UNIVERSITY OF THESSALY

Diploma Thesis

Machine learning algorithms for fake news detection

Αλγόριθμοι μηχανικής μάθησης για την αναγνώριση ψευδών ειδήσεων

Author:
Georgios
STAVROPOULOS

Supervisors:
Dimitrios KATSAROS
Gerasimos POTAMIANOS

A thesis submitted in fulfillment of the requirements for the degree of Diploma Thesis

in the

Department of Electrical And Computer Engineering

July 4, 2018

Declaration of Authorship

I, Georgios STAVROPOULOS, declare that this thesis titled, "Machine learning algorithms for fake news detection" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: _____

Date: _____

"I am not young enough to know everything"

Oscar Wilde

ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

Περίληψη

Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

Διπλωματική Εργασία

Αλγόριθμοι μηχανικής μάθησης για την αναγνώριση ψευδών ειδήσεων

Γεώργιος Σταυρόπουλος

Η αναγνώριση ψευδών ειδήσεων αποκτά σταδιακά πρωταρχική σημασία για την κοινωνία προκειμένου να αποφευχθεί το λεγόμενο **reality-vertigo** και να προστατευθούν ιδιαίτερα τα λιγότερο μορφωμένα άτομα ή ακόμη και τα μορφωμένα σε τομείς όπως τα ιατρικά νέα όπου η τεχνογνωσία τους δεν επαρκεί για την άμεση αναγνώριση των ψευδών ειδήσεων. Διάφορες τεχνικές μηχανικής μάθησης έχουν προταθεί εώς τώρα για την αντιμετώπιση αυτού του ζητήματος. Η διπλωματική αυτή παρουσιάζει μια ολοκληρωμένη αξιολόγηση της απόδοσης οκτώ αλγορίθμων μηχανικής μάθησης για την αναγνώριση/ταξινόμηση ψευδών ειδήσεων. Αυτές οι μέθοδοι βασίζονται στην Παλινδρόμηση (Regression), τις Μηχανές Διανυσμάτων Υποστήριξης (SVMs), τα Νευρωνικά Δίκτυα (Neural Networks), τα Δέντρα Αποφάσεων (Decision Trees) και τη θεωρία Bayes. Η αξιολόγηση διεξάγεται σε σχέση με τρία διαθέσιμα **datasets** και απαντά σε ερωτήσεις που σχετίζονται με το είδος του **training** και τον αριθμό των διαστάσεων που θα πρέπει να χρησιμοποιηθούν για κάθε αλγόριθμο και τελικώς, σχετικά με τη μέθοδο δημιουργίας διανυσματικών αναπαραστάσεων κειμένου. Όπως αναμενόταν, η μελέτη αυτή επιβεβαιώνει το αναμενόμενο αποτέλεσμα, ότι δηλαδή η φύση των δεδομένων είναι η παράμετρος που καθορίζει την απόδοση κάθε αλγορίθμου. Παρ' όλα αυτά, η μελέτη ρίχνει φως και αντλεί γενικά συμπεράσματα σχετικά με το ζήτημα του αριθμού των διαστάσεων που πρέπει να χρησιμοποιηθεί ως χαρακτηριστικό των αλγορίθμων και του είδους των διανυσματικών αναπαραστάσεων του κειμένου.

UNIVERSITY OF THESSALY

Abstract

Department of Electrical And Computer Engineering

Diploma Thesis

Machine learning algorithms for fake news detection

by Georgios STAVROPOULOS

Fake news detection/classification is gradually becoming of paramount importance to out society in order to avoid the so-called reality vertigo, and protect in particular the less educated persons or even the educated ones in fields such as medical news where their expertise is not adequate to immediate understand the fraudulent articles. Various machine learning techniques have been proposed to address this issue. This thesis presents a comprehensive performance evaluation of eight machine learning algorithms for fake news detection/classification. These methods are based on Regression, Support Vector Machines, Neural Networks, Decision Trees and Bayes theory. The evaluation is conducted against three publicly available datasets, and answers questions related to the number of dimensions that should be used for each algorithm, to the kind of training that must be performed, and finally related to the method of generating vector representations of textual information. As expected, this study confirms the expected result that the nature of data is the parameter that defines the performance of each algorithm. Nevertheless, this study sheds light and draws safe generic conclusions with respect to the issue of the number of dimensions that should be used as features for the algorithms and the kind of vector representation of the information.

Acknowledgements

First and foremost, I am particularly grateful to my advisor Dimitrios Katsaros for all his help, the valuable critique and the time he has spent during the development of this thesis. His guidance and support has been instrumental in helping me bring this work to a successful end. I would also like to thank all of my friends, old and new, that made these 5 years a great experience. Finally, I would like to thank my family, for their support and their continuous efforts until today.

Thank you all!

Contents

List of Abbreviations

LR	**L**ogistic **R**egression
SVM	**S**upport **V**ector **M**achines
MLP	**M**ulti **L**ayer **P**erceptron
GNB	**G**aussian **N**aive **B**ayes
MNB	**M**ultinomial **N**aive **B**ayes
DT	**D**ecision **T**rees
RF	**R**andom **F**orests
CNN	**C**onvolutional **N**eural **N**etworks

To my dear Mirsini...

Chapter 1

Introduction

1.1 Context

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 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 by 2020, people in developed countries will encounter more fake 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 [12] shows that false medical information gets more views, likes, comments than true medical information. Even worse, fake news are not only (more) popular, but they are spreading at a faster pace [20] than real news.

So, countermeasures against fake news started to develop rapidly. There are already some fact-checking organizations, such as snopes.com, politifact.com, fact-check.com, truthorfiction.com. Efforts are taking place to deploy fact checking services in browsers; a notable effort by two Yalies which have garnered media attention[2] is the development of a Google Chrome extension to combat fake news. Other anti-fake news techniques include tagging with "Disputed by 3^{rd} party fact-checkers", and adding publisher logos to the information items.

1.2 Thesis Motivations & 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 crowd-sourced verification efforts could be a solid

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

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

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 proven very effective in combating spam email, which is one type of misinformation; so, algorithms belonging to this category of techniques were among the very first whose efficacy has been investigated. Support Vector Machines (SVM), Neural Networks (NN), Bayes Classifiers are some popular families of machine learning algorithms. In particular, the following algorithms have been examined in the context of fake news detection:

- L1 Regularized Logistic Regression
- C-Support Vector Classification
- Multi-Layer Perceptron
- Gaussian Naive Bayes
- Multinomial Naive Bayes
- Decision Trees
- Random Forests
- Convolutional Neural Networks

However, their relative performance is unknown, and so is their generic behavior when tested against diverse datasets. The aim of this thesis is to answer these two broad questions. In this context the present thesis makes the following 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.
- It introduces a public Web-based application to test the competitors against any real URL for possible fake news.

The rest of the thesis is organized as follows: Chapter 2 presents briefly the related work. Chapter 3 introduces the algorithms that will be evaluated, the preprocessing that was made and the word representation techniques. Chapter 4 describes the evaluation environment, i.e., competitors, datasets, performance measures, and on, and also presents the actual evaluation of the competing algorithms. Chapter 5 presents the Web-Based Application that was made in order to make our findings publicly available. Finally, Chapter 6 concludes this thesis and presents the future work.

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 [7]. A credible news classifier based on regression was proposed in [8]. SVM on content-based features was utilized in [9] 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 [16].

A different line of research was taken by [4, 10] 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.

Finally, some works investigated the issue of fake news detection following a credibility diffusion-based approach. All these works [6] construct complex networks of heterogeneous entities (persons, tweets, events, message, etc) and study the paths of fake news propagation in order to find out non-credible sources of information, and thus infer fake news.

There are academic efforts to develop online services which will study how misinformation spreads and competes in online social networks. For instance Hoaxy[1] [12] is such a service for Twitter; it as actually a platform for the study of diffusion of misinformation in Twitter.

Similarly to this thesis' work, [22] did a research on the topic but emphasized more in Decision Trees, Random Forests and Gradient Descent algorithms. He concludes that term frequency is potentially predictive of fake news which is an important first step toward using machine classification for identification and that there is definitely scope for more improvements.

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 that areas and fake news detection as explained in [16], and thus we do not consider them here. Finally, there are algorithms for detecting fake images online [5], but these are beyond the scope of the current thesis.

On the other hand, there are some more precise efforts that are based on expert-oriented fact-checking which obviously heavily relies on human

[1] https://hoaxy.iuni.iu.edu/

domain experts to investigate relevant data and documents to construct the verdicts of claim veracity, for example PolitiFact[2] and Snopes[3]. However, expert-oriented fact-checking is a very demanding and time-consuming process, which limits the potential for high efficiency and scalability. So although it seems like a more confident solution to the problem, because of the volume of information it becomes imposssible.

Other than expert-oriented efforts, Crowdsourcing-oriented fact-checking has given the opportunity to normal people to annotate news content. These annotations are then aggregated to produce an overall assessment of the news veracity. For example, Fiskkit[4] allows users to discuss and annotate the accuracy of specific parts of a news article. As another example, an anti-fake news bot named "For real" is a public account in the instant communication mobile application LINE[5], which allows people to report suspicious news content which is then further checked by editors.

[2]http://www.politifact.com/
[3]http://www.snopes.com
[4]http://fiskkit.com/
[5]https://grants.g0v.tw/projects/588fa7b382223f001e022944

Chapter 3

Models And Approch

3.1 Preprocessing

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 [18]. This was done in order to avoid noise in our data and make classification faster and more efficient.

Stop Word Removal

Stop words are insignificant words in a language that will create noise when used as features in text classification.These are words commonly used in a lot sentences to help connect thought or to assist in the sentence structure. Prepositions, conjunctions and some pronouns are considered stop words. We removed common words such as, a, an, i, me, my, we, the, and, but, if, as, of, at, by, for, there, when, where, why, will, etc. Those words were removed from each document, and the processed documents were stored and passed on to the next step.

Stemming

Following the Stop Word Removal, Stemming, simply, is changing the words into their original form, and decreasing the number of word types or classes in the data. For example, the words "Running," "Ran" and "Runner" will be reduced to the word "run." We use stemming to make classification faster and efficient. It is important to appreciate that we use stemming with the intention of improving the performance of our models. It is not an exercise in etymology or grammar. In fact from an etymological or grammatical viewpoint, a stemming algorithm is liable to make many mistakes.

3.2 Representation

In order to use a machine learning algorithm or a statistical technique on any form of text, it is prescribed 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:

3.2.1 Term Frequency-Inverse Document Frequency

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. More specifically,

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

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

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

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

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 doesn't occur in the document, we assign this weight equal to zero.

3.2.2 Word Embeddings

A word embedding is a parameterized function mapping words in 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 [17] trained on 6B tokens of Wikipedia 2014 + Gigaword 5. The vectors have dimensionality of 50, 100, 300.

- *Trained Word Vectors Based on the 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 architecture.

In order to get a single vector representation within each headline/article we averaged the corresponding word vectors.

3.3 Competing algorithms

The version of the first seven algorithms is that provided by scikit-learn[1], whereas for the last one we developed our own code according to [11].

3.3.1 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 $\| \bullet \|_1$ 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:

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

With L1 regularization only a sparse subset of the most important inputs is used and neurons become nearly invariant to the noisy inputs.

3.3.2 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 $x_i \in R_n$ i=1,. . . ,l in the two class case and the corresponding class labels decision $y_i \in$ {-1,1}, the statement of C-SVC optimization for classification problems may be the following [3, 19]:

$$\min_{w,b,\xi} \frac{1}{2} w^T w + C \sum_{i=1}^{l} \xi_i$$

with constraints: $y_i(w^T \phi(x_i) + b \geq 1 - \xi_i$ where ξ_i are the slack variables. The dual problem definition is:

$$\min_{\propto} \frac{1}{2} \propto^T Q \propto - e^T \propto,\ 0 \leq a_i \leq C,\ i = 1, ..., l$$

with constraints: $y^T \propto = 0$, where e is the vector of all ones, $C > 0$ is the upper bound, Q is a l by l positive semidefinite matrix, $Q_{ij} \equiv y_i y_j K(x_i, x_j)$ and

$$K(x_i, x_j) \equiv \phi(x_i)^T \phi(x_j)$$

[1] http://scikit-learn.org/stable/index.html

is the kernel. Function ϕ transforms training vectors x_i into a higher dimensional space. The decision function is:

$$\text{sgn}(\sum_{i=1}^{l} y_i \propto_i K(x_i, x) + b).$$

The choice of the appropriate kernel for a specific application is often a difficult task. A necessary and sufficient condition for a kernel to be valid is that it must satisfy Mercer's theorem, but other than that, there is really no mathematically structured approach to prefer one kernel to the other. Obviously, we would expect that a non-linear kernel based C-SVC would perform better than the one based on a linear kernel, if the data is known to be not linearly separable. The choice of kernel results indifferent kinds of C-SVCs with different performance levels. For the purpose of this research, radial basis function (RBF) is being used.

3.3.3 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. An MLP with a single hidden layer can be represented graphically as shown in Figure 3.1.

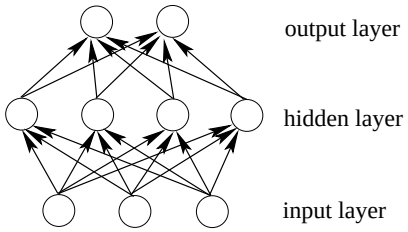

FIGURE 3.1: The topology of a multi-layer perceptron.

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

$$f(x) = G(b(2) + W(2)(s(b(1) + W(1)x)))$$

with $b(1), b(2)$ being the bias vectors, $W(1), W(2)$ the weight matrices, G the logistic sigmoid function and s the rectified linear unit function. To train the MLP, in order to learn the set of parameters $\{b(1), W(1), b(2), W(2)\}$ the L-BFGS optimization algorithm is used.

3.3.4 Gaussian and Multinomial Naive Bayes

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

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

Using the naive independence assumption that

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

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

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

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

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

The different naive Bayes classifiers differ mainly by the assumptions they make regarding the distribution of $P(x_i|y)$.

Gaussian Naive Bayes

The likelihood of the features is assumed to be Gaussian:

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

Multinomial Naive Bayes

Multinomial Naive Bayes implements the naive Bayes algorithm for multinomially distributed data. The distribution is parameterized by vectors

$$\theta_y = (\theta_{y_1}, \ldots, \theta_{y_n})$$

for each class y, where n is the number of features (in text classification, the size of the vocabulary) and θ_{y_i} is the probability $P(x_i|y)$ of feature i appearing in a sample belonging to class y. The parameter y is estimated by a smoothed version of maximum likelihood, i.e., relative frequency counting:

$$\hat{\theta_{yi}} = \frac{N_{yi} + a}{N_y + an}$$

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

3.3.5 Decision Trees

Despite the various decision tree algorithms, the type of decision tree that we used was first discussed by Breiman [1] 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:

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

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 $p(w_j|t)$.

3.3.6 Random Forests

Random forests is a popular and efficient algorithm which belongs to the family of ensemble methods. It was introduced by Breiman [2] and can be used for either classification or regression problems. During training, the algorithm creates multiple trees using the CART [1] 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.

3.3.7 Convolutional Neural Networks

The model architecture, shown in Figure 3.2, is a slight variant of the CNN architecture of Collobert et al. (2011) [23]. Let $x_i \in R^k$ be the k-dimensional word vector corresponding to the i-th word in the sentence. A sentence of

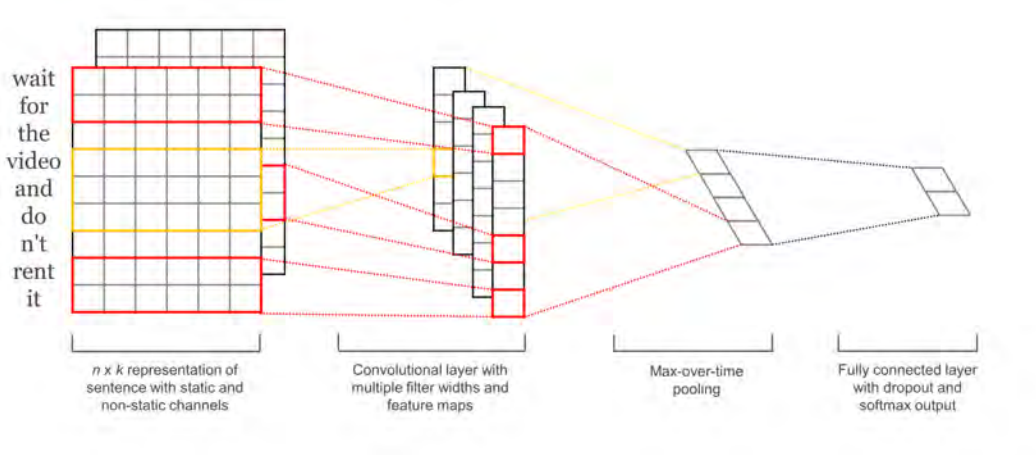

FIGURE 3.2: Model architecture with two channels for an example sentence.

length n (padded where necessary) is represented as

$$x_{1:n} = x_1 \oplus x_2 ... \oplus x_n \tag{3.1}$$

where $\oplus$ is the concatenation operator. In general, let $x_{i:i+j}$ refer to the concatenation of words $x_i, x_{i+1}, ..., x_i + j$. A convolution operation involves a filter $w \in R^{hk}$, which is applied to a window of h words to produce a new feature. For example, a feature c_i is generated from a window of words $x_{i:i+h-1}$ by

$$c_i = f(w \cdot x_{i:i+h-1} + b)$$

Here $b \in R$ is a bias term and f is a non-linear function such as the hyperbolic tangent. This filter is applied to each possible window of words in the sentence $\{x_{1:h}, x_{2:h+1}, ..., x_{n-h+1:n}\}$ to produce a feature map

$$c = [c_1, c_2, \ldots, c_{n-h+1}]$$

with $c \in R^{(n-h+1)}$. We then apply a max-over- time pooling operation (Collobert et al., 2011) over the feature map and take the maximum value $\hat{c} = \max\{c\}$ as the feature corresponding to this particular filter. The idea is to capture the most important feature–one with the highest value–for each feature map. This pooling scheme naturally deals with variable sentence lengths.

We have described the process by which *one* feature is extracted from *one* filter. The model uses multiple filters (with varying window sizes) to obtain multiple features. These features form the penultimate layer and are passed to a fully connected softmax layer whose output is the probability distribution over labels.

Regularization

For regularization dropout is employed on the penultimate layer with a constraint on $l2$-norms of the weight vectors (Hinton et al., 2012 [24]).

Dropout prevents co-adaptation of hidden units by ran- domly dropping out–i.e., setting to zero–a proportion p of the hidden units during foward-backpropagation. That is, given the penultimate layer $z = [\hat{c_1}, \ldots, \hat{c_m}]$ (note that here we have m filters), instead of using

$$y = w \cdot z + b \tag{3.2}$$

for output unit y in forward propagation, dropout uses

$$y = w \cdot (z \circ r) + b \tag{3.3}$$

where $\circ$ is the element-wise multiplication operator and $r \in R^m$ is a "masking" vector of Bernoulli random variables with probability p of being 1. Gradients are backpropagated only through the unmasked units. At test time, the learned weight vectors are scaled by p such that $\hat{w} = pw$, and $\hat{w}$ is used (without dropout) to score unseen sentences. We additionally constrain $l_2 - norms$ of the weight vectors by rescaling w to have $\| w \|_2 = s$ whenever $\| w \|_2 > s$ after a gradient descent step.

Chapter 4

Experimental Setup and Evaluation

4.1 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 intensive task compared to the rest of the algorithms. The following table has the detailed specifications of the machines used in our experiments.

	Server Name 1	Server Name 2
CPU Architecture	Haswell	Ivy Bridge
Model No.	Xeon E5-2695V3	Xeon E5-2620V2
# of Cores	14	6
Core Frequency	2.30 GHz	2.10 GHz
Main Memory	128 GB	128 GB
GPU	Nvidia Tesla K20x	None

TABLE 4.1: Server Specifications

4.2 Datasets

We strived for using freely available datasets that have been used in earlier studies, to ease reproducibility. The datasets are described in Table 4.2.

Before using any of our datasets, firstly we subjected them to some refinements that were described in Section 3.1. As it was already mentioned, this was done in order to make our data less "noisy" and speedup the classification.

Using the datasets from Table 4.2, we created three input datasets (experiments) on which we evaluated the algorithms. For the first experiment we used the Wang's training dataset [21] which contains various statements from PolitiFact[3], 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).

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

TABLE 4.2: Datasets used in the evaluation.

dataset name	**Dataset properties**		
	size	***property***	***source***
"Liar, liar pants on fire": A new benchmark dataset for fake news detection	Training set size of 10269 articles	Two labels for the truthfulness ratings (real/fake) were used instead of the original six	[21]
The Signal Media One-Million News Articles Dataset	1 million articles	13000 articles were selected at random and marked as real news	Signalmedia[1]
Getting Real about Fake News	13000 articles	All 13000 articles were marked as fake news	Kaggle[2]

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 2016 was between 200–1000 words[4], we collected 10000 body texts of a length between 150–4000 words. We will call these three datasets as Dataset 1, Dataset 2 and Dataset 3.

4.3 Performance measures

In this section, we'll define the primary metrics we'll use to evaluate the classification models.

A *true positive* is an outcome where the model correctly predicts a fake news article. Similarly, a *true negative* is an outcome where the model correctly predicts a non-fake news article.

A *false positive* is an outcome where the model incorrectly predicts a fake news article. And a *false negative* is an outcome where the model incorrectly predicts a non-fake news article.

[4]https://www.newswhip.com/2017/01/long-shared-stories-social-media/

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. More precisely:

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}} = \frac{TP + TN}{TP + TN + FP + FN}$$

- *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:

$$\text{Recall} = \frac{TP}{TP + FN}$$

  - *Precision* is the percentage of news items being actually fake out of all news items returned as fake by the model:

$$\text{Precision} = \frac{TP}{TP + FP}$$

  So *F1-Measure* can be written as:

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

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.

4.4 Performance Evaluation

4.4.1 K-Fold Cross-Validation

Learning the parameters of a prediction function and testing it on the same data is a methodological mistake: a model that would just repeat the labels of the samples that it has just seen would have a perfect score but would fail to predict anything useful on yet-unseen data. This situation is called overfitting. To avoid it, it is common practice when performing a (supervised) machine learning experiment to hold out part of the available data as a test set ($Xtest, Ytest$).

When evaluating different settings ("hyperparameters") for estimators (such as the C parameter that must be manually set for an SVM) there is still a risk of overfitting on the test set because the parameters can be tweaked until the estimator performs optimally. This way, knowledge about the test set can "leak" into the model and evaluation metrics no longer report on generalization performance. To solve this problem, yet another part of the dataset can be held out as a so-called "validation set": training proceeds on the training set, after which evaluation is done on the validation set, and when the experiment seems to be successful, final evaluation can be done on the test set.

However, by partitioning the available data into three sets, we drastically reduce the number of samples which can be used for learning the model, and the results can depend on a particular random choice for the pair of (train, validation) sets.

A solution to this problem is a procedure called cross-validation (CV for short). A test set should still be held out for final evaluation, but the validation set is no longer needed when doing CV. In the basic approach, called k-fold CV, the training set is split into k smaller sets (other approaches are described below, but generally follow the same principles). The following procedure is followed for each of the k "folds":

- A model is trained using $k - 1$ of the folds as training data;
- the resulting model is validated on the remaining part of the data (i.e., it is used as a test set to compute a performance measure such as accuracy).

The performance measure reported by k-fold cross-validation is then the average of the values computed in the loop. This approach can be computationally expensive, but does not waste too much data (as it is the case when fixing an arbitrary test set), which is a major advantage in problem such as inverse inference where the number of samples is very small.

In our tests we chose $K = 10$ which is generally preferred by most researches and can be represented graphically as shown in Figure 4.1

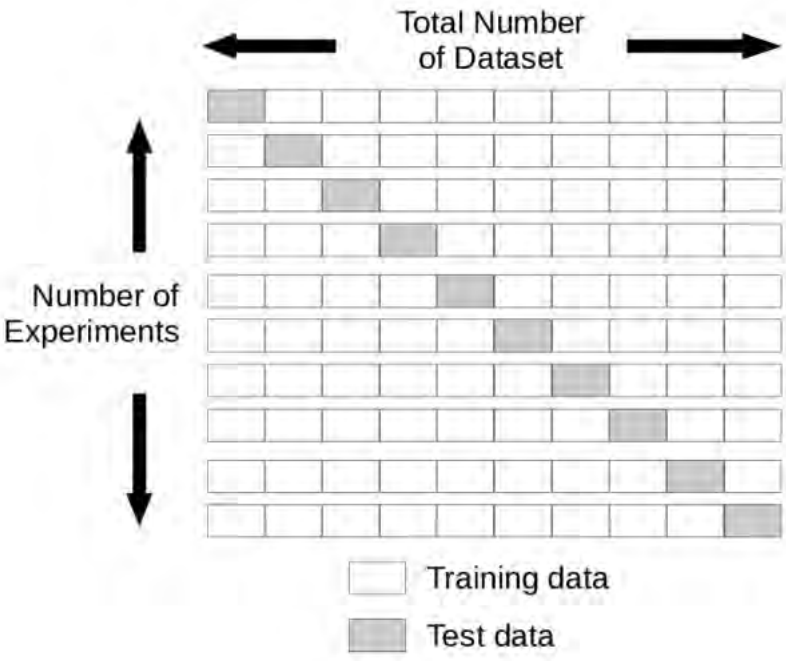

FIGURE 4.1: 10-Fold Cross-Validation

4.4.2 Execution time

As far as the execution time is concerned, Tables 4.3–4.5 show the execution time – training and classification time – of all variants of the algorithms for Dataset 1, Dataset 2 & Dataset 3. In general, SVM and the neural network-based algorithms are the most time-consuming during the training phase, which is expected.

TABLE 4.3: Training/classification times (in seconds) Times (in seconds) on Dataset 1

				Glove Vectors			
Model	50D	100D	300D	50D	100D	300D	TF/IDF
LR	0.69 0.01	0.97 0.01	0.58 0.01	5.75 0.01	7.36 0.0	3.13 0.01	0.04 0.0
MLP	8.37 0.0	7.4 0.0	11.45 0.0	8.12 0.0	6.45 0.0	10.74 0.0	8.46 0.0
DT	1.1 0.0	02.01 0.0	6.39 0.0	1.1 0.0	1.76 0.0	5.44 0.0	0.58 0.0
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 0.0	0.01 0.0	0.03 0.01	0.01 0.0	0.01 0.0	0.03 0.01	0.05 0.01
MNB	0.01 0.0	0.01 0.0	0.03 0.0	0.01 0.0	0.01 0.0	0.02 0.0	0.00 0.00
SVM	14.44 01.08	19.08 1.68	54.86 4.81	13.04 01.09	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	

TABLE 4.4: Training/classification times (in seconds) Times (in seconds) on Dataset 2

				Glove Vectors			
Model	50D	100D	300D	50D	100D	300D	TF/IDF
LR	12.85 0.01	13.82 0.01	8.57 0.01	128.28 0.01	154.76 0.0	39.14 0.01	0.03 0.0
MLP	16.57 0.01	15.47 0.0	25.49 0.01	15.33 0.0	13.39 0.0	20.67 0.01	10.69 0.0
DT	2.64 0.0	5.74 0.0	18.48 0.0	2.6 0.0	5.44 0.0	17.81 0.0	0.06 0.0
RF	2.82 0.01	4.01 0.01	6.95 0.02	2.82 0.01	4.03 0.01	6.87 0.02	0.29 0.01
GNB	0.03 0.0	0.05 0.0	0.19 0.02	0.03 0.0	0.04 0.0	0.17 0.01	0.03 0.0
MNB	0.01 0.0	0.02 0.0	0.04 0.0	0.01 0.0	0.02 0.0	0.04 0.0	0.01 0.0
SVM	81.71 6.94	159.31 11.78	350.03 28.46	77.13 6.36	149.52 11.37	365.56 30.93	21.01 1.77
CNN	25.85 0.63	36.26 0.76	56.09 0.7	26.41 0.6	30.35 0.6	56.49 0.64	

TABLE 4.5: Training/classification times (in seconds) (in seconds) on Dataset 3

				Glove Vectors			
Model	50D	100D	300D	50D	100D	300D	TF/IDF
LR	4.32 0.0	12.36 0.0	3.99 0.0	23.07 0.0	31.59 0.0	33.04 0.01	0.12 0.0
MLP	5.8 0.0	6.6 0.0	10.26 0.0	5.69 0.0	06.07 0.0	10.1 0.0	27.26 0.01
DT	0.72 0.0	1.59 0.0	4.74 0.0	0.76 0.0	1.49 0.0	4.72 0.0	3.3 0.0
RF	0.78 0.01	1.13 0.01	1.96 0.01	0.77 0.01	1.1 0.01	1.95 0.01	1.18 0.01
GNB	0.01 0.0	0.01 0.0	0.03 0.01	0.01 0.0	0.01 0.0	0.03 0.01	0.56 0.03
MNB	0.0 0.0	0.01 0.0	0.02 0.0	0.0 0.0	0.01 0.0	0.02 0.0	0.01 0.0
SVM	12.18 0.97	19.39 1.69	48.14 4.36	12.42 1.0	17.95 1.58	52.57 4.85	44.83 4.3
CNN	21.95 0.93	32.71 0.97	66.25 0.9	21.46 0.84	31.01 0.9	67.12 0.95	

4.4.3 Algorithms Comparison

We ask the following two questions: *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 Figures 4.2–4.4. Deviation is small, so average is quite a good measure for all algorithms with the exception of *DT* for Dataset 1.

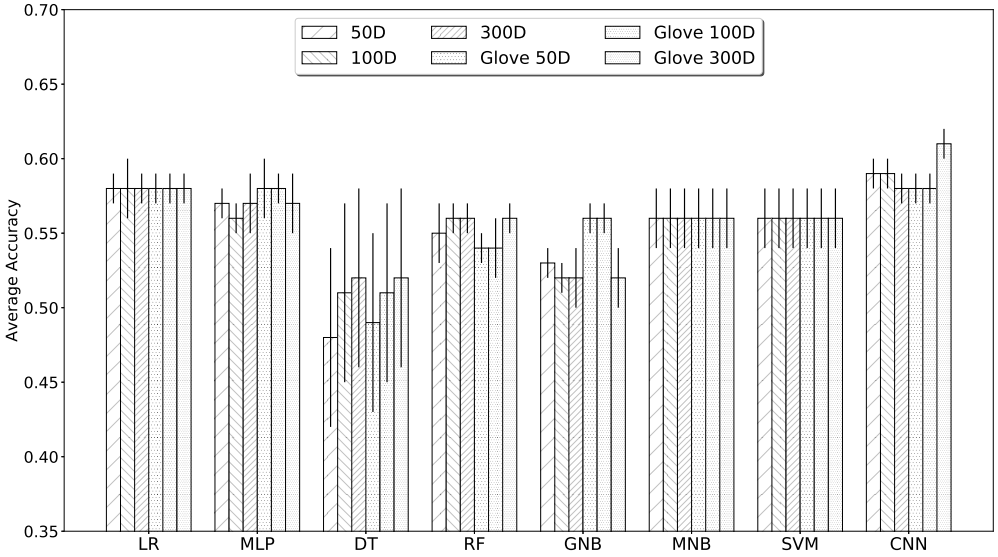

FIGURE 4.2: Average accuracies on Dataset 1.

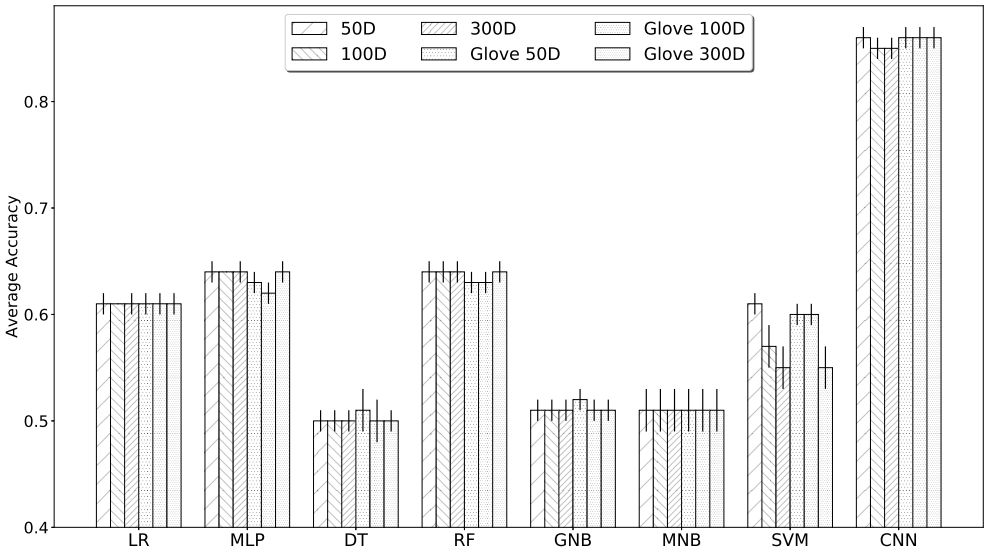

FIGURE 4.3: Average accuracies on Dataset 2.

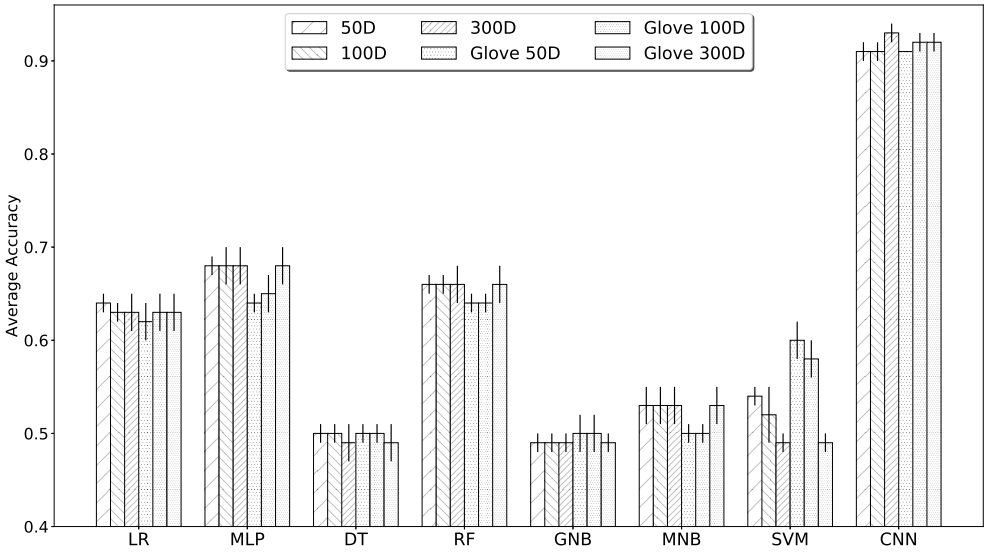

FIGURE 4.4: Average accuracies on Dataset 3.

We present the average F1-measure of the six variants of each algorithm in Figures 4.5–4.7.

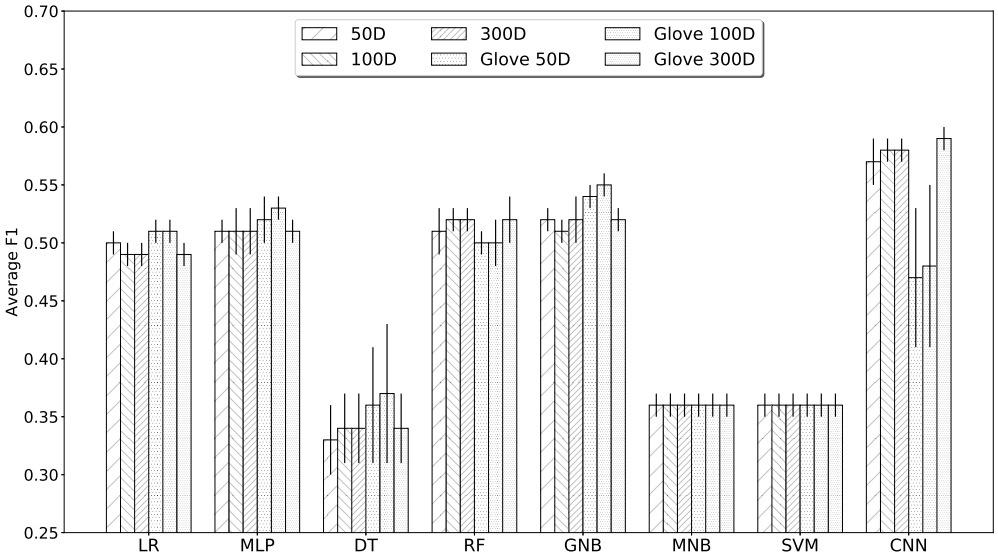

FIGURE 4.5: Average F1-measure on Dataset 1.

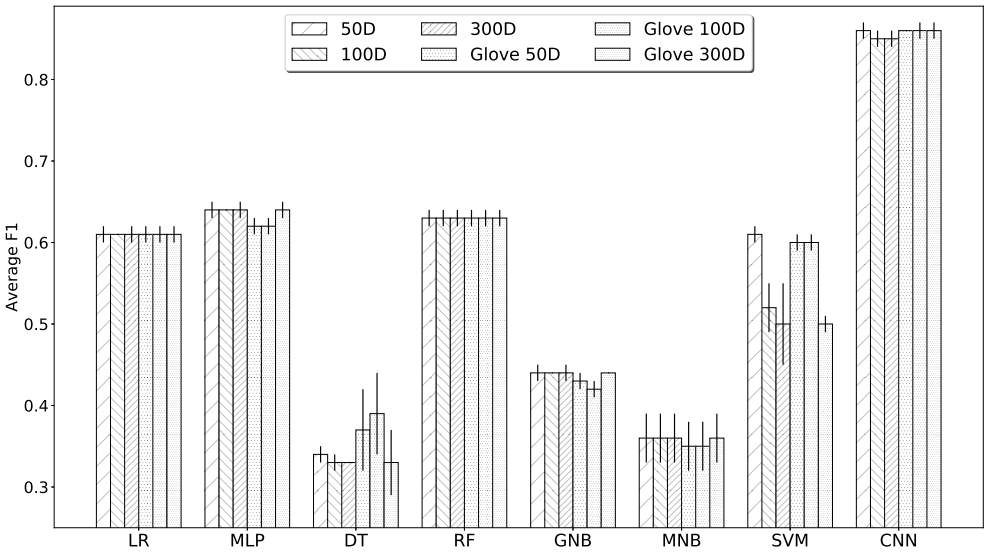

FIGURE 4.6: Average F1-measure on Dataset 2.

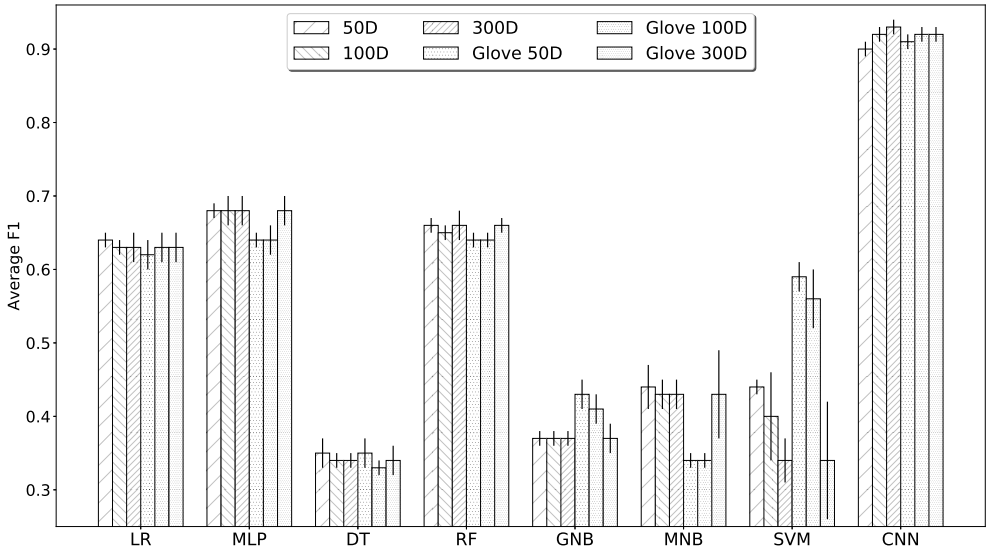

FIGURE 4.7: Average F1-measure on Dataset 3.

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 problems and the associated algorithms are highly dependent on data distributions. In Table 4.6 we present the variant of each algorithm that showed the best performance.

TABLE 4.6: Champion variant of each algorithm with respect to the number of dimensions and type of training.

Algorithm	*Dataset1*	*Dataset2*	*Dataset3*
LT	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

We can draw two quite evident conclusions from Table 4.6. The first observations 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.

4.4.4 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 is 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 Figures 4.8–4.13. 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 cases, in some cases this gap widens to reach a 30%. The only exception is for SVM in the case of Dataset 3.

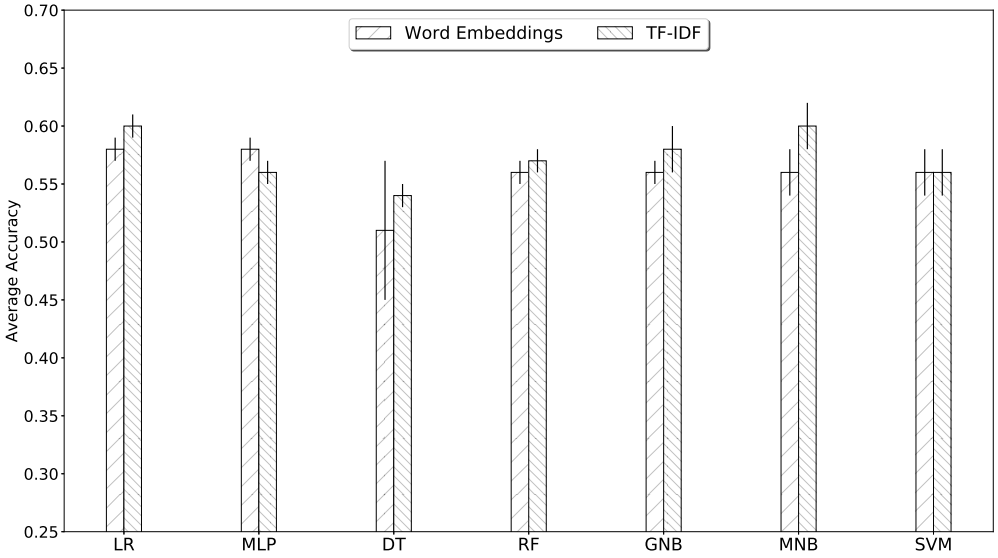

FIGURE 4.8: Average accuracies on Dataset 1.

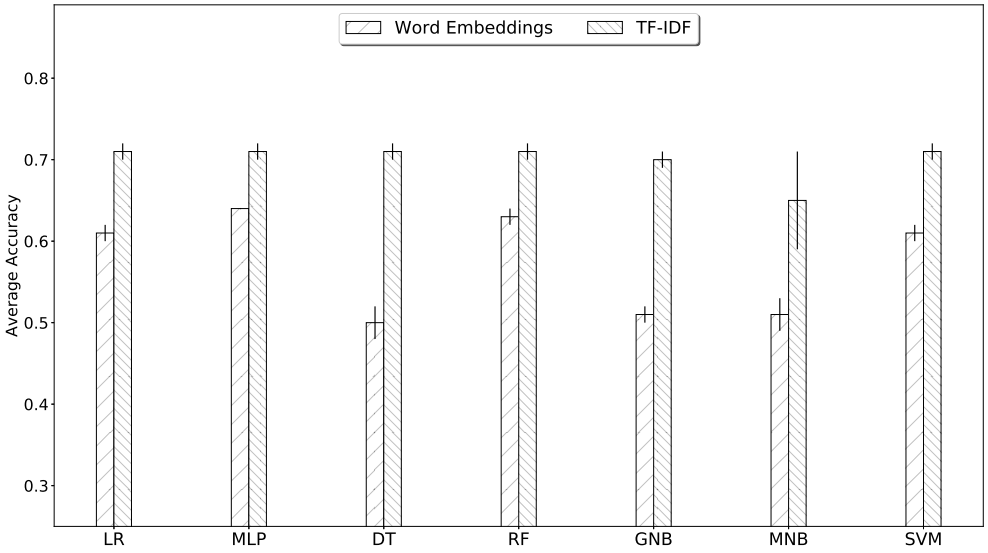

FIGURE 4.9: Average accuracies on Dataset 2.

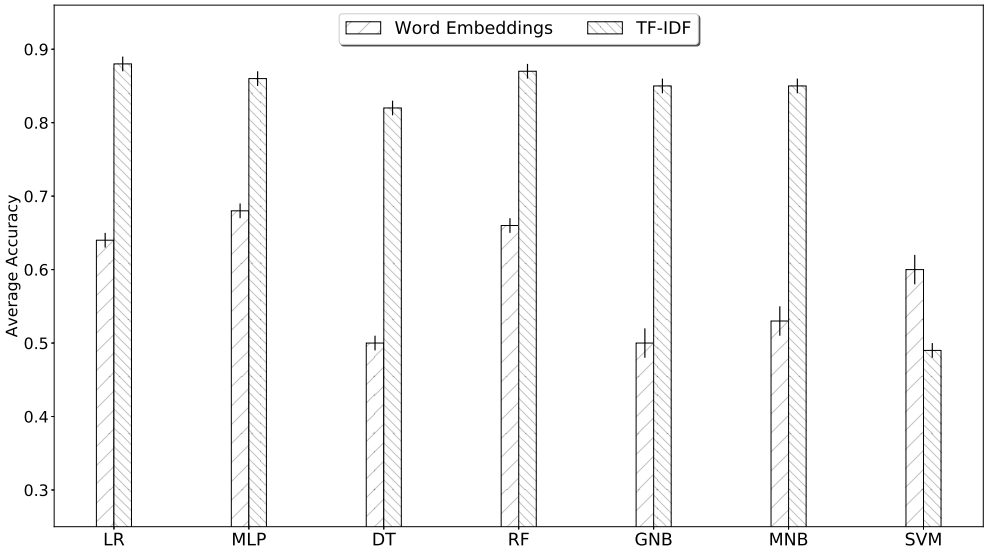

FIGURE 4.10: Average accuracies on Dataset 3.

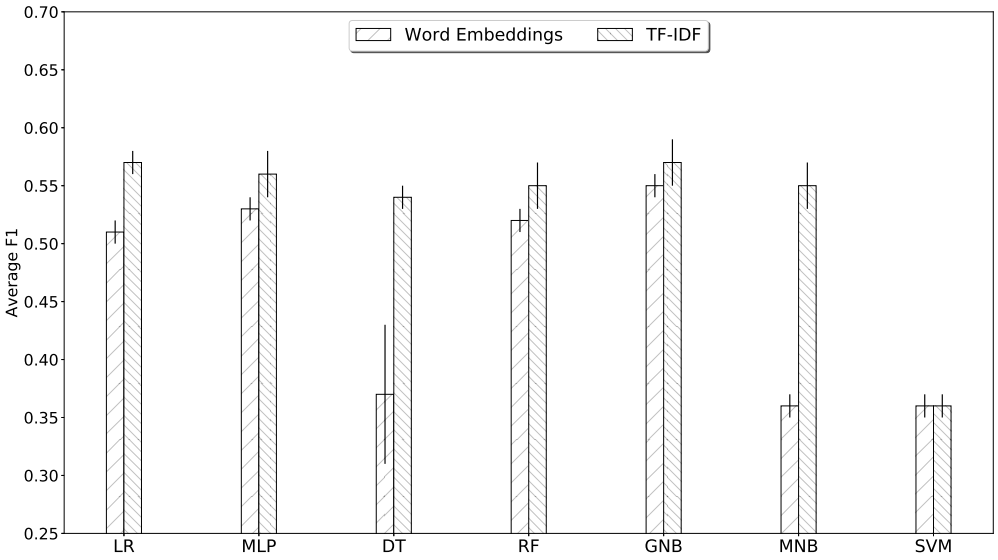

FIGURE 4.11: Average F1-measure on Dataset 1.

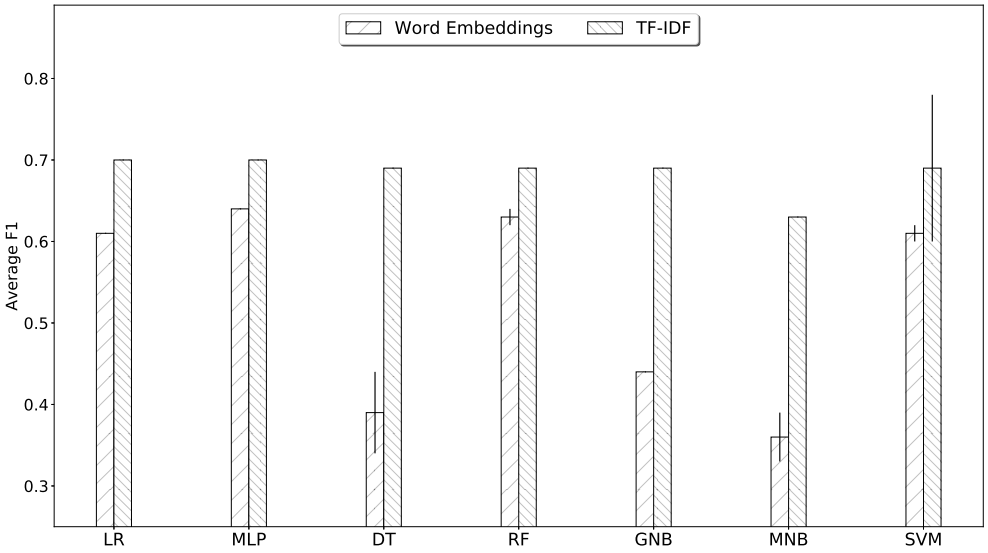

FIGURE 4.12: Average F1-measure on Dataset 2.

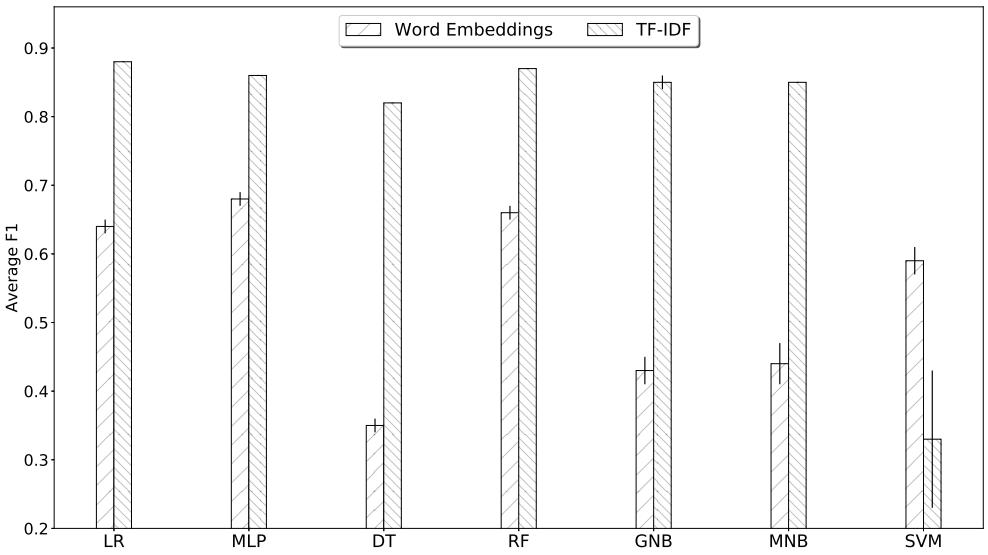

FIGURE 4.13: Average F1-measure on Dataset 3.

Chapter 5

Web-Based Application

We have also implemented a Web-based application in order to make our implemented models publicly available. Our application is able to extract the heading and the body text of a news article, and then classifies it as real or fake news based on our models. The application is available at `http://fenaki.e-ce.uth.gr`.[1] In the entry page of the application the user can provide a URL in the textbox, and then click on the button to run the algorithms.

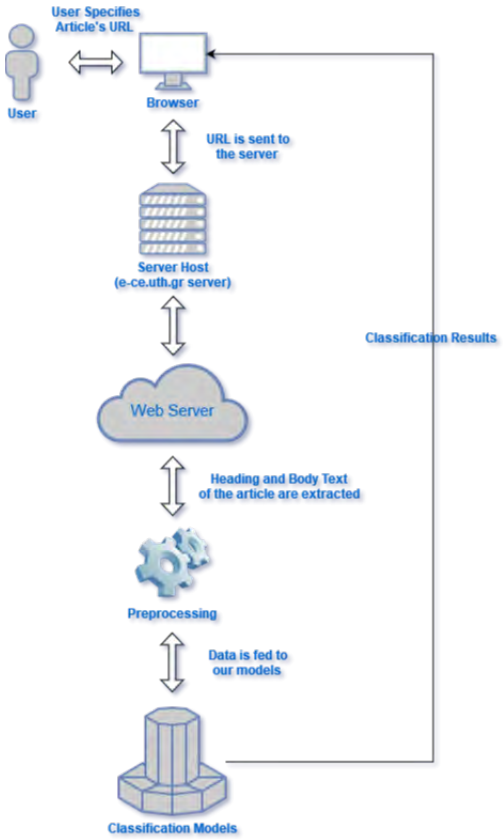

FIGURE 5.1: Fenaki execution flowchart

[1]The greek word *φενακη* (spelled 'fenaki' with latin characters) describes a lie that is being told with the purpose to deceive or to mislead, i.e., the fraudulent lie.

The home page is shown in Figure 5.2

FIGURE 5.2: Fenaki Home-Page

Following the home-page, the results are shown in a page such as the one shown in Figures 5.3-5.5.

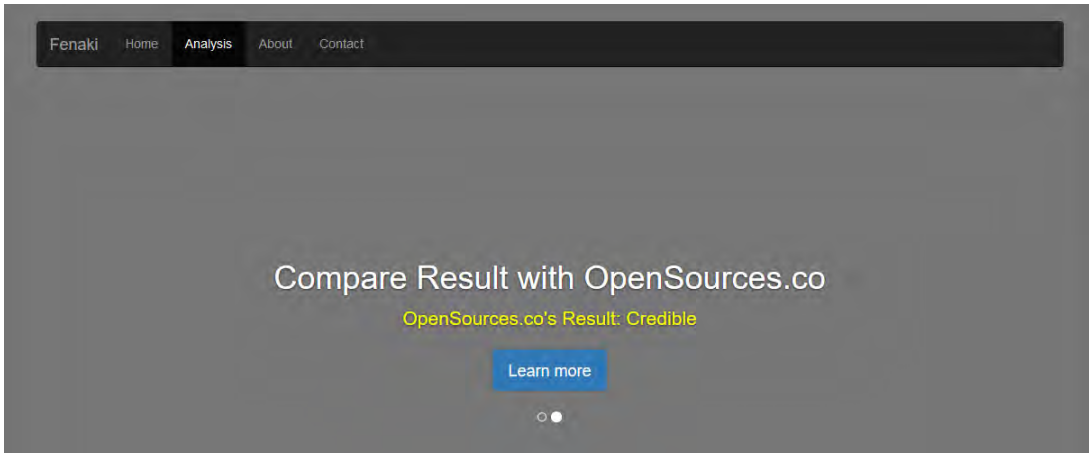

FIGURE 5.3: Domain evaluation based on its credibility from OpenSources.co

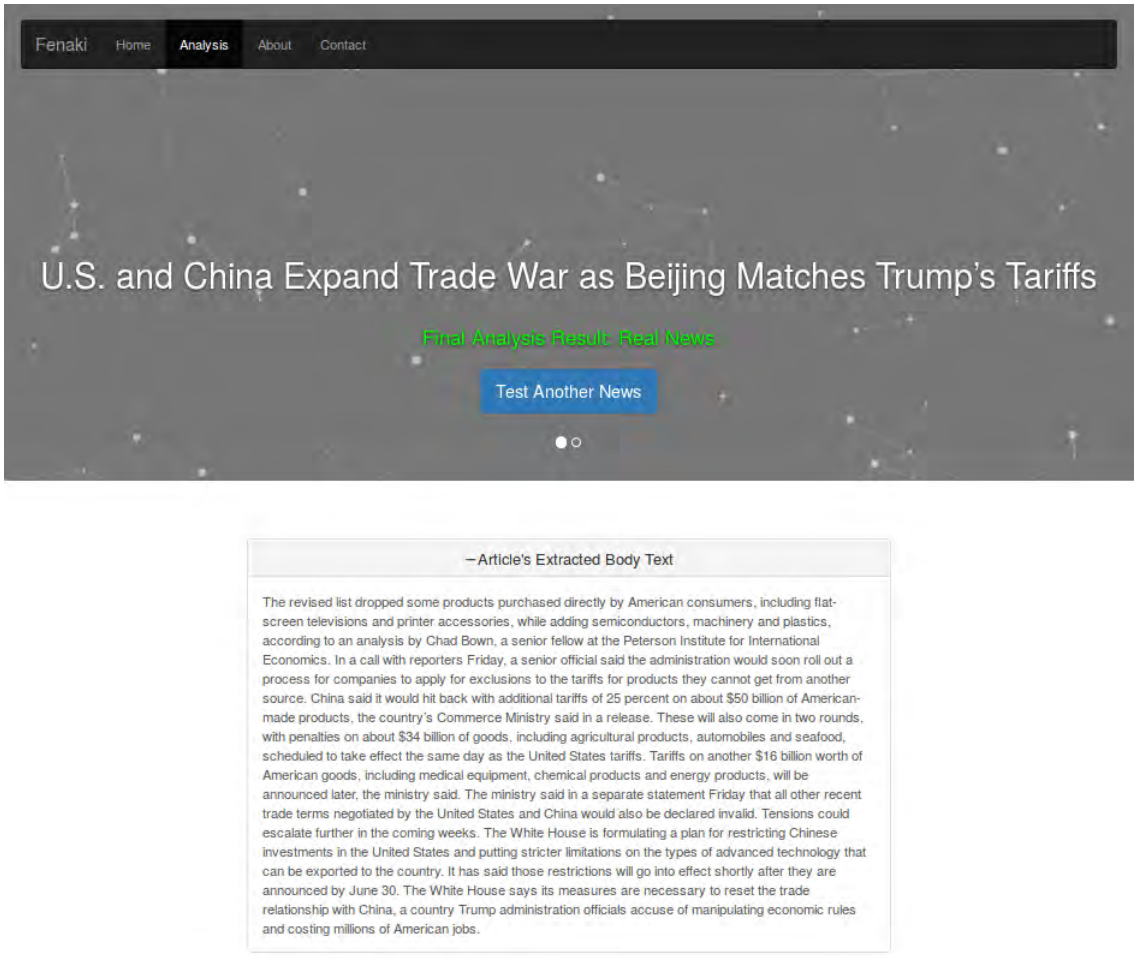

FIGURE 5.4: Results page for the query with the following input: https://www.nytimes.com/2018/06/15/us/politics/us-china-tariffs-trade.html which is classified as "Real news".

TF-IDF Results

Algorithm	Headline Evaluation (Using Dataset 1 for training)	Headline Evaluation (Using Dataset 2 for training)	Body-Text Evaluation (Using Dataset 3 for training)
LR	Fake News	Fake News	Real News
MLP	Fake News	Fake News	Real News
DT	Fake News	Fake News	Fake News
RF	Fake News	Fake News	Fake News
GNB	Fake News	Fake News	Real News
MNB	Real News	Fake News	Real News

Word-Embeddings Results

Algorithm	Headline Evaluation (Using Dataset 1 for training)	Headline Evaluation (Using Dataset 2 for training)	Body-Text Evaluation (Using Dataset 3 for training)
LR	Real News	Fake News	Real News
MLP	Real News	Fake News	Real News
DT	Real News	Real News	Real News
RF	Real News	Real News	Real News
GNB	Fake News	Real News	Real News
MNB	Real News	Real News	Real News

FIGURE 5.5: Classification results of all the different versions of our models

Chapter 6

Conclusions

6.1 Summary

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.

The fast spreading of fake news and the impact they are having on our society, along with the inscalability of manually detecting them, have created a surge of research and development in machine learning algorithms to battle them. In this thesis, 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.

6.2 Future Work

The findings of this thesis reinforce the fact that fake news detection is a difficult task to achieve and needs more research. Probably, the biggest problem is the lack of a reliable dataset that contains news articles and their corresponding label of being real or fake news. The Datasets that we used, although LIAR Dataset is the first of its kind with such a large magnitude, its main focus is politics and statements. Regarding the other two Datasets, although the Dataset from Signal Media includes a huge amount of real news articles, only some of them could be used because of the Dataset from Kaggle which is 1:100 in size compared to the one from Signal Media.

The work of this thesis showed that it is possible to create an unbiased system capable of classifying fake, or maliciously written, news by using only the articles' content (mainly headline and bodytext), without any meta-data from the original source. Of course, more work still needs to be done to improve the performance of the techniques used in this study.

In the future, a proper Dataset of a large magnitude might enable more space for research with diverse data. Along with that, it might be useful to combine NLP-techniques with information retrieval and reinforcement learning to see if those hybrid-approaches improve the distinction.

Moreover, the Web Application we built could be further developed in order to be made available as an extension for any browser but more importantly in order to see some improvements the system must be fair, robust, and always learning.

Bibliography

[1] L. Breiman, J.H. Friedman, R.A. Olshen, C.J. Stone, *Classification and Regression Trees*, Brooks/Cole Publishing, 1984.

[2] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[3] C. Cortes and V. Vapnik, "Support-vector network," *Machine Learning*, vol. 20, pp. 273–297, 1995.

[4] G.B. Guacho, S. Abdali, N. Shah and E. Papalexakis, "Semi-supervised content-based detection of misinformation via tensor embeddings," Technical Report. Available at: https://arxiv.org/abs/1804.09088, 2018.

[5] A. Gupta, H. Lamba, P. Kumaraguru and A. Joshi, "Faking Sandy: Characterizing and identifying fake images on Twitter during Hurricane Sandy," *Proceedings of the ACM International Conference on World Wide Web (WWW)*, pp. 729–736, 2010.

[6] M. Gupta, P. Zhao and J. Han, "Evaluating event credibility on Twitter," *Proceedings of the SIAM International Conference on Data Mining (SDM)*, pp. 153–164, 2012.

[7] A. Gupta and P. Kumaraguru, "Credibility ranking of tweets during high impact events," *Proceedings of the Workshop on Privacy and Security in Online Social Media*, 2012.

[8] M. Hardalov, I. Koychev and P. Nakov, "In search of credible news," *Proceedings of the Artificial Intelligence: Methodology, Systems and Applications*, pp. 172–180, 2016.

[9] B.D. Horne and S. Adali, "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

[10] S. Hosseinimotlagh, E. Papalexakis, "Unsupervised content-based identification of fake news articles with tensor decomposition ensembles," *Proceedings of the Workshop on Misinformation and Misbehavior Mining on the Web (MIS2)*, 2018.

[11] Y. Kim, "Convolutional neural networks for sentence classification," *Proceedings of the Empirical Methods in Natural Language Processing (EMNLP)*, 2014.

[12] X. Liu, B. Zhang, A. Susarla, R. Padman, “Go to YouTube and See me tomorrow: The role of social media in managing chronic conditions." Available at https://ssrn.com/abstract=3061149.

[13] C.D. Manning, P. Ragnavan and H. Schütze, *Introduction to Information Retrieval*, Cambridge University Press, 2008.

[14] G. Pennycook, D.G. Rand, “Assessing the effect of “disputed" warnings and source salience on perceptions of fake news accuracy," Working paper, Yale University, September, 2017.

[15] C. Shao, P.-M. Hui, L. Wang, X. Jiang, A. Flammini, F. Menczer, G.L. Ciampaglia, “Anatomy of an online misinformation network," *PLOS One*, vol. 13, no. 4, e0196087, 2018.

[16] K. Shu, A. Sliva, S. Wang, J. Tang, H. Liu, “Fake news detection on social media: A data mining perspective," *ACM SIGKDD Explorations*, vol. 19, iss. 1, pp. 22–36, 2017.

[17] J. Pennington, R. Socher, and C.D. Manning, “Glove: Global vectors for word representation," *Proceedings of the Empirical Methods in Natural Language Processing (EMNLP)*, 2014.

[18] M.F. Porter, “An algorithm for suffix stripping", *Program*, vol. 14, no. 3, pp. 130–137, 1980.

[19] V. Vapnik, *Statistical Learning Theory*, Wiley, 1998.

[20] S. Vosoughi, D. Roy, S. Aral, “The spread of true and false news online," *Science*, vol. 359, iss. 6380, pp. 1146–1151, 2018.

[21] W.Y. Wang, ““Liar, Liar Pants on Fire": A New Benchmark Dataset for Fake News Detection," *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pp. 422–426, 2017.

[22] S. Gilda, "Evaluating machine learning algorithms for fake news detection", *IEEE 15th Student Conference on Research and Development (SCOReD)*, 2017.

[23] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuglu, P. Kuksa. "Natural Language Processing (Almost) from Scratch",*Journal of Machine Learning Research*, 2011.

[24] G. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, R. Salakhutdinov. "Improving neural networks by preventing co-adaptation of feature detectors", *CoRR*, 2012.