CLFD: A Novel Vectorization Technique and Its Application in Fake News Detection

Michail Mersinias[1], Stergos Afantenos[2], Georgios Chalkiadakis[1]
[1]ECE, Technical University of Crete, Chania, Greece
[2]IRIT, Universite Paul Sabatier, Toulouse, France
mmersinias@isc.tuc.gr, stergos.afantenos@irit.fr, gehalk@intelligence.tuc.gr

Abstract

In recent years, fake news detection has been an emerging research area. In this paper, we put forward a novel statistical approach for the generation of feature vectors to describe a document. Our so-called *class label frequency distance (clfd)*, is shown experimentally to provide an effective way for boosting the performance of machine learning methods. Specifically, our experiments, carried out in the fake news detection domain, verify that efficient traditional machine learning methods that use our vectorization approach, consistently outperform deep learning methods that use word embeddings for small and medium sized datasets, while the results are comparable for large datasets. In addition, we demonstrate that a novel *hybrid* method that utilizes both a *clfd*-boosted logistic regression classifier and a deep learning one, clearly outperforms deep learning methods even in large datasets.

Keywords: machine learning, document classification, fake news detection

1. Introduction

The frequency and impact of fake news has substantially increased in the 21st century, due to the easy access to unfiltered information that is available to the public, mainly anonymously across the Web. It has been shown that fake news spreads faster than credible news (Vosoughi et al., 2018) while at the same time, fake news generation is greater than news fact checking (Rimer-Boston, 2017). According to a March 2018 Eurobarometer survey (Eurobarometer, 2018), fake news constitutes a threat to democracy for 83% of the EU population. Many governments have signed fake news laws which make it a crime to spread fake news online (Ikagai Law, 2018). This means that there is a growing demand for automatically detecting and stopping the spread of fake news, as its quick and effective detection can have a positive social impact. However, the complex nature of fake news makes its detection a challenging task.

There has been a considerable amount of work on fake news detection in the past few years. Existing research falls broadly within the realm of *linguistic analysis*, which uses natural language processing and machine learning methods; or that of *network analysis*, which utilizes knowledge network graphs (Conroy et al., 2015). We follow the first approach and treat fake news detection as a binary text classification task. Considering a fake news detection system, a constant flow of news articles has to be classified as either credible or fake. More generally, a dataset is composed of a number of news articles, or documents, each of which contains a certain number of features. The features always included are the title and the body text of the news article. Other features such as the author may sometimes be included. Therefore, two types of fake news detection systems may be created depending on the assumption we make about the features included in the classification. In *content-based* fake news detection systems, the data provided only contains the title and the body text of each news article; while in *multi-criteria* fake news detection systems, the data provided contains the title and the body text of each news article as well as other characteristics such as the author, the country of origin, the profiles of the readers, the date, and more.

In practice, fake news is often created anonymously and spread by other sources, therefore there is little to no information about the source, the original author or the country of origin. Moreover, making use of profile information contained in a social media platform might raise concerns over the privacy of its users or be unavailable in the first place. Another practical issue is the lack of large quantities of training data, as large datasets are difficult to build and annotate or are proprietary. Furthermore, due to the need of early detection, classification time efficiency is another important factor.

To combat some of the issues identified above, in our work we take a content-based fake news detection approach, so that classification can be performed on textual features that are always available. Our contributions can be summarized as follows. First, we propose *clfd*, a novel approach for the construction of numerical feature vectors to describe a document. We perform a systematic comparison of three *clfd* variants, and demonstrate experimentally that our text vectorization approach can boost the performance of traditional machine learning methods, that are also time-efficient when compared to deep learning methods. Importantly, we show that methods employing *clfd* exhibit classification performance that is superior to that of deep neural networks in small and medium sized datasets (as those more readily available in practice), while being comparable in larger ones. At the same time, our *clfd*-based approach produces results that are consistently better than those reported in recent work concerning those exact same small and medium-sized datasets. Finally, we develop a *hybrid* method which combines a *clfd*-based and a deep learning-based method into an effective classifier that outperforms "pure" deep learning methods even in large datasets.

2. Background and Related work

The goal of a fake news detection system such as ours is to classify news articles into credible or fake by making use of machine learning classification methods. Since most such methods have been designed to learn from numerical data, as a step that precedes classification, it is necessary to acquire a numerical representation of documents which is equivalent to their string representation. This process is known as *text vectorization* (Patterson and Gibson, 2017). Each term in a document is regarded as a feature, and is assigned a numerical representation based on certain attributes such as its frequency of occurrence, or the context around it. The result is a feature vector for each document. Common vectorization techniques (Manning et al., 2010) include *term frequency* (*tf*) and *term frequency - inverse document frequency* (*tf-idf*). *Term frequency* vectorization creates document vectors where each term t is assigned a weight w_t which corresponds to its number of occurrences in the respective document d. A variant of *tf*, called *binary term frequency* (*b-tf*), is a vectorization method where w_t corresponds to the existence (value "1") or absence ("0") of t in d. On the other hand, *tf-idf* vectorization creates document feature vectors where w_t is not only based on the number of occurrences of t in d (tf component), but also on the number of documents n_d in the corpus that include t (idf component). Finally, a well-known vectorization technique for deep learning methods is the use of *word embeddings* which associate each term with a numerical vector that encodes information about its co-occurrence with other terms in the vocabulary.

Related work regarding content-based fake news detection systems has been to date relatively limited. A comparison of several deep learning methods (Bajaj, 2017) has shown that recurrent neural networks and specifically LSTMs, Bidirectional LSTMs (Bi-lstm), and GRUs, which use pre-trained GloVE word embeddings (Pennington et al., 2014) as input, are suitable for identifying fake news as they achieved the highest F1-score (81%). Another content-based fake news detection system found in the literature (Roy et al., 2018), combines the representation results from different methods into a single classification method. More specifically, a multilayer perceptron is used so the representation results of a convolutional (CNN) and a Bi-lstm neural network are combined into an effective classifier for fake news detection. Other content-based systems are based on syntactic and semantic analysis (Rashkin et al., 2017; Badaskar et al., 2008; Pérez-Rosas et al., 2017) and show interesting results as well.

A recent content-based fake news detection system (Bali et al., 2019) utilizes traditional machine learning methods, such as *Gradient Boosting* (Bishop, 2006), in order to achieve a compromise between classification time and high performance. In our fake news detection system, we make use of the two datasets used in (Bali et al., 2019), and demonstrate that we increase the performance of traditional machine learning methods through our novel vectorization approach. In this way, we retain the classification time advantage of traditional machine learning methods, while achieving very high performance: one that is on par or higher than that of deep learning methods, and significantly higher than the aforementioned state-of-the-art for those datasets.

3. Our approach

We propose a novel statistical approach which leads to feature vectors that can be used for the effective numerical representation of documents. While the traditional vectorization methods, such as *tf* or *tf-idf*, treat each term as part of a document, this vectorization technique assigns weights to each term based on the frequency it has within different class labels. We call this approach *class label frequency distance* or *clfd*. In what follows, we introduce our novel *clfd* technique and discuss its benefits; present its specification to fake news detection; and present machine learning algorithms that employ it or are used as baselines.

3.1. Mathematical Framework

We first provide a brief explanation of *clfd* before providing the details. Consider a corpus of documents D. Our approach works by determining the relative frequency of a term in a specific set of documents $D_i \in D$ which belong to a class label i within a set C of possible class labels, compared to the frequency of that term in the set of documents $\{D - D_i\}$ which belong to all other class labels. Intuitively, this calculation determines how relevant a given term is to a particular class label, relatively to its relevance to the rest of the class labels. We call this *class label frequency ratio* (*clfr*), and it is the main component of *clfd*. After calculating the *clfr* weight of a term t for each class label i, we calculate the maximum distance between those *clfr* weights by subtracting the smallest *clfr* value from the largest one. The result is the *clfd* weight for t, which signifies how likely it is that t belongs to a specific class label.

The formal procedure for determining *clfd* is the following. Given a corpus of documents D, a term t, and a set of documents $D_i \in D$ for each class label i, we first calculate three *class label term frequency* (*cltf*) vectors as follows:

Algorithm 1: CLTF vectors calculation

```
Data: corpus, class_labels
Result: Generation of terms, occ, and total vectors
for document d in corpus do
    for class_label i in class_labels do
        if d belongs to i then
            for term t in d do
                terms(t,i) = t;
                occ(t,i) += tf(t,d);
                total(i) += tf(t,d);
            end
        end
    end
end
```

In the above algorithm, $tf(t, d)$ represents the number of occurrences of a term t in a document d. For each class label i, the vector *terms* contains the vocabulary and the vector *occ* the respective number of occurrences of each term t contained in *terms*. Finally, for each class label i, *total(i)* represents the total number of occurrences of all terms contained in the corresponding set of documents D_i.

Now that we have the three *cltf* vectors for each set of documents D_i that belong to class label i, we can calculate a *clfr* weight vector for each class label i, as follows:

$$\textit{clfr}^i(t) = \log_e\left(\frac{(1 + \textit{occ}(t, i)) \cdot \textit{total}(\hat{i})}{(1 + \textit{occ}(t, \hat{i})) \cdot \textit{total}(i)} + 1\right), \forall t \quad (1)$$

In Equation 1, *occ(t,i)* represents the number of occurrences of term t in D_i, while *occ(t,$\hat{i}$)* represents the number of occurrences of term t in $\{D - D_i\}$. Furthermore, *total(i)* contains the total number of occurrences of all terms which appear in D_i, while *total($\hat{i}$)* contains the total number of occurrences of all terms which appear in $\{D - D_i\}$. The $\textit{clfr}^i$ vectors are used to calculate the *clfd* weights as follows:

$$clfd(t) = max_{i \in C}(\textit{clfr}^i(t)) - min_{i \in C}(\textit{clfr}^i(t)), \forall t \quad (2)$$

In Equation 2, the *clfd* weight for a term t is the maximum difference among its *clfr* values. The algorithm for the generation of the (corpus-wide) *clfd* weight vector is provided immediately below:

Algorithm 2: CLFD calculation

```
Data: corpus, vocabulary, class_labels
Result: Generation of clfd vector
terms, occ, total = cltf(corpus, class_labels);
for term t in vocabulary do
    for class_label i in class_labels do
        $\textit{clfr}^i(t)$ computed as in Eq.1;
    end
    clfd(t) computed as in Eq.2;
end
```

In the above algorithm, we first calculate the *cltf* vectors *occ* and *total*, as in Algorithm 1, which represent, for each class label i, the number of occurrences of a term t in D_i and the total number of occurrences of all terms which appear in D_i respectively. The $clfr^i$ vectors contain the *clfr* weights of each term t for every class label, calculated according to Equation 1. Furthermore, *clfd* represents the maximum distance between those *clfr* weights, and is calculated according to Equation 2. Then, the final *clfd* vector for a document d is the result of taking the Hadamard product between the computed *clfd* vector and one generated by a term frequency-based vectorizer such as *b-tf*, *tf* or *tf-idf*:

$$\textit{clfd}^d(t) \leftarrow \textit{clfd}(t) \circ \textit{tf-based\_vectorizer}(t, d), \forall t \quad (3)$$

This step is required in order to remove from a document's *clfd* vector the values that correspond to terms not appearing in that specific document, as a tf-based vectorizer calculates document-related numerical statistics, and would have assigned 0 values to such terms. All tf-based vectorizers create vectors of size equal to vocabulary size v, and Alg. 2 computes a *clfd* vector of size v, thus a simple element-wise multiplication is required in order to generate the final feature vectors for each document.

3.2. Method Discussion

The *clfd* technique provides a sophisticated weighting scheme with certain advantages compared to existing vectorization methods. This is because treating a term t as part of a set of documents D_i which belong to a class label i provides better insights for classification than simply treating t as part of a single document or the entire corpus.

The *clfd* weights represent the importance or relevance of every term for classification. If the *clfd* value of a term t is high, then that term is very likely to be related to documents associated with a specific class label. If the *clfd* value of a term t is low, then that term is not likely to occur in any documents associated with a specific class label. In case the *clfd* value is zero, then the term t is equally likely to occur in documents associated with any class label.

Compared to *term frequency* (*tf*) vectorization, *clfd* contains a natural filter to stopword noise due to the *class label frequency ratio* (*clfr*) component. Therefore, the performance of *clfd* is practically not affected by the quality or even the absence of data preprocessing: a term which has an equal number of occurrences between all class labels will have a *clfd* weight value of zero and it is practically removed. If a term has a significantly higher number of occurrences in documents which belong to a specific class label, and thus a high *clfd* weight, then it should not necessarily be considered a stopword for this domain in the first place, regardless of its frequency of occurrence in the English language.

Consider now *tf-idf* vectorization: in *tf-idf*, terms that are very common in certain domains, such as the term "like" in a sentiment analysis task, will have a low weight value due to the *idf* component. However, in the *clfd* approach, the term "like" will have a very high *clfr* value for the positive class label and a very low *clfr* value for the negative class label, thereby making the final *clfd* weight and relevance of the term "like", high.

Therefore, we can conclude that our vectorization approach provides important advantages over traditional vectorization methods. As such, it is conceivable that it can be used to boost the performance of machine learning algorithms, a fact verified experimentally in the fake news detection domain, as we detail later in this paper.

3.3. Application to fake news detection

While our *clfd* approach can find application in several practical machine learning or natural language processing tasks, we use it here for fake news detection. As this is a binary classification problem, there are only two possible class labels: credible news (c) and fake news (f). Therefore, for the two classes i={c, f}, Equation 1 becomes:

$$\textit{clfr}^c(t) = \log_e\left(\frac{(1 + \textit{occ}(t, c)) \cdot \textit{total}(f)}{(1 + \textit{occ}(t, f)) \cdot \textit{total}(c)} + 1\right), \forall t \quad (4)$$

$$\textit{clfr}^f(t) = \log_e\left(\frac{(1 + \textit{occ}(t, f)) \cdot \textit{total}(c)}{(1 + \textit{occ}(t, c)) \cdot \textit{total}(f)} + 1\right), \forall t \quad (5)$$

Furthermore, the resulting *clfr* vectors will now correspond to class labels c and f only. Therefore, Equation 2 can be specified for these two classes i={c, f} by simply calculating the absolute value of the subtraction between the two aforementioned *clfr* weights as follows:

$$clfd(t) = |clfr^c(t) - clfr^f(t)|, \forall t \tag{6}$$

Now that we defined the equations for the generation of *clfd* weights for our binary text classification task, we provide an example to describe the process in detail. Consider four documents, the first and the third belonging in the class of credible news (c) and the second and the fourth belonging in the class of fake news (f).

	Documents
1	Congress approved the controversial bill
2	The congress is filled with lies, lies and deceit
3	The movie received plenty of controversial reviews
4	This movie was made by aliens to brainwash us

Table 1: Corpus

The first step is to generate the *cltf* vectors for each class label $i = \{c, f\}$ as described in Algorithm 1. Data preprocessing such as stopword removal and lemmatizing has been applied (Manning and Schütze, 1999). In this example, the generated *cltf* vectors, *terms*, *occ* and *total*, are as shown below:

	Terms (c)	Occ (c)	Terms (f)	Occ (f)
1	congress	1	congress	1
2	approve	1	fill	1
3	controversial	2	lies	2
4	bill	1	deceit	1
5	movie	1	movie	1
6	receive	1	make	1
7	plenty	1	alien	1
8	review	1	brainwash	1
Total	-	9	-	9

Table 2: Class label term frequency (cltf) vectors

The next step is to calculate the *clfr* weights according to Equations 4 and 5. The resulting *clfr* weight vectors are:

	Vocabulary	Credible (c)	Fake (f)
1	congress	0.69	0.69
2	approve	1.10	0.41
3	controversial	1.39	0.29
4	bill	1.10	0.41
5	movie	0.69	0.69
6	receive	1.10	0.41
7	plenty	1.10	0.41
8	review	1.10	0.41
9	fill	0.41	1.10
10	lies	0.29	1.39
11	deceit	0.41	1.10
12	make	0.41	1.10
13	alien	0.41	1.10
14	brainwash	0.41	1.10

Table 3: Class label frequency ratio (clfr) vectors

Finally, we calculate the *clfd* weights according to Equation 6. The resulting *clfd* weight vector is as follows:

	Vocabulary	clfd weight
1	congress	0.00
2	approve	0.69
3	controversial	1.10
4	bill	0.69
5	movie	0.00
6	receive	0.69
7	plenty	0.69
8	review	0.69
9	fill	0.69
10	lies	1.10
11	deceit	0.69
12	make	0.69
13	alien	0.69
14	brainwash	0.69

Table 4: Class label frequency distance (clfd) vector

Clfd Variants The final step is to multiply the *clfd* weight vector calculated above, with a vector generated by a vectorizer of our choice as in Equation 3, in order to apply the *clfd* weighting scheme to each document. Our choices in this work were *(i)* a *b-tf* vectorizer, *(ii)* a *tf* vectorizer, and *(iii)* a *tf-idf* vectorizer. This gives rise to three respective *clfd* variants: *b-clfd*, *tf-clfd*, and *tfidf-clfd*.

Analysis In the clfd weight vector of Table 4, we observe that our clfd vectorization approach generates a vector of feature importance that is distinct to those that *b-tf*, *tf* or *tf-idf* vectorizers had produced. Specifically, we present the following example for the corpus of Table 1. Consider document 3:

	Vocabulary	tf weight	tf-clfd weight
1	movie	1.00	0.00
2	receive	1.00	0.69
3	plenty	1.00	0.69
4	controversial	1.00	1.10
5	review	1.00	0.69

Table 5: Clfd example - Comparison of tf and tf-clfd

In Table 5, we notice that a *tf* vectorizer considers all terms to have an equal importance in document 3. For instance, the term “movie” and the term “controversial” are considered equally important as they both have a weight of 1.00. However, a *tf-clfd* approach considers the term “controversial” very important (weight 1.10), the rest of the terms less important (weight 0.69) and the term “movie” non-existent (weight 0.00).

Therefore, we notice that the term “controversial” has been assigned by *clfd* a high weight as it appears in one more “credible” document (document 1) in the collection. Indeed, “controversial” is a rather elaborate and polarizing word, more likely to appear in credible news articles. On the other hand, a neutral term such as the term “movie” has a zero *clfd* weight. As such, *clfd* weights could potentially be used as hints in order to identify the set of terms which have led us to classify a document as credible or fake.

Thus, *clfd* can arguably be viewed as a tool that could help provide explainability of machine learning outcomes, a very important concern in the further analysis of news articles. More specifically, its feature importance vector (*clfd* weights) can help provide useful information which can be used as elements of a system in an explainable AI approach that will provide the reasons behind the classification decision. This is in contrast to the use of deep neural network approaches for classification tasks, where explainability is a big challenge.

3.4. Machine learning methods

Since we treat fake news detection as a binary text classification task, we can either use a linear model, or a non-linear model with a deep learning architecture. In our work, we use both linear and non-linear approaches.

Specifically, we incorporated *clfd* into four traditional machine learning methods, two probabilistic (*Logistic Regression*, *Naive Bayes*) and two ensemble (*Random Forest*, *Gradient Boosting*) ones (Bishop, 2006), and evaluated them against each other and against four deep learning methods. The latter used word embeddings as input, with pre-padding of zeroes and no limit regarding the input length or the vocabulary size, in order to achieve maximum performance. Each deep architecture begins with an embedding layer followed by a dropout layer to reduce overfitting (Srivastava et al., 2014). Then, they differentiate as follows:

1. *Long short-term memory neural network (lstm):* This architecture continues with an lstm layer and its output is passed into a dense layer with a rectified linear unit (ReLU) activation function. The classification is done in a final dense layer with a sigmoid activation function.

2. *Bidirectional long short-term memory neural network (Bi-lstm):* The architecture continues with a Bi-lstm layer and its output is passed into a dense layer with a rectified linear unit (ReLU) activation function. Finally, the classification is performed in a dense layer with a sigmoid activation function.

3. *Combined convolutional and long short-term memory neural network (cnn+lstm):* This architecture continues with three repetitions of the combination of an 1-dimensional convolutional layer and an 1-dimensional max pooling layer. The resulting output is passed into an lstm layer followed by a dense layer with a rectified linear unit (ReLU) activation function. The classification is done in a final dense layer with a sigmoid activation function.

4. *Multiple long short-term memory layers (mult.lstm):* This architecture continues with a first lstm layer and its output is passed, through a repeat vector layer, into a second lstm layer. The output of the second lstm layer is passed into a dense layer with a rectified linear unit (ReLU) activation function. Finally, the classification is performed in a dense layer with a sigmoid activation function.

A novel clfd-based Hybrid method Finally, prompted by the observed performance of the methods above during our experimental evaluation, we developed an algorithm which combines the representation results of two classifiers: *(i)* the traditional machine learning method of *Logistic Regression* utilizing *b-clfd* feature vectors (*l*), and *(ii)* the deep learning method of *cnn+lstm* (*c*). As mentioned above, a dropout layer (0.2) is utilized in order to reduce overfitting. The parameters of the convolutional layers were as follows: filters=128, kernel_size=3, activation='relu'. The subsequent dense layer has 128 units and a ReLU activation function, while the final dense layer has a single unit and a sigmoid activation function. Finally, the *Adam* optimization algorithm (Kingma and Ba, 2014) was used for the training of the neural network.

We first calculate the classification probabilities of these two machine learning methods for each class label i. Our novel *hybrid* method combines these probability vectors by calculating the *weighted average probability* h_i of each news article d to belong to class label i, as follows:

$$h_i = \alpha \cdot l_i + (1 - \alpha) \cdot c_i \tag{7}$$

where l_i and c_i denote the probability assigned to d belonging to class label i by the *l* (i.e., the *b-clfd*) and *c* (i.e., the *cnn+lstm*) methods respectively; and where α represents the weight for l_i. In our binary text classification task of fake news detection, we only have two h values for each d document, h_c and h_f, which represent the probability of d belonging to the class label of credible or fake news respectively. Since $h_c + h_f = 1$, we classify d into credible news if $h_c > 0.5$ or into fake news if $h_f > 0.5$. Notice that the hybrid method's application to non-binary classification problems is entirely straightforward.

4. Experimental Evaluation

In this section, we experimentally evaluate both our vectorization and machine learning methods, on three datasets which differ in size, class balance and data homogeneity. Each dataset was split into training (75%) and test (25%) sets. This setting was used in all models which incorporated a neural architecture; in these cases overfitting was avoided using dropout. For linear models the effects of potential overfitting were reduced using 5-fold cross validation. For the statistical significance of our results, we calculated confidence intervals ($p < 0.05$).

For the experiments, we have built a fake news detection system in the Python programming language, utilizing libraries such as NumPy, Pandas, RegEx, NLTK, scikit-learn and Keras. The experiments were conducted on Google Colab, a free online Jupyter Notebook environment. It provides 12.72 GB of RAM, 48.97 GB of disk space, and, importantly, a Tesla K80 GPU which increases computational effectiveness. The metrics used for evaluation were accuracy, precision, recall and F-1 score (Sokolova and Lapalme, 2009).

4.1. Evaluation Corpora

Label	Dataset 1	Dataset 2	Dataset 3
Fake	3164	10369	24396
Credible	3171	10349	13614
Total	6335	20718	38010

Table 5: Structure of the datasets

Our first dataset (Dataset 1) was George Mcintyre's dataset (McIntire, 2017) which contains 6335 news articles, evenly distributed between the two classes. The main attribute of this dataset is its homogeneity, as the 2016 US election news is the common topic of all articles.

Our second dataset (Dataset 2) was a Kaggle dataset (Kaggle, 2017) which contains 20718 news articles, evenly distributed between the two classes. This open sourced dataset has been reviewed by the Kaggle community and contains credible and fake news articles taken from the Web. Compared to the first dataset, apart from the difference in size, this dataset provides news articles about various topics which will show how our algorithms and methods work in non-homogeneous data.

Our third dataset (Dataset 3) contains 38010 news articles, and is the union of the previous two datasets and another Kaggle dataset of 13000 fake news articles (Risdal, 2016). Adding such a number of fake news articles had the effect of making this dataset large and imbalanced, as it now contains a greater quantity of fake than credible news. Dataset 3 is also even more heterogeneous, as it is composed of three datasets which not only have news articles about various topics, but are also taken from various different sources.

4.2. Evaluation of vectorization techniques

Dataset 1

	Logistic Regression	Random Forest	Naive Bayes	Gradient Boosting
tf	92.73	88.64	89.02	91.37
tf-idf	92.13	89.37	82.15	**91.74**
b-clfd	**94.61**	88.59	**90.63**	91.18
tf-clfd	94.16	88.8	90.27	91.37
tfidf-clfd	92.69	**89.5**	90.38	**91.74**

Dataset 2

	Logistic Regression	Random Forest	Naive Bayes	Gradient Boosting
tf	95.51	92.29	88.82	**94.46**
tf-idf	95.08	92.78	81.78	94.41
b-clfd	**97.41**	91.9	90.58	94.23
tf-clfd	96.79	92.45	86.75	**94.46**
tfidf-clfd	95.66	**92.81**	**90.8**	94.41

Dataset 3

	Logistic Regression	Random Forest	Naive Bayes	Gradient Boosting
tf	96.59	95.18	94.34	92.36
tf-idf	95.07	95.06	**95.82**	**92.42**
b-clfd	**97.17**	95.2	94.44	92.17
tf-clfd	96.89	**95.21**	94.2	92.35
tfidf-clfd	94.87	95.1	95.65	**92.42**

Table 6: F-1 score of vectorization methods

Regarding the comparison of vectorization techniques for traditional machine learning methods, we observe certain patterns in the results. We notice in Table 6 that the ensemble machine learning methods, *Random Forest* and *Gradient Boosting*, do not show any significant difference in their performance with varying vectorization approaches. However, the probabilistic machine learning methods, *Logistic Regression* and *Naive Bayes*, achieve a higher performance by using *clfd*-based feature vectors. In fact, our *b-clfd* vectorization approach consistently increases the performance of *Logistic Regression* in Dataset 1, Dataset 2 and Dataset 3 by at least 1.88%, 1.9% and 0.58% respectively. The only exception is the performance of *Naive Bayes* in Dataset 3 which is slightly higher (0.17%) with *tf-idf* feature vectors. Therefore, there is a consistent high ranking for machine learning methods that utilize our *clfd* vectorization approach, as it allows them to achieve comparable or higher results in all datasets. The same pattern of results also appears in other metrics such as accuracy, precision and recall. The best performing traditional machine learning method is *Logistic Regression with b-clfd vectorization*, as it consistently provides the highest results, outperforming the other methods in F-1 score in Datasets 1, 2 and 3 by at least 2.87%, 2.95% and 1.35% respectively.

Finally, we report that the performance of *clfd*-based methods is not affected by the quality or absence of data preprocessing. As an example, consider *Logistic Regression (LR)* for Dataset 1. Without preprocessing, *LR* with *b-clfd* provides results that are actually better than its results with preprocessing (Table 6), with an F-1 score of 95.71%, while the F-1 scores of *LR with tf* and *LR with tf-idf* are reduced to 91.73% and 91.75% respectively.

4.3. Performance against deep learning and state-of-the-art

Dataset 1

	Accuracy	Precision	Recall	F-1 score
Bali 2019	87.3	89	87	89
lstm	90.9	92.4	88.96	90.65
Bi-lstm	92.09	92.25	91.76	92
cnn+lstm	92.22	90.33	94.41	92.33
Mult.lstm	92.62	90.51	**95.08**	92.74
LR+bclfd	**94.53**	**94.6**	94.64	**94.61**

Dataset 2

	Accuracy	Precision	Recall	F-1 score
Bali 2019	91.05	93	94	94
lstm	94.54	96.43	91.74	94.02
Bi-lstm	95	95.45	93.8	94.62
cnn+lstm	96.55	97.11	95.48	96.29
Mult.lstm	94.68	95.86	92.64	94.22
LR+bclfd	**97.52**	**97.17**	**97.65**	**97.41**

Dataset 3

	Accuracy	Precision	Recall	F-1 score
lstm	96.24	96.61	97.63	97.12
Bi-lstm	95.64	96.69	96.36	96.63
cnn+lstm	**96.78**	**97.26**	97.78	**97.52**
Mult.lstm	95.96	97.2	96.54	96.87
LR+bclfd	96.21	95.22	**99.29**	97.17

Table 7: Comparison of machine learning methods

Regarding the deep learning methods' performance, we see in Table 7 that *cnn+lstm* outperforms the other deep learning methods for Datasets 2 and 3, while its performance is comparable to that of *mult.lstm* for Dataset 1. In Dataset 1, *lstm* and *Bi-lstm* have the highest precision score among deep learning methods, but are otherwise performing at least slightly worse than *cnn+lstm* in all other cases (regardless of evaluation metric or dataset). We thus consider *cnn+lstm* as the best performing deep learning method.

However, we notice in Table 7 that deep learning methods are outperformed in many occasions. In Dataset 1, *Logistic Regression* with *b-clfd* vectorization consistently achieves the highest performance. It outperforms all deep learning methods in accuracy, precision and F-1 score by at least 1.91%, 2.2% and 1.87% respectively. The *mult.lstm* achieves a slightly (0.44%) higher recall score. In Dataset 2, *Logistic Regression* with *b-clfd* consistently achieves the highest results across all metrics. It outperforms deep learning methods in accuracy, precision, recall and F-1 scores by at least 0.97%, 0.06%, 2.17% and 1.12% respectively. In Dataset 3, the results of deep learning methods and those of *Logistic Regression* with *b-clfd* are comparable. The best performing deep learning method, *cnn+lstm*, provides higher accuracy, precision and F-1 scores by 0.57%, 2.04% and 0.35% respectively. However, *Logistic Regression* with *b-clfd* feature vectors outperforms all other deep learning methods in those metrics. Furthermore, it achieves the highest overall recall score (at least 1.51% higher than others). Therefore, *Logistic Regression with b-clfd vectorization* achieves clearly higher results than deep learning methods in Datasets 1 and 2, while the results are comparable for Dataset 3.

Furthermore, for Datasets 1 and 2, our *clfd* vectorization approach allows *Logistic Regression* to outperform existing state-of-the-art work (Bali et al., 2019), which also utilizes traditional machine learning methods, by a significant margin. More specifically, for Dataset 1, it outperforms state-of-the-art results by 7.23% in accuracy, 5.6% in precision, 7.64% in recall and 5.61% in F-1 score. For Dataset 2, it outperforms state-of-the-art results by 6.47% in accuracy, 4.17% in precision, 3.65% in recall and 3.41% in F-1 score.

4.4. Evaluation of the Hybrid method

In this section, we discuss in detail the performance of our novel *hybrid* method. To begin, it is obvious from Equation 7 that *hybrid* is a generic method that allows for much flexibility and fine-tuning regarding the α parameter, which represents the weight of the *clfd*-boosted *Logistic Regression* component. Given this, in the following table we present an assessment of the performance of *hybrid* when trying a rather exhaustive list of different α weight values.

α	Dataset 1	Dataset 2	Dataset 3
0.1	92.45	96.49	97.69
0.2	92.7	96.7	97.81
0.3	93.09	96.92	97.87
0.4	93.49	97.29	98.11
0.5	93.95	97.79	**98.4**
0.6	94.76	**97.89**	98.34
0.7	95.25	97.7	98.28
0.8	**95.33**	97.6	98.18
0.9	94.99	97.49	98.00

Table 8: F-1 score of hybrid with different α weights

It is interesting to note that increasing α, and therefore progressively making the *clfd*-boosted *Logistic Regression* the strongest component of *hybrid*, results in an increase in the method's performance. However, this phenomenon occurs up to a certain threshold which is 0.8, 0.6 and 0.5 for datasets 1, 2 and 3 respectively. After these dataset-specific thresholds are reached, increasing the α weight results in a slight drop in performance.

We can also report that for *all* weight values of α which make the *clfd*-boosted *Logistic Regression* the strongest component (i.e., for all $\alpha > 0.5$ values), *hybrid* outperforms both traditional and deep learning methods across all datasets.

Summing up our observations from Table 8, we can infer that an α value of approximately 0.7 provides a consistently good (consistently close to the best observed) performance across all our specific datasets. Therefore this is the value of α considered henceforth in this and the next section (for instance, this is the value of α used for the results in the figures below).

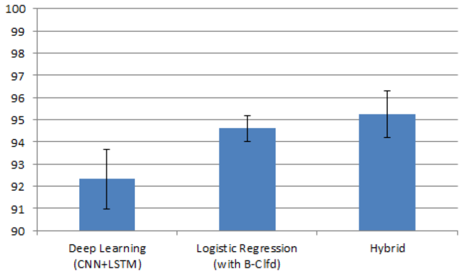

Figure 1: F-1 score for Dataset 1 (95% confidence)

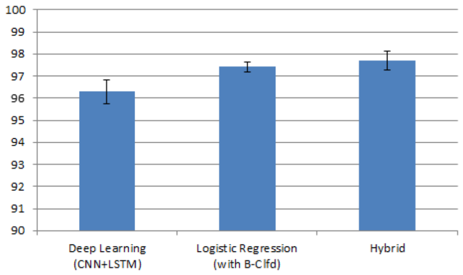

Figure 2: F-1 score for Dataset 2 (95% confidence)

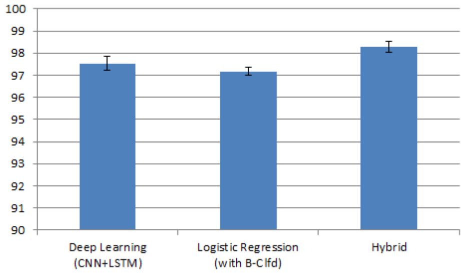

Figure 3: F-1 score for Dataset 3 (95% confidence)

Now, in Figures 1, 2 and 3 we present the mean F-1 scores of *hybrid* and the best traditional and deep learning techniques for the three datasets, along with error bars which correspond to 95% confidence intervals. We can see that *hybrid* consistently outperforms the other methods in terms of mean F-1 score, while *Logistic Regression* with *b-clfd* feature vectors has the lowest variance. In Datasets 1 and 2, *Logistic Regression* with *b-clfd* feature vectors provides results that are *significantly higher* than those of the best deep learning method, while they are comparable to those of *hybrid*. In Dataset 3, *Logistic Regression* with *b-clfd* feature vectors and deep learning methods provide results which are comparable to each other. However, in Dataset 3, *hybrid* provides results that are clearly superior to those of other methods, across all metrics. Specifically, we report that *hybrid* consistently outperforms deep learning methods in accuracy, precision, recall and F-1 score by at least 1.13%, 0.49%, 1.27% and 0.88% respectively.

4.5. Time Complexity

	Dataset 1	Dataset 2	Dataset 3
LR+bclfd	6.91	28.53	54.67
cnn+lstm	110.17	273.63	1295.32
Hybrid	112.22	292.88	1330.92

Table 9: Time comparison (sec)

Finally, in Table 9 we compare the best performing machine learning methods in terms of classification time required after data preprocessing. Note that this does not include the time required for the training of deep learning methods. We observe that *Logistic Regression* with *b-clfd* feature vectors is approximately 16, 10 and 24 times more time efficient than the other methods for Datasets 1, 2 and 3 respectively. The other two methods are comparable, with *hybrid* requiring approximately 5% more time than *cnn+lstm*. Time scalability is also shown in Figure 4, where the significant classification time advantage of *Logistic Regression* with *b-clfd* is clearly demonstrated.

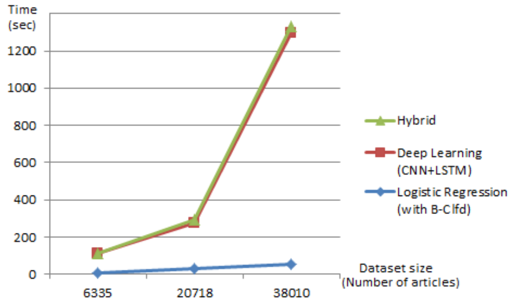

Figure 4: Time Scalability

4.6. Discussion of results

We demonstrated that our novel vectorization approach, *clfd*, is a simple and effective way for boosting the performance of certain machine learning methods. The experimental results show that a *clfd*-based vectorization approach consistently provides comparable or better results than other vectorization techniques, such as *tf* or *tf-idf*, for traditional machine learning methods. In addition, the experimental results also show that the *Logistic Regression* method can outperform deep learning methods if *b-clfd* feature vectors are used. More specifically, it provides consistently higher results than all deep learning methods for small and medium sized datasets (Datasets 1 and 2), while there is no significant difference in performance for large datasets (Dataset 3). However, compared to the deep learning methods, *Logistic Regression* with *b-clfd* feature vectors requires only a fraction of the cost paid for classification time. This can be of particular interest to online news agencies, as it can be used for the quick and effective classification of news articles in an online fashion. Moreover, for large datasets (Dataset 3), the *hybrid* method successfully combines a *cnn+lstm* neural network with *Logistic Regression* which utilizes *b-clfd* feature vectors, to achieve the highest performance and significantly outperform deep learning methods across all metrics.

5. Conclusions and Future Work

In this work we proposed a novel text vectorization technique, *clfd*; discussed its advantages with respect to “classic” vectorization approaches; and showed that its employment by machine learning methods can lead to the development of a content-based fake news detection system that achieves high performance and requires a minimal amount of classification time. In particular, we showed that *clfd* can turn *Logistic Regression* into a method that is a winner for small and medium-sized datasets; a method the classification performance of which is comparable to that of deep neural networks in large datasets, while it retains the advantage of minimal classification time. Moreover, when used as a component of our novel *hybrid* method, its performance clearly surpasses that of “pure” deep learning methods across all metrics, even for large datasets.

As a natural next step, we intend to evaluate our approach in other domains of interest. In ongoing work, we have already applied *clfd* in a sentiment analysis setting, with preliminary results that confirm the technique’s effectiveness. Future work also includes improving the *hybrid* method further, via equipping it with a more sophisticated weighting scheme. Moreover, additional machine learning methods can be added to its existing pool of methods, to increase its diversity and potentially its power. More advanced deep learning architectures, such as hierarchical attention networks (Yang et al., 2016), can also be considered. Finally, another interesting research direction is assessing the ways and extent to which the articles’ titles can aid classification (Horne and Adali, 2017).

6. Bibliographical References

Badaskar, S., Agarwal, S., and Arora, S. (2008). Identifying real or fake articles: Towards better language modeling. In *Proceedings of the Third International Joint Conference on Natural Language Processing: Volume-II*.

Bajaj, S. (2017). The pope has a new baby! fake news detection using deep learning. Technical report, Stanford Univ.

Bali, A. P. S., Fernandes, M., Choubey, S., and Goel, M. (2019). Comparative performance of machine learning algorithms for fake news detection. In *International Conference on Advances in Computing and Data Sciences*, pages 420–430. Springer.

Bishop, C. M. (2006). *Pattern recognition and machine learning*. Springer.

Conroy, N. J., Rubin, V. L., and Chen, Y. (2015). Automatic deception detection: Methods for finding fake news. *Proceedings of the Association for Information Science and Technology*, 52(1):1–4.

Eurobarometer. (2018). Final results of the Eurobarometer on fake news and online disinformation. Available at: https://ec.europa.eu/digital-single-market/en/news/final-results-eurobarometer-fake-news-and-online-disinformation. [Online; accessed 25-April-2019].

Horne, B. D. and Adali, S. (2017). This just in: Fake news packs a lot in title, uses simpler, repetitive content in text body, more similar to satire than real news. In *Eleventh International AAAI Conference on Web and Social Media*.

Ikagai Law. (2018). The ”fake news” problem: What are governments doing about it? Available at: https://www.ikigailaw.com/the_fake_news_problem. [Online; accessed 26-February-2019].

Kaggle. (2017). Fake News. Available at: https://www.kaggle.com/c/fake-news/data. [Online; accessed 20-January-2019].

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Manning, C. and Schütze, H. (1999). *Foundations of statistical natural language processing*. MIT press.

Manning, C., Raghavan, P., and Schütze, H. (2010). Introduction to information retrieval. *Natural Language Engineering*, 16(1):100–103.

McIntire, G. (2017). Fake news dataset. Available at: https://github.com/docketrun/Detecting-Fake-News-with-Scikit-Learn/tree/master/data. [Online; accessed 17-November-2018].

Patterson, J. and Gibson, A. (2017). *Deep learning: A practitioner's approach*. O'Reilly Media, Inc.

Pennington, J., Socher, R., and Manning, C. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Pérez-Rosas, V., Kleinberg, B., Lefevre, A., and Mihalcea, R. (2017). Automatic detection of fake news. *arXiv preprint arXiv:1708.07104*.

Rashkin, H., Choi, E., Jang, J. Y., Volkova, S., and Choi, Y. (2017). Truth of varying shades: Analyzing language in fake news and political fact-checking. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2931–2937.

Rimer-Boston, S. (2017). Is fact-checking 'fake news' a waste of time? Available at: https://www.futurity.org/fact-checking-fake-news-1475152/. [Online; accessed 26-February-2019].

Risdal, M. (2016). Getting Real about Fake News. Available at: https://www.kaggle.com/mrisdal/fake-news. [Online; accessed 20-January-2019].

Roy, A., Basak, K., Ekbal, A., and Bhattacharyya, P. (2018). A deep ensemble framework for fake news detection and classification. *arXiv preprint arXiv:1811.04670*.

Sokolova, M. and Lapalme, G. (2009). A systematic analysis of performance measures for classification tasks. *Information processing & management*, 45(4):427–437.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958.

Vosoughi, S., Roy, D., and Aral, S. (2018). The spread of true and false news online. *Science*, 359(6380):1146–1151.

Yang, Z., Yang, D., Dyer, C., He, X., Smola, A., and Hovy, E. (2016). Hierarchical attention networks for document classification. In *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies*, pages 1480–1489.