the kind of training that should be performed beforehand for each one of them, and finally the method for generating vector representations of textual information.

Keywords Fake news · Misinformation · Reality vertigo · Machine learning · Algorithms

1 Introduction

Nowadays online information grows at unprecedented rates, and gradually more and more people consult online media, e.g., the Web, Online Social Networks (OSN) such as Facebook and Twitter, for satisfying their information needs. However, not

D. Papakostas (✉) · G. Stavropoulos · D. Katsaros
Department of Electrical and Computer Engineering, University of Thessaly, Volos, Greece
e-mail: papdimit@e-ce.uth.gr

G. Stavropoulos
e-mail: gstavropoulos@e-ce.uth.gr

D. Katsaros
e-mail: dkatsar@e-ce.uth.gr

M. Lahby et al. (eds.), *Combating Fake News with Computational Intelligence Techniques*, Studies in Computational Intelligence 1001,
https://doi.org/10.1007/978-3-030-90087-8_8

all information/knowledge producers are trustworthy, and the problem of fake news—fabricated stories presented as if they were originating from legitimate sources with an intention to deceive—and their spreading is getting more and more severe. It is speculated that in the current decade, people in developed countries will encounter more fake news than real news. This phenomenon is termed *reality vertigo.*[1]

This problem emerged as a major issue particularly during the 2016 US Presidential election, and it is even believed that fake news affected the final outcome. Unfortunately, this is not an isolated event; a study [1] shows that false medical information gets more views, likes, comments than true medical information. The COVID-19 pandemic is the most recent example of this; nearly 80% of consumers in the United States reported seeing misleading news about the coronavirus outbreak,[2] highlighting the extent of the issue and the reach fake news can achieve. Even worse, fake news not only is (more) popular, but spreads at a faster pace [2] than real news, too.

As a result, countermeasures against fake news began to emerge quickly. There are already fact-checking organizations, such as snopes.com, politifact.com, factcheck.com, truthorfiction.com. Efforts are taking place to deploy fact checking services in browsers; a notable effort which attracted media attention[3] is the development of a Google Chrome extension to combat fake news. Other anti-fake news techniques include adding publisher logos to the information items.

1.1 Motivations and Contributions

The need for detecting fake news—or classifying a news item as fake, true, or suspicious—is of paramount importance if we wish to avoid reality vertigo and protect our society, especially the less educated persons of our society. Even though manual or crowdsourced verification efforts could be a solid solution to the problem, scalability issues, due to the tremendous volume of items to be examined, would soon turn such efforts of limited applicability. Thus, algorithmic techniques are the only viable option for addressing the problem at its full scale.

Machine learning has been shown to be particularly effective in eliminating spam email, which is one type of disinformation. Consequently, algorithms in this category were among the first to be tested for efficacy. On the topic of detecting false news, the following machine learning paradigms have been investigated:

- Regression
  - L1 regularized logistic regression
- Support Vector Machines (SVM)

[1] https://www.nature.com/news/astronomers-explore-uses-for-ai-generated-images-1.21398.

[2] https://www.statista.com/statistics/1105067/coronavirus-fake-news-by-politics-us/.

[3] https://yaledailynews.com/blog/2018/01/22/yale-students-design-chrome-extension-to-combat-fake-news/.

  - C-support vector classification
- Bayesian methods
  - Gaussian naive Bayes
  - Multinomial naive Bayes
- Decision tree-based methods
  - Decision trees
  - Random forests
- Neural networks
  - Multi-layer perceptron (MLP)
  - Convolutional neural networks (CNNs)

The present chapter deals with the problem of detecting fake (or real) news from textual resources, and in particular it focuses on the exhaustive comparison of best performing algorithms from the most significant families of machine learning algorithms, i.e., those mentioned above, since their relative performance is unknown, and so is their generic behavior when tested against diverse datasets.

In that context, this chapter is going to answer these two broad questions, and make the corresponding contributions:

- It contrasts the effectiveness and efficiency of the competitors for several diverse datasets, and various performance measures.
- It contrasts the speed of the competitors for these datasets.

The rest of the chapter is organized as follows: Sect. 2 presents briefly the related work. Section 3 introduces the algorithms that will be evaluated. Section 4 describes the evaluation environment, i.e., competitors, datasets, performance measures, and on, and Sect. 5 presents the actual evaluation of the competing algorithms. Section 6 provides some future research directions, and finally Sect. 7 concludes the chapter.

2 Related Work

Machine learning and data mining algorithms have been considered as a very significant arsenal in the battle against fake news. Several supervised models have been proposed. For instance, a ranking model based on SVM and Pseudo-Relevance Feedback for tweet credibility has been developed in [3]. A credible news classifier based on regression was proposed in [4]. SVM on content-based features was utilized in [5] in order to detect fake, satirical and real news items. A comprehensive survey of data mining algorithms employed for fake news detection is contained in article [6].

A different line of research was taken by [7, 8] where the actual content was analyzed and news items were represented as multi-dimensional tensors. This is in contrast to aforementioned works which are based on feature extraction.

Some works investigated the issue of fake news detection following a credibility diffusion-based approach. These works [9] construct complex networks of heterogeneous entities (persons, tweets, events, message, etc.) and study the paths of fake news propagation in order to find non-credible sources of information, and thus infer fake news.

The authors in [10] investigated the characteristics that are more predictive for identifying social network accounts responsible for spreading fake news in the online environment, both from an automatic and human perspective. They conducted an offline analysis using deep learning techniques, as well as an online analysis involving real users in the classification of reliable/unreliable user profiles. The experimental results revealed the information that best enables machines and humans to detect rogue users effectively.

Interestingly, in [11] the authors proposed a fake news approach that included identifying potential fake news spreaders on social media as a first step toward preventing fake news from being spread among internet users. Thus, they investigated whether it is possible to distinguish credible authors from other authors who have shared fake news in the past. They conducted different learning experiments from a multilingual perspective (English and Spanish) and evaluated different textual features, handcrafted and automatically learned, that are primarily not tied to a specific language. The performance of their system achieved an overall accuracy of 78% and 87% on the English and Spanish corpus, respectively.

There are academic efforts to develop web services that will investigate how disinformation spreads and competes in online social networks. For instance, Hoaxy [1] is such a service for Twitter; it is actually a platform for the study of diffusion of misinformation in Twitter.

Less related areas are those concerning rumor classification, trust discovery, clickbait detection, spammer and bot detection, as well as related online services e.g., Botometer which checks Twitter accounts and assigns them a score based on how likely they are to be a bot. However, there are significant differences among those areas and fake news detection as explained in [6], and thus we do not consider them here. Finally, there are algorithms for detecting fake images [12, 13] and fake videos [14, 15], but these are beyond the scope of this chapter.

The interested reader may consult the following articles [16–22] for complete surveys on articles related to fake news detection.

3 Investigated Algorithms

In this section, we provide some background information which concerns the algorithms that are the focus of this chapter.

3.1 L1 Regularized Logistic Regression

Logistic Regression is basically a linear model accompanied by the sigmoid function which is being applied to the linear model in order to convert the output from any real number into the range of [0, 1]. Using the L1-regularization, we add the term w_1 to the cost function where $\|\cdot\|_1$ denotes the 1-norm and w values are the model's learned weights. So as an optimization problem is trying to minimize the following cost function:

$$\min_{w,c} \|w\|_1 + C \sum_{i=1}^{n} \log\left(\exp\left(-y_i\left(X_i^T w + c\right)\right) + 1\right),$$

with X_i, y_i being the input variables, c is the regularization parameter and C is the inverse of regularization strength.

3.2 C-Support Vector Classification

C-Support Vector Classification is one type of Support Vector Machines (SVM) that can incorporate different basic kernels. Given training vectors $x_i \in R^p$ $i = 1 \ldots, n$ in the two class case and the corresponding class labels decision $y_i \in \{-1, 1\}^n$, C-SVC solves the following problem [23, 24]:

$$\min_{w,b,\zeta} \frac{1}{2} w^T + C \sum_{i=1}^{n} \zeta_i$$

with constraints: $y_i\left(w^T \phi(x_i) + b\right) \geq 1 - \zeta_i$, and $\zeta_i \geq 0$, $i = 1 \ldots, n$ and $\phi()$ being the kernel function.

3.3 Gaussian and Multinomial Naive Bayes

Naive Bayes methods are a set of algorithms based on applying Bayes' theorem with the naive assumption of independence between every pair of features. For a given data point $x = \{x_1. \ldots x_n\}$ of n features and a class variable y, Bayes' theorem states the following relationship:

$$P(y|x_1, x_2, \ldots, x_n) = P(y) \frac{P(x_1, x_2, \ldots, x_n|y)}{P(x_1, x_2, \ldots, x_n)}.$$

Using the naive independence assumption that

$$P(x_i|y, x_1, x_2, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n) = P(x_i|y)$$

and since $P(x_1, x_2, \ldots, x_n)$ is constant given the input, this can be formulated as:

$$P(y|x_1, x_2, \ldots, x_n) \propto P(y) \prod_{i=1}^{n} P(x_i|y).$$

Thus, the most likely class assignment for a data point $x = x_1, x_2, \ldots, x_n$ can be found by assigning the class for which the above value is largest. In mathematical notation, this is defined as:

$$\hat{y} = \arg\max_{y} P(y) \prod_{i=1}^{n} P(x_i|y).$$

Gaussian Naive Bayes The likelihood of the features is assumed to be Gaussian:

$$P(x_i|y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}\right).$$

Multinomial Naive Bayes Multinomial Naive Bayes adapts the naive Bayes algorithm for multinomially distributed data. The distribution is parameterized by vectors $\theta_y = (\theta_{y1} \ldots \theta_{yn})$ for each class y, where n is the number of features and θ_{yi} is the probability $P(x_i|y)$ of feature i appearing in a sample of class y. The parameter θ_y is estimated by relative frequency counting:

$$\hat{\theta}_{yi} = \frac{N_{yi} + \alpha}{N_{y+}an}$$

where $\boldsymbol{N_{yi}} = \sum_{\boldsymbol{x} \in \boldsymbol{T}} \boldsymbol{x_i}$ is the number of times feature i appears in a sample of class y in the training set $\boldsymbol{T}$, and $\boldsymbol{N_y} = \sum_{\boldsymbol{i=1}}^{n} \boldsymbol{N_{yi}}$ is the total count of all features for class y. In our tests we used Laplace smoothing by setting $\alpha = 1$.

3.4 Decision Trees

Despite the various decision tree algorithms, the type of decision tree that we used was first discussed by Breiman [25] and is known as CART (Classification And Regression Trees). The decision tree begins with a root node t derived from whichever variable in the feature space minimizes a measure of the impurity of the two sibling nodes. Let $p(w_j|t)$ be the proportion of patterns x_i allocated to class w_j at node t. Then, the measure of the impurity (in our case we chose Gini) at node t, denoted by

$i(t)$ is computed by:

$$i(t) = \sum_k p(w_j|t)\big(1 - p(w_j|t)\big).$$

Each non-terminal node is then divided into two further nodes, t_L and t_R, such that p_L, p_R are the proportions of entities passed to the new nodes t_L, t_R respectively. The best division is that which maximizes the difference given in the equation below:

$$\Delta_i(s,t) = i(t) - p_L i(t_L) p_R i(t_R).$$

The decision tree grows by means of the successive sub-divisions until a stage is reached in which there is no significant decrease in the measure of impurity when a further additional division s is implemented. When this stage is reached, the node t is not subdivided further, and automatically becomes a terminal node. The class w_j associated with the terminal node t is that which maximizes the conditional probability $p(w_j|t)$.

3.5 Random Forests

The Random forests algorithm belongs to the family of ensemble methods. It was introduced by Breiman [26]. During training, the algorithm creates multiple trees using the CART [25] methodology with each tree trained on a bootstrapped sample of the original training data. In contrast to the original publication, the scikit-learn implementation combines classifiers by averaging their probabilistic prediction, instead of letting each classifier vote for a single class.

3.6 Multi-Layer Perceptron (MLP)

A Multi-Layer Perceptron belongs to the class of feed-forward neural networks, and it includes at least three layers of nodes; an input layer, an output layer, and an arbitrary number of hidden layers. A fully connected MLP with three input neurons, a single hidden layer, and an output layer with two-output neurons can be represented graphically as shown in Figure 1. MLPs can be trained using first-order methods, such as classical backpropagation [27], Stochastic Gradient Descent [28], Adam [29] or second-order methods, such as L-BFGS [30].

A one-hidden-layer MLP is a function $f : R^D \rightarrow R^L$, where D is the size of input vector x, L is the size of the output vector such that:

$$f(\mathrm{x}) = softmax(\boldsymbol{W}_1^T logsig(\boldsymbol{W}_2^T \mathrm{x} + \boldsymbol{b_1}) + \boldsymbol{b_2}$$

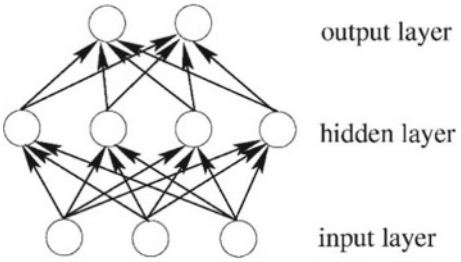

Fig. 1 The topology of a multi-layer perceptron

with $\boldsymbol{b_1}, \boldsymbol{b_2}$ being the bias vectors of the two layers, $\boldsymbol{W_1}, \boldsymbol{W_2}$ being the weight matrices of the two layers, $logsig$ being the logistic sigmoid function, and the $softmax$ function being defined as $softmax(z_i) = \frac{\exp(z_i)}{\sum_{l=1}^{k} \exp(z_l)}$ (where z_i represents the ith element of the input to $softmax$, which corresponds to class i, and K is the number of classes). To train the MLP, in order to learn the set of parameters $\boldsymbol{b_1}, \boldsymbol{b_2}, \boldsymbol{W_1}, \boldsymbol{W_2}$ the L-BFGS quasi-Newton optimization algorithm is used in our experiments.

3.7 Convolutional Neural Networks (CNNs)

A Convolutional Neural Network (CNN) is a type of multi-layer feed-forward neural network with a grid-like topology, which uses the mathematical operation of convolution in place of general matrix multiplication in at least one of its layers. Usually, convolution used in CNNs does not correspond precisely to the convolution employed in other engineering fields and mathematics; almost all CNNs use the so-called pooling operation.

A typical layer in a CNN consists of three stages; in the first stage the layer performs several convolutions in order to produce a set of linear activations; each one of them—in the second or detector stage—passes through nonlinear activation function, and finally, in the third stage a pooling function modifies the output. This pooling function is usually an aggregation (summary) statistic, e.g., max, over the nearby outputs. So, pooling makes the representation invariant to small translations. Training of CNNs can be performed with standard backpropagation methods [32, Ch. 8].

Figure 2 presents a traditional Deep CNN network [31]. The fully connected layer is a standard MPL that uses a softmax activation function in the output layer. The term "fully connected" implies that every neuron in the previous layer is connected to every neuron on the next layer. The output of the feature extraction stage represents the input image's high-level features. The fully connected layer's goal is to use these features to classify the input image into several classes based on the training dataset.

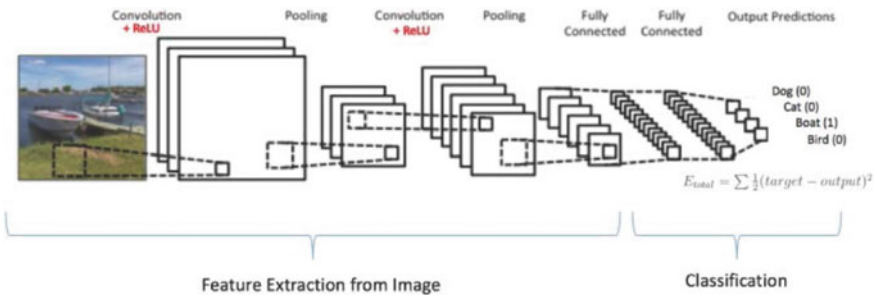

Fig. 2 A typical CNN [31]

4 Evaluation Environment and Settings

This section presents the performance evaluation of the algorithms for fake news classification. We will briefly present the competitors, the datasets, and the performance measures, whereas the actual evaluation will be presented in the next section.

4.1 Competing Algorithms

The competitors are the eight algorithms presented in Sect. 3, namely L1 Regularized Logistic Regression, C-Support Vector Classification, Gaussian Naive Bayes, Multinomial Naive Bayes, Decision Trees, Random Forests, Multi-Layer Perceptron, and Convolutional Neural Networks. The version of the first seven algorithms is that provided by scikit-learn [33], whereas for the last one we developed our own code according to [34].

4.2 Execution Environment

Our tests were executed in two different servers, the first one was used for training the CNNs on a Tesla K20x GPU, and the second one for the rest of the algorithms. This is due to the fact that CNN training is a highly CPU-intensive task. The following Table 1 has the detailed specifications of the machines used in our experiments.

4.3 Datasets

We strived for using freely available datasets that have been used in earlier studies, to ease reproducibility. The datasets are described in Table 2.

Table 1 Servers specifications

	Server 1	Server 2
CPU architecture	Haswell	Ivy Bridge
Model No.	Xeon E5-2695V3	Xeon E5-2620V2
# of cores	14	6
Core frequency (GHz)	2.30	2.10
Main memory (GB)	128	128
GPU	Nvidia Tesla K20x	None

Table 2 Datasets used in the evaluation

	Dataset properties		
Dataset name	Size	Property	Source
"Liar, liar pants on fire": a new benchmark dataset for fake news detection	Training set size of 10,269 articles	Two labels for the truthfulness ratings (real/fake) were used instead of the original six	[36]
The signal media one-million news articles dataset	1 million articles	13,000 articles were selected at random and marked as real news	Signalmedia[4]
Getting real about fake news	13,000 articles	All 13,000 articles were marked as fake news	Kaggle[5]

Before using any of our datasets, firstly we subjected them to some refinements like stop-word, punctuation and non-letters removal and finally we used the Porter2 English Stemmer algorithm for stemming, due to its improvements over the widely used Porter stemmer [35]. This was done in order to avoid noise in our data and make classification faster and more efficient.

Using the datasets from Table 2, we created three input datasets (experiments) on which we evaluated the algorithms. For the first experiment we used the Wang's training dataset [36] which contains various statements from PolitiFact,[6] a Pulitzer Prize-winning Website. From this dataset we used only the headline of each news story and two labels for the truthfulness ratings (real/fake).

Using the two remaining datasets, we created two new datasets which contained a mix of true/fake headlines and a mix of true/fake body texts respectively. For the newly created datasets we chose to keep a balance between the true and fake news using the same number for them from the original datasets. The headlines dataset finally contained 25000 news stories titles that were selected at random from both original datasets and about the body text dataset, using the fact that the average length of stories from five of the top sites that were shared on social media on December

[4] http://research.signalmedia.co/newsir16/signal-dataset.html.

[5] https://www.kaggle.com/mrisdal/fake-news.

[6] http://www.politifact.com/.

2016 was between 200 and 1000 words7, we collected 10000 body texts of a length between 150 and 4000 words. We will call these three datasets as Dataset1, Dataset2 and Dataset3.

4.4 Performance Measures

Since we consider the fake news detection problem as a binary classification task, we evaluated the competitors in terms of the following commonly used measures, namely F1-measure and accuracy whose precise definition are as follows:

- *Accuracy* is the fraction of predictions that are correctly classified as either fake or real news by the model.
- *F1-measure* is the harmonic mean of precision and recall, where precision and recall are defined as follows:
  - *Recall* is the percentage of all fake news that are correctly classified as fake by the model.
  - *Precision* is the percentage of news items being actually fake out of all news items returned as fake by the model.

Moreover, we consider the execution time as another significant quantity to measure; it is comprised by the time to complete two tasks, namely training and classification. So, we measured the following two quantities:

- *Training time*, which indicates the total time (in seconds) needed for training the model.
- *Classification time*, which indicates the total time (in seconds) needed for providing the classification decision.

5 Performance Evaluation

In this section we will present the details of the evaluation setting and illustrate the results.

5.1 Text-To-Vector Transformation

First of all, we needed to transform the text into some numeric or vector representation. This numeric representation should depict significant characteristics of the text. There are many such techniques, for example, occurrence, term-frequency, TF-IDF, word co-occurrence matrix, word2vec and GloVe. In our tests, we used the following two techniques:

- *Word Embeddings.* A word embedding is a parameterized function mapping words of some language to high-dimensional vectors $W : words \rightarrow R^n$. In our tests two different techniques were used:
  - *Pre-trained Word Vectors.* We use the publicly available Glove vectors [37] trained on 6 billion tokens of Wikipedia 2014 + Gigaword 5. The vectors have dimensionality of 50, 100, and 300.[7]
  - *Trained Word Vectors Based on our datasets.* We use word2vec from genism library to train our own vectors based on the selected datasets. The vectors have dimensionality of 50, 100, 300 and were trained using the continuous bag-of-words model. In order to get a single vector representation within each headline/article we averaged the corresponding word vectors.
- *Term Frequency-Inverse Document Frequency (TF-IDF).* TF-IDF weighting scheme is the combination of two terms, the Term Frequency (TF) and Inverse Document Frequency (IDF). Term Frequency, measures how frequently a term t occurs in a document and Inverse Document Frequency, measures the importance of this term t in the whole collection, i.e., its rareness. Even though there are exist many variation of the scheme in literature [38], we use a simple formula; more specifically, we define TF-IDF as follows:

$$tf_{t,d} = \frac{\text{number of times term } t \text{ appears in a document}}{\text{total number of terms in the document}}$$

$$idf_t = \log \frac{\text{total number of documents}}{\text{number of documents with term } t \text{ in it}}$$

So, the final TF-IDF weight of the term t is given by the following product:

$$tf - idf_{t.d} = tf_{t,d} \times idf_t$$

As a result, every document can be interpreted as a vector with one component corresponding to each term in the dictionary together with its weight. For any other term that does not occur in the document, we assign this weight equal to zero.

So, each competitor has seven variants, i.e., three variants due to the three different dimensions of the pre-training, three variants due to the three different dimensions of the training based on our datasets, and one variant based on TF-IDF. So our first step is to discover which of the six former variants is the best one for each competitor.

5.2 How Many Dimensions Are Necessary?

We ask the following two questions:

[7] https://nlp.stanford.edu/projects/glove/

- *How many dimensions are preferable for our algorithms? and*
- *Is it training based on the examined dataset or on benchmark datasets a better solution?*

We present the average accuracy of the six variants of each algorithm in Fig. 3. Deviation is small, so average is quite a good measure for all algorithms with the exception of DT for Dataset1. We observe that CNN is the best performing algorithm, with a significant gap from the next best performing which are Multi-Layer Perceptron and Random Forest. For Dataset1 all algorithms achieve the same performance.

We present the average F1-measure of the six variants of each algorithm in Fig. 4. The obtained results are similar to those observed for average accuracy, with CNN being again the champion method.

We present the average precision of the six variants of each algorithm in Fig. 5. Deviation is small, so average is quite a good measure for all algorithms with the exception of DT for any Dataset and MNB for Dataset2. The relative performance of the methods remains the same as for the accuracy measure.

We present the average recall of the six variants of each algorithm in Fig. 6. Here we see that CNN is again the champion algorithm, but the differentiation of the rest is not very sound.

It is expected that no choice on the number of dimensions and/or training on any kind of data can generate a variant of an algorithm that will be the champion one; such

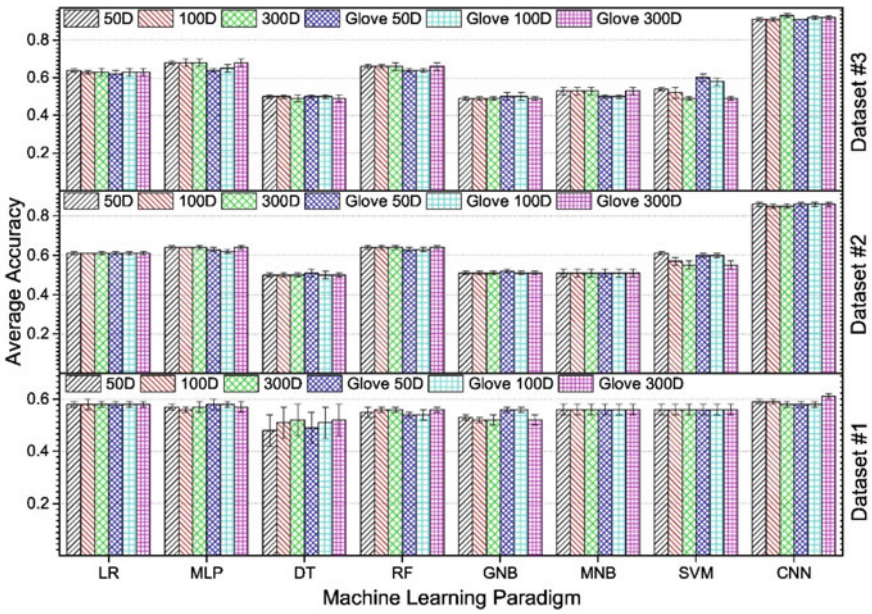

Fig. 3 Average accuracies

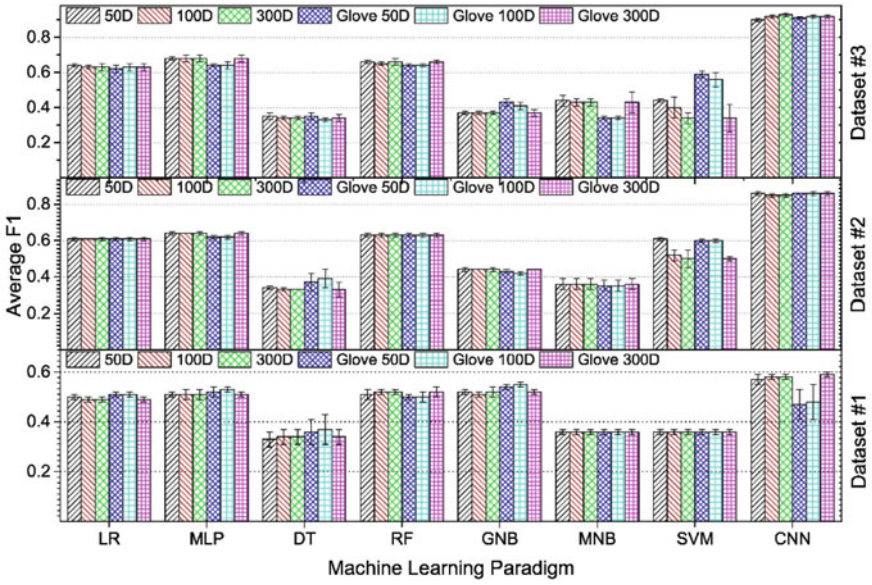

Fig. 4 Average F1-measure

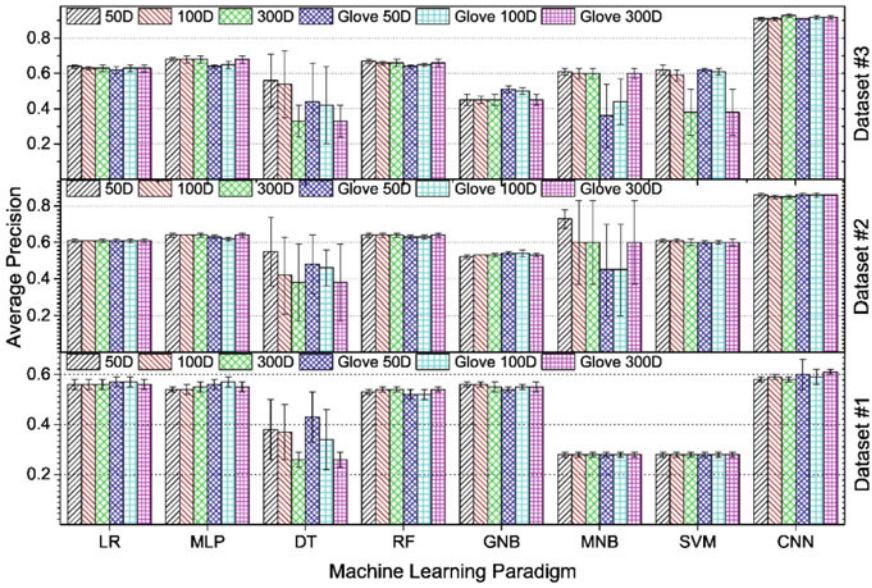

Fig. 5 Average precision

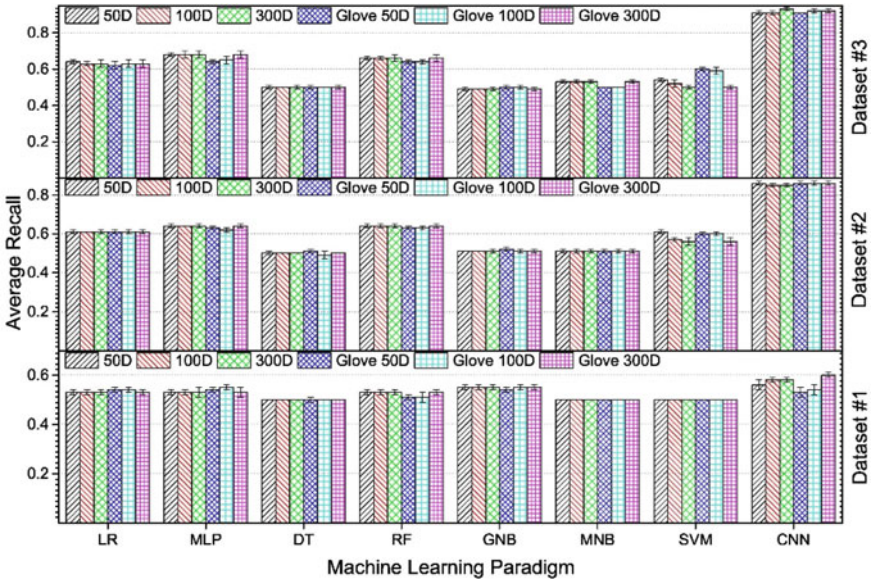

Fig. 6 Average recall

problems and the associated algorithms are highly dependent on data distributions. In Table 3 we present the variant of each algorithm that showed the best performance.

We can draw two quite evident conclusions from Table 3. The first observation is that a small or moderate number of dimensions is preferable because they do not create overfitted models. Secondly, pretraining based on benchmark datasets can be quite effective, meaning that such kind of pretraining is able to create models beating those generated on the specific data that are the target of investigation; this is a quite encouraging result.

Table 3 Champion variant of each algorithm with respect to the number of dimensions and type of training

Algorithm	Dataset1	Dataset2	Dataset3
LR	100D glove	100D	50D
MLP	100D glove	100D	50D
DT	100D glove	100D glove	50D glove
RF	100D	300D glove	50D
GNB	100D Glove	300D glove	50D glove
MNB	Any variant	300D glove	50D
SVM	Any variant	50D	50D glove
CNN	300D glove	100D glove	300D glove

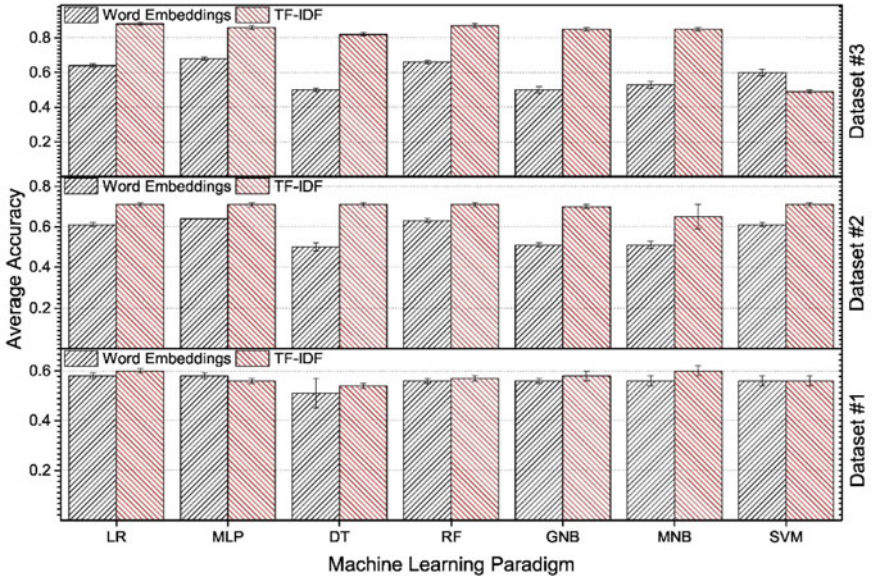

Fig. 7 Average accuracies

5.3 Method of Choice to Generate Vector Representations

Based on the identified "champion" variant of each algorithm from the previous section, we ask the following question:

Is it preferable to use a TF-IDF scheme or word embeddings to generate vector representations of textual information?

The answer to this question is illustrated in Figs. 7 and 8. The first three plots compare the performance of the champion word embedding variant against the TF-IDF variant of each algorithm from the perspective of average accuracy; whereas the other three plots contain the results from the perspective of average F1-measure.

The results show clearly that the TF-IDF representation is a better alternative for the great majority of cases and algorithms. In particular, this representation achieves a 10% better performance in almost all cases, in some cases this gap widens to reach a 30%. The only exception is for SVM in the case of Dataset3.

5.4 Execution Time

As far as the execution time is concerned, Table 4 shows the execution time—training and classification time—of all variants of the algorithms for Dataset1. In general, SVM and the neural network-based algorithms are the most time-consuming during the training phase, which is expected.

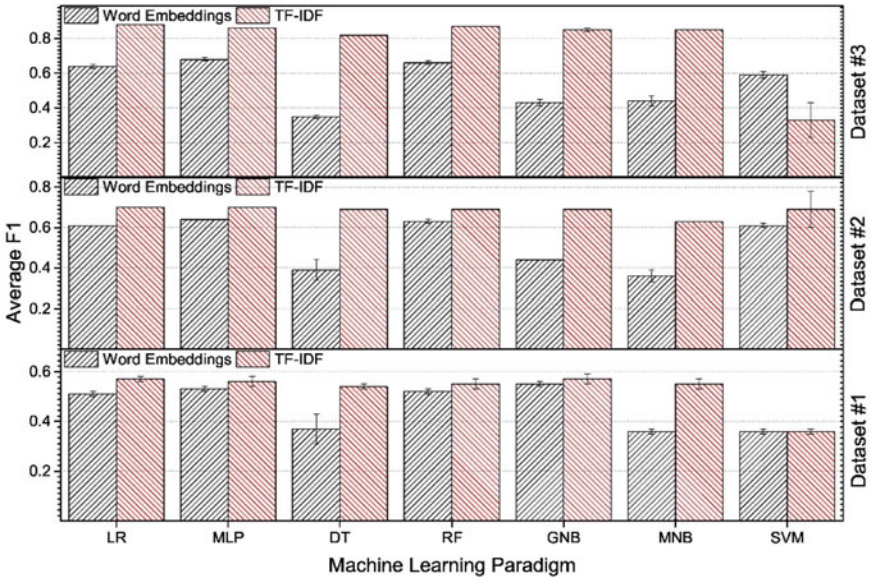

Fig. 8 Average F1-measure

Table 4 Training/classification times (in seconds) for Dataset1

				Glove vectors			
Model	50D	100D	300D	50D	100D	300D	TF/DT
LR	0.69–0.01	0.97–0.01	0.58–0.01	5.75–0.01	7.36–0.00	3.13–0.01	0.04–0.00
MLP	8.37–0.00	7.40–0.00	11.45–0.00	8.12–0.00	6.45–0.00	10.74–0.0	8.46–0.00
DT	1.10–0.00	2.01–0.00	6.39–0.00	1.10–0.00	1.76–0.00	5.44–0.00	0.58–0.00
RF	1.02–0.01	1.39–0.01	2.31–0.01	0.96–0.01	1.33–0.01	2.26–0.01	0.84–0.01
GNB	0.01–0.00	0.01–0.00	0.03–0.01	0.01–0.00	0.01–0.00	0.03–0.01	0.05–0.01
MNB	0.01–0.00	0.01–0.00	0.03–0.00	0.01–0.00	0.01–0.00	0.02–0.00	0.00–0.00
SVM	14.44–1.08	19.08–1.68	54.86–4.81	13.04–1.09	19.09–1.72	53.44–4.78	10.39–0.91
CNN	9.88–0.24	12.28–0.27	16.99–0.27	12.11–0.29	14.72–0.28	17.15–0.29	

5.5 Summary of findings

In summary, our experimentation showed that a small or moderate number of dimensions is adequate, and that pre-trained models based on benchmark datasets can achieve steadily good performance. As far as the method to generate vector representation of textual information is concerned we found out that the TF-IDF method is the clear winner. Finally, among all examined methods and their variants, convolutional neural networks can be considered as the champion algorithm.

6 Future Directions

Although significant developments have been made in recent years with regards to combating the spread of fake news, the lack of standard datasets and benchmarks generates uncertainty, basically due to the lack of authoritative benchmarks within the respective IT community, that precludes more robust achievements. In future research, standard datasets and practical evaluation metrics are needed for comparing various fake news algorithms and promoting the development of more efficient methods.

Another significant problem is the increase in the number of users that spread or share information and the quality of the disseminated data that might be potentially uncertain due to inconsistencies, incompleteness, noise and unstructured nature. This complexity probably jeopardizes the legitimacy of the results of any standard analytic processes and decisions that would be based on them. Designing tailor-made advanced analytical techniques that could conclude on future courses of action with efficacy remains very challenging.

A worthwhile future research point is to investigate the cognitive mechanisms of false information. To elaborate, if we manage to perceive the cognitive mechanisms that fabricated information dissemination is built upon, then more effort can be focused on the respective counter measures/tactics, thus, making them more efficient.

Taking into account that the battle against fake news never ends, counter measures/tactics generality or adaptability is quite important in order to improve their robustness. Fake news identification methods should be able to track unseen, newly coming events, even if the internal system data may differ from contents of emerging events. An insightful research direction to be explored and then adapted accordingly on the fake news detection domain concerns web security, virus/spam detection methods, which also suffer from similar issues such as early detection and model generalization.

Future work is also required in areas of social bots and troll detection, which often act as a catalyst in generating and spreading fake news. The main problem in this particular example is not the fake news rather it is the magnitude of sharing and speed of spreading of the fake news that is causing more harm.

The process of detecting fake information is by nature the learning of a classifier to identify the credibility of some distributed material information. Embracing of novel machine learning models, combining the characteristics of different machine learning models to provide adaptability and improve system efficacy, or even further exploring and extending the potential capabilities of readily available machine learning models signify that there are still more that can be explored.

Providing explainable results should improve fake news detection system efficacy since it is increasing user trust in the detection models. In this research direction the respective experience investigated in other related domains, such as recommender systems, could be very useful.

Moreover, developing counter measures to confront adversarial attacks that target the fake information detection systems is also an area of interest. These adversarial

attacks might impede the robustness of the fake news identification models which means that adding some small perturbations to input vectors could make these models get wrong results.

7 Conclusions

The fast spreading of fake news and the impact they are having on our society, along with the in scalability of manually detecting them, have created a surge of research and development in machine learning algorithms to battle them. In this article, we evaluated representatives from eight well-known families of algorithms, namely regression, support vector classification, multi-layer perceptron, Gaussian and multinomial naive Bayes, random forests, decision trees and convolutional neural networks against three publicly available datasets. We tested the efficiency and training speed of these algorithms. We concluded that a space with a hundred dimensions is of adequate dimensionality to capture the needed text features and get high accuracy of detection. Moreover, we established that the TF-IDF method for generating vectors from the text is a better alternative relative to word embeddings, and finally that pretraining based on benchmark datasets is able to reap performance benefits similar to that when training is performed based on the data under study. As far as the champion algorithm is concerned, we have shown that convolutional neural networks is the best performing algorithm with the downside of requiring significantly higher training time.

Remarks This chapter is an extended version of [39]. We are making this version available in order to have more clear results and discussions in comparison to its short version.

Referesnces

1. Liu, X., Zhang, B., Susarla, A., & Padman, R. Go to YouTube and see me tomorrow: The role of social media in managing chronic conditions. Available at https://ssrn.com/abstract=3061149
2. Vosoughi, S., Roy, D., & Aral, S. (2018). The spread of true and false news online. *Science, 359*(6380), 1146–1151.
3. Gupta, A., & Kumaraguru, P. (2012). Credibility ranking of tweets during high impact events. In *Proceedings of the workshop on privacy and security in online social media.*
4. Hardalov, M., Koychev, I., & Nakov, P. (2016). In search of credible news. In *Proceedings of the artificial intelligence: Methodology, systems and applications* (pp. 172–180).
5. Horne, B. D., & Adali, S. *This just in: Fake news packs a lot in title, uses simpler, repetitive content in text body, more similar to satire than real news.* Technical Report. Available at http://arxiv.org/abs/1703.09398
6. Shu, K., Sliva, A., Wang, S., Tang, J., & Liu, H. (2017). Fake news detection on social media: A data mining perspective. *ACM SIGKDD Explorations, 19*(1), 22–36.

7. Guacho, G. B., Abdali, S., Shah, N., & Papalexakis, E. (2018). *Semi-supervised content-based detection of misinformation via tensor embeddings*. Technical Report. Available at https://arxiv.org/abs/1804.09088
8. Hosseinimotlagh, S., & Papalexakis, E. (2018). Unsupervised content-based identification of fake news articles with tensor decomposition ensembles. In *Proceedings of the workshop on misinformation and misbehavior mining on the web (MIS2).*
9. Gupta, M., Zhao, P., & Han, J. (2012). Evaluating event credibility on Twitter. In *Proceedings of the SIAM international conference on data mining (SDM)* (pp. 153–164).
10. Sansonetti, G., Gasparetti, F., D'aniello, G., & Micarelli, A. (2020). Unreliable users detection in social media: Deep learning techniques for automatic detection. *IEEE Access*, 213154–213167.
11. Vogel, I., & Meghana, M. (2020). Detecting fake news spreaders on Twitter from a multilingual perspective. In *Proceedings of the 2020 IEEE 7th international conference on data science and advanced analytics (DSAA)* (pp. 599–606).
12. Gupta, A., Lamba, H., Kumaraguru, P., & Joshi, A. (2010). Faking sandy: Characterizing and identifying fake images on Twitter during hurricane sandy. In *Proceedings of the ACM international conference on world wide web (WWW)* (pp. 729–736).
13. Galbally, J., Marcel, S., & Fierrez, J. (2014). Image quality assessment for fake biometric detection: Application to iris, fingerprint, and face recognition. *IEEE Transactions on Image Processing*, 710–724.
14. Güera, D., & Delp, E. J. (2018). Deep fake video detection using recurrent neural networks. In *15th IEEE international conference on advanced video and signal based surveillance (AVSS)* (pp. 1–6).
15. Agrawal, R., & Sharma, D. K. (2021). A survey on video-based fake news detection techniques. In *8th International conference on computing for sustainable global development (INDIACom)* (pp. 663–669).
16. Guo, B., Ding, Y., Yao, L., Liang, Y., & Yu, Z. (2020). The future of false information detection on social media: New perspectives and trends. *ACM Computing Surveys.*
17. Zhou, X., & Zafarani, R. (2020). A survey of fake news: Fundamental theories, detection methods, and opportunities. *ACM Computing Surveys,* 1–40.
18. Ghafari, S. M., Beheshti, A., Joshi, A., Paris, C., Mahmood, A., Yakhchi, S., & Orgun, M. A. (2020). A survey on trust prediction in online social networks. *IEEE Access*, 144292–144309.
19. Collins, B., Hoang, D. T., Nguyen, N. T., & Hwang, D. (2021). Trends in combating fake news on social media—A survey. *Journal of Information and Telecommunication*, 247–266.
20. Kumar, S., Kumar, S., Yadav, P., & Bagri, M. (2021). A survey on analysis of fake news detection techniques. In *Proceedings of the 2021 international conference on artificial intelligence and smart systems (ICAIS)* (pp. 894–899).
21. Choudhary, M., Jha, S., Prashant, Saxena, D., & Singh, A. K. (2021). A review of fake news detection methods using machine learning. In *Proceedings of the 2021 2nd international conference for emerging technology (INCET)* (pp. 1–5).
22. Kumar, P. J. S., Devi, P. R., Kumar, S. S., & Benarji, T. (2021). Battling fake news: A survey on mitigation techniques and identification. In *Proceedings of the 2021 5th international conference on trends in electronics and informatics (ICOEI)* (pp. 829–835).
23. Cortes, C., & Vapnik, V. (1995). Support-vector network. *Machine Learning, 20*, 273–297.
24. Vapnik, V. (1998). *Statistical learning theory.* Wiley.
25. Breiman, L., Friedman, J. H., Olshen, R. A., & Stone, C. J. (1984). *Classification and regression trees.* Brooks/Cole Publishing.
26. Breiman, L. (2001). Random forests. *Machine Learning, 45*(1), 5–32.
27. Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning.* The MIT Press.
28. Robbins, H., & Monro, S. (1951). A stochastic approximation method. *The Annals of Mathematical Statistics, 22*(3).
29. Kingma, D. P., & Ba, J. L. (2015). ADAM: A method for stochastic optimization. In *Proceedings of the international conference on learning representations (ICLR).*

30. Nocedal, J. (1980). Updating quasi-Newton matrices with limited storage. *Mathematics of Computation, 35*, 773–782.
31. Selva. CNN classifier using 1D, 2D and 3D feature vectors. MATLAB central file exchange. https://www.mathworks.com/matlabcentral/fileexchange/68882-cnn-classifier-using-1d-2d-and-3d-feature-vectors. Retrieved August 4, 2021.
32. Aggarwal, C. C. (2018). *Neural networks and deep learning: A textbook*. Springer.
33. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine learning in python. *Journal of Machine Learning Research, 12*, 2825–2830.
34. Kim, Y. (2014). Convolutional neural networks for sentence classification. In *Proceedings of the empirical methods in natural language processing (EMNLP).*
35. Porter, M. F. (1980). An algorithm for suffix stripping. *Program, 14*(3), 130–137.
36. Wang, W. Y. (2017). "Liar, liar pants on fire: A new benchmark dataset for fake news detection. In *Proceedings of the annual meeting of the association for computational linguistics* (pp. 422–426).
37. Pennington, J., Socher, R., & Manning, C. D. (2014). Glove: Global vectors for word representation. In *Proceedings of the empirical methods in natural language processing (EMNLP).*
38. Manning, C. D., Ragnavan, P., & Schütze, H. (2008). *Introduction to information retrieval*. Cambridge University Press.
39. Katsaros, D., Stavropoulos, G., & Papakostas, D. (2019). Which machine learning paradigm for fake news detection? In *Proceedings of the 2019 IEEE/WIC/ACM international conference on web intelligence (WI)* (pp. 383–387).