# Some experiments on Deep Learning for Fake News Detection (DISCUSSION PAPER)

Angelo Chianese, Elio Masciari, Vincenzo Moscato, Antonio Picariello, and Giancarlo Sperli

University Federico II of Naples, Italy
{name.surname}@unina.it

**Abstract.** The uncontrolled growth of fake news creation and dissemination we observed in recent years causes continuous threats to democracy, justice, and public trust. This problem has significantly driven the effort of both academia and industries for developing more accurate fake news detection strategies. Early detection of fake news is crucial, however the availability of information about news propagation is limited. Moreover, it has been shown that people tend to believe more fake news due to their features [13]. In this paper, we present our framework for fake news detection and we discuss in detail a solution based on deep learning methodologies we implemented by leveraging Google Bert features. Our experiments conducted on two well-known and widely used real-world datasets suggest that our method can outperform the state-of-the-art approaches and allows fake news accurate detection, even in the case of limited content information.

## 1 Introduction

Social media are nowadays the main medium for large-scale information sharing and communication and they can be considered the main drivers of the Big Data revolution we observed in recent years[1]. Unfortunately, due to malicious user having fraudulent goals fake news on social media are growing quickly both in volume and their potential influence thus leading to very negative social effects. In this respect, identifying and moderating fake news is a quite challenging problem[14]. Indeed, fighting fake news in order to stem their extremely negative effects on individuals and society is crucial in many real life scenarios. Therefore, fake news detection on social media has recently become an hot research topic both for academia and industry.

Fake news detection dates back long time ago[15], for a very long time journalist and scientists fought against misinformation, however, the pervasive use of internet for communication allows for a quicker spread of false information.

Copyright © 2020 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0). This volume is published and copyrighted by its editors. SEBD 2020, June 21-24, 2020, Villasimius, Italy.

CEUR-WS.org/Vol-2646/33-paper.pdf

Indeed, the term *fake news* has grown in popularity in recent years, especially after the 2016 United States elections but there is still no standard definition of fake news [11]. Aside the definition that can be found in literature, one of the most well accepted definition of fake news is the following: *Fake news is a news article that is intentionally and verifiable false and could mislead readers* [2]. There are two key features of this definition: authenticity and intent. First, fake news includes false information that can be verified as such[8]. Second, fake news is created with dishonest intention to mislead consumers[11].

The content of fake news exhibits heterogeneous topics, styles and media platforms, it aims to mystify truth by diverse linguistic styles while insulting true news. Fake news are generally related to newly emerging, time-critical events, which may not have been properly verified by existing knowledge bases due to the lack of confirmed evidence or claims. Thus, fake news detection on social media poses peculiar challenges due to the inherent nature of social networks that requires both the analysis of their content [10, 6] and their social context[12, 3].

**Our approach in a nutshell.** Fake news detection problem can be formalized as a classification task thus requiring features extraction and model construction. The detection phase is a crucial task as it is devoted to guarantee users to receive authentic information. We will focus on finding clues from news contents. Our goal is to improve the existing approaches defined so far when fake news is intentionally written to mislead users by mimicking true news. More in detail, traditional approaches are based on verification by human editors and expert journalists but do not scale to the volume of news content that is generated in online social networks. As a matter of fact, the huge amount of data to be analyzed calls for the development of new computational techniques. It is worth noticing that, such computational techniques, even if the news is detected as fake, require some sort of expert verification before being blocked. In our framework, we perform an accurate pre-processing of news data and then we apply three different approaches. The first approach is based on classical classification approaches. We also implemented a deep learning approach that leverages neural network features for fake news detection. Finally, for the sake of completeness we implemented some multimedia approaches in order to take into account misleading images. Due to space limitation, we discuss in this paper the deep learning approach.

## 2 Our Fake News Detection Framework

Our framework is based on news flow processing and data management in a pre-processing block which performs filtering and aggregation operation over the news content. Moreover, filtered data are processed by two independent blocks: the first one performs natural language processing over data while the second one performs a multimedia analysis. The overall process we execute for fake news detection is depicted in Figure 1. In the following, we describe each module in more detail:

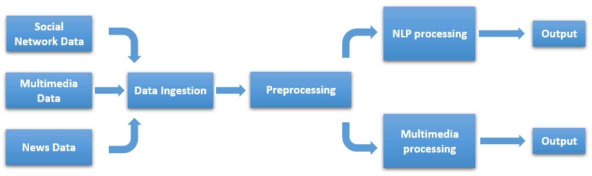


Fig. 1: The overall process at a glance

**Data Ingestion Module.** This module take care of data collection tasks. Data can be highly heterogeneous: social network data, multimedia data and news data. We collect the news text and eventual related contents and images.

**Pre-processing Module.** This component is devoted to the acquisition of the incoming data flow. It performs filtering, data aggregation, data cleaning and enrichment operations.

**NLP Processing Module.** It performs the crucial task of generating a binary classification of the news articles, i.e., whether they are fake or reliable news. It is split in two submodules. The *Machine Learning* module performs classification using an ad-hoc implemented Logistic Regression algorithm after an extensive process of feature extraction and selection TF-IDF based in order to reduce the number of extracted features. The *Deep Learning* module classifies data using Google Bert algorithm after a tuning phase on the vocabulary. It also performs a binary transformation and eventual text padding in order to better analyze the input data.

**Multimedia Processing Module.** This module is tailored for Fake Image Classification through Deep Learning algorithms, using ELA (Error Level Analysis) and CNN.

Due to space limitation, we discuss in the following only the details of the deep learning module and the obtained results.

## 2.1 The Deep Learning Module

The Deep Learning Module computes a binary classification on a text datasets of news that will be labelled as 0 if a news is marked as Real, and as 1 if it is marked as Fake. The Deep Learning Module classifies news content using a new language model called *B.E.R.T.* (Bidirectional Encoder Representations from Transformers) developed and released by Google. Prior to describing the algorithm features in detail, we briefly describe the auxiliary tools being used, while in Section 3 we describe the experimental evaluation that lead to our choice on BERT.

**Colaboratory.** Colab is intended for machine learning education and research, it requires no setup and runs entirely on the cloud. By using Colab it's possible to write and execute code, save and share analytics and it provides

https://research.google.com/colaboratory access to expensive and powerful computing resources for free by a web interface. More in detail, Colab's hardware is powered by: Intel(R) Xeon(R) CPU @ 2.00GHz, nVidia T4 16 GB GDDR6 @ 300 GB/sec, 15GB RAM and 350GB storage.

**Tensor Flow.** It is devoted to train and run neural networks for image recognition, word embeddings, recurrent neural networks, and natural language processing. It is a cross-platform tool and runs on CPUs, GPUs, even on mobile and embedded platforms. TensorFlow uses dataflow graphs to represent the computation flow, i.e., these structures describe the data flow through the processing nodes. Each node in the graph represents a mathematical operation, and each connection between nodes is a multidimensional data array called tensor. The TensorFlow Distributed Execution Engine abstracts from the supported devices and provides a high performance-core implemented in C++ for the TensorFlow platform. On top there are Python and C++ frontends. The Layers API provides a simple interface for most of the layers used in deep learning models. Finally, higher-level APIs, including Keras, makes training and evaluating distributed models easier.

**Keras.** It is a high-level neural network API, implemented in Python and capable of running on top of TensorFlow. It allows for easy and fast prototyping through: 1) User Friendliness as it offers consistent and simple APIs that minimizes the number of user actions required for common use cases; 2) Modularity as neural layers, cost functions, optimizers, initialization schemes, activation functions and regularization schemes are all standalone modules that can be combined to create new models; 3) Extensibility as new modules are simple to add as new classes and functions.

**Google BERT.** This tool has been developed in order to allow an easier implementation of two crucial tasks for Natural Language Processing (NLP): Transfer Learning through unsupervised pre-training and Transformer architecture. The idea behind Transfer Learning is to train a model in a given domain on a large text corpus, and then leverage the gathered knowledge to improve the model's performance in a different domain. In this respect, BERT has been pre-trained on Wikipedia and BooksCorpus. On the opposite side, the Transformer architecture processes all elements simultaneously by linking individual elements through a process known as attention. This mechanism allows a deep parallelization and guarantee higher accuracy across a wide range of tasks. BERT outperforms previous proposed approaches as it is the first unsupervised, fully bidirectional system for NLP pre-training. Pre-trained representations can be either context-free or context based dependig on user needs. Due to space limitations we do not describe in detail the BERT's architecture and the encoder mechanism.

## 3 Our Benchmark

In this section we will describe the fake news detection process for the deep learing module and the datasets we used as a benchmark for our algorithms.

### 3.1 Dataset Description

**Liar Dataset.** This dataset includes 12.8K human labelled short statements from fact-checking website Politifact.com. Each statement is evaluated by a Politifact.com editor for its truthfulness. The dataset has six fine-grained labels: pants-fire, false, barely-true, half-true, mostly-true, and true. The distribution of labels is relatively well- balanced. [12] For our purposes the six fine-grained labels of the dataset have been collapsed in a binary classification, i.e., label 1 for fake news and label 0 for reliable ones. This choice has been made due to binary Fake News Dataset feature. The dataset is partitioned into three files: 1) Training Set: 5770 real news and 4497 fake news; 2) Test Set: 1382 real news and 1169 fake news; 3) Validation Set: 1382 real news and 1169 fake news. The three subsets are well balanced so there is no need to perform oversampling or undersampling.

The processed dataset has been uploaded in Google Drive and, then, loaded in Colab's Jupyter as a Pandas Dataframe. It has been added a new column with the number of words for each row article. Using the command $df.describe()$ on this column it is possible to print the following statistical information: count 15389.000000, mean 17.962311, std 8.569879, min 1.000000, 25% 12.000000, 50% 17.000000, 75% 22.000000, max 66.000000. These statistics show that there are articles with only one word in the dataset, so it has been decided to remove all rows with less than 10 words as they are considered poorly informative. The resulting dataset contains 1657 less rows than the original one. The updated statistics are reported in what follows: count 13732.000000, mean 19.228663, std 8.192268, min 10.000000, 25% 14.000000, 50% 18.000000, 75% 23.000000, max 66.000000. Finally, the average number of words per article is 19.

**FakeNewsNet.** This dataset has been built by gathering information from two fact-checking websites to obtain news contents for fake news and real news such as PolitiFact and GossipCop. In PolitiFact, journalists and domain experts review the political news and provide fact-checking evaluation results to claim news articles as fake or real. Instead, in GossipCop, entertainment stories, from various media outlets, are evaluated by a rating score on the scale of 0 to 10 as the degree from fake to real. The dataset contains about 900 political news and 20k gossip news and has only two labels: true and false. [14]

This dataset is publicly available by the functions provided by the FakeNewsNet team and the Twitter API. As mentioned above, FakeNewsNet can be split in two subsets: GossipCop and Politifact.com. We decided to analyse only political news as they produce worse consequences in real world than gossip ones. The dataset is well balanced and contains 434 real news and 367 fake news. Most of the news regards the US as it has already been noticed in LIAR. Fake news topics concern Obama, police, Clinton and Trump while real news topics refer to Trump, Republicans and Obama. Such as the LIAR dataset, it has been added a new column and used the command $df.describe()$ to print out the following statistical information: count 801, mean 1459.217228, std 3141.157565, min 3, 25% 114, 50% 351, 75% 893, max 17377. The average number of words per articles in Politifact dataset is 1459, which is far longer than the average sentence length in Liar Dataset that is 19 words per articles. Such a statistics suggested us to compare the model performances on datasets with such different features.

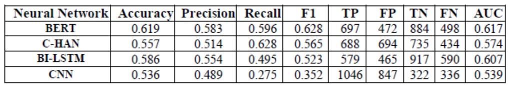
### 3.2 Pre-elaboration steps

The above mentioned datasets are available in CSV format and are composed of two columns: text and label. The news text need to be pre-processed for our analysis. In this respect, an ad-hoc Python function has been developed for unnecessary IP and URL addresses removal, HTML tags checking and words spell-check. Due to neural features, we decide to maintain some stop words in order to allow a proper context analysis. Thus, to ameliorate the noise problem, we created a custom list of stop words. We leverage Keras Tokenizer for preparing text documents for subsequent deep learning steps. More in detail, we create a vocabulary index based on word frequency, e.g., given the sentence *The cat sat on the mat* we create the following dictionary *word_index*[the] = 1, *word_index*[cat] = 2 so every word gets a unique integer value; the 0 value is reserved for padding. Lower integer means more frequent word. After this encoding step, we obtain for each text a sequence of integers. As BERT needs a more elaborated input than other neural networks wed need to produce a *tsv* file, with four columns, and no header. The columns to be added to dataset are: 1) *guid*, i.e., a row ID; 2)*label*, i.e., the label for the row (it should be an int); 3) *alpha*, a dummy column containing the same letter for all rows, it is not used for classification but it is needed for proper running of the algorithm and 4) *text*, i.e., the news content. The data needs to be converted in InputFeature object to be compatible with Transformer Architecture. The conversion process includes tokenization and converting all sentences to a given sequence length (truncating longer sequences, and padding shorter sequences). Tokenization is performed using WordPiece tokenization, where the vocabulary is initialized with all the individual characters in the language, and then the most frequent/likely combinations of the existing words in the vocabulary are iteratively added. Words that does not occur in the vocabulary are broken down into sub-words in order to search for possible matches in the collection.

## 4 Evaluation

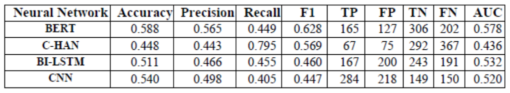
In order to show that the Google BERT model we implemented outperforms the results of the current performance state of art both on Liar dataset and Polifact dataset, we report in Figure 2 and 3 the best results obtained for the other approaches commonly used in literature for those datasets. In particular, neural betworks such as CNN, BI-LSTM and C-HAN were initialized with 300-dimensional pre-trained embeddings from GloVe [9], trained on a dataset of one billion tokens (words) with a vocabulary of 400 thousand words.

We compared the performances on well-established evaluation measure like: Accuracy, Precision, Recall, F1 measure, Area Under Curve (AUC) [5] and the



| Neural Network | Accuracy | Precision | Recall | F1 | TP | FP | TN | FN | AUC |
|---|---|---|---|---|---|---|---|---|---|
| BERT | 0.619 | 0.583 | 0.596 | 0.628 | 697 | 472 | 884 | 498 | 0.617 |
| C-HAN | 0.557 | 0.514 | 0.628 | 0.565 | 688 | 694 | 735 | 434 | 0.574 |
| BI-LSTM | 0.586 | 0.554 | 0.495 | 0.523 | 579 | 465 | 917 | 590 | 0.607 |
| CNN | 0.536 | 0.489 | 0.275 | 0.352 | 1046 | 847 | 322 | 336 | 0.539 |

Fig. 2: Comparison for Google BERT against state of the art approaches on LIAR datataset

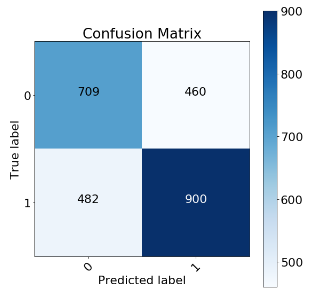


| Neural Network | Accuracy | Precision | Recall | F1 | TP | FP | TN | FN | AUC |
|---|---|---|---|---|---|---|---|---|---|
| BERT | 0.588 | 0.565 | 0.449 | 0.628 | 165 | 127 | 306 | 202 | 0.578 |
| C-HAN | 0.448 | 0.443 | 0.795 | 0.569 | 67 | 75 | 292 | 367 | 0.436 |
| BI-LSTM | 0.511 | 0.466 | 0.455 | 0.460 | 167 | 200 | 243 | 191 | 0.532 |
| CNN | 0.540 | 0.498 | 0.405 | 0.447 | 284 | 218 | 149 | 150 | 0.520 |

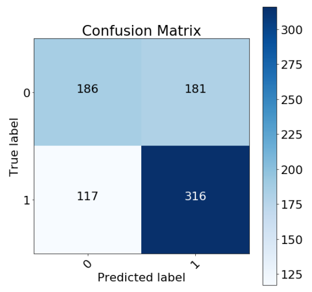
Fig. 3: Comparison for Google BERT against state of the art approaches on Polifact datataset

values reported in the obtained confusion matrices for each algorithm, i.e., True Positive (TP), False Positive (FP), True Negative (TN) and False Negative (FN).

We hypothesize that our results are quite better due to a fine hyper parameter tuning we performed, a better pre-processing step and the proper transformation.



Confusion Matrix for LIAR dataset



Confusion Matrix for Polifact dataset

For the sake of completeness, we report in Figure 4a and Figure 4b the detailed confusion matrices obtained for LIAR and Polifact datasets.

## 5 Conclusion and Future Work

In this paper, we investigated the problem of fake news detection by deep learning algorithms. We developed a framework the leverage Google BERT for analyzing real-life datasets and the results we obtained are quite encouraging. As for future work, we would like to extend our analysis by considering also user profiles' features, some kind of dynamic analysis of news diffusion mechanism[14] and geometric features[8] in our fake news detection model[7].

## References

1. D. Agrawal et al. Challenges and opportunities with big data. A community white paper developed by leading researchers across the United States. Technical report, Purdue University, Mar 2012.
2. Hunt Allcott and Matthew Gentzkow. Social media and fake news in the 2016 election. Working Paper 23089, National Bureau of Economic Research, January 2017.
3. Nunziato Cassavia, Elio Masciari, Chiara Pulice, and Domenico Saccà. Discovering user behavioral features to enhance information search on big data. *TiiS*, 7(2):7:1–7:33, 2017.
4. J. Shane Culpepper, Alistair Moffat, Paul N. Bennett, and Kristina Lerman, editors. *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining, WSDM 2019, Melbourne, VIC, Australia, February 11-15, 2019*. ACM, 2019.
5. Peter A. Flach and Meelis Kull. Precision-recall-gain curves: PR analysis done right. In Corinna Cortes, Neil D. Lawrence, Daniel D. Lee, Masashi Sugiyama, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pages 838–846, 2015.
6. Chuan Guo, Juan Cao, Xueyao Zhang, Kai Shu, and Miao Yu. Exploiting emotions for fake news detection on social media. *CoRR*, abs/1903.01728, 2019.
7. Elio Masciari. SMART: stream monitoring enterprise activities by RFID tags. *Inf. Sci.*, 195:25–44, 2012.
8. Federico Monti, Fabrizio Frasca, Davide Eynard, Damon Mannion, and Michael M Bronstein. Fake news detection on social media using geometric deep learning. *arXiv preprint arXiv:1902.06673*, 2019.
9. Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.
10. Martin Potthast, Johannes Kiesel, Kevin Reinartz, Janek Bevendorff, and Benno Stein. A stylometric inquiry into hyperpartisan and fake news. *CoRR*, abs/1702.05638, 2017.
11. Kai Shu, Amy Sliva, Suhang Wang, Jiliang Tang, and Huan Liu. Fake news detection on social media: A data mining perspective. *CoRR*, abs/1708.01967, 2017.
12. Kai Shu, Suhang Wang, and Huan Liu. Beyond news contents: The role of social context for fake news detection. In Culpepper et al. [4], pages 312–320.
13. Soroush Vosoughi, Deb Roy, and Sinan Aral. The spread of true and false news online. *Science*, 359(6380):1146–1151, 2018.
14. Shuo Yang, Kai Shu, Suhang Wang, Renjie Gu, Fan Wu, and Huan Liu. Unsupervised fake news detection on social media: A generative approach. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 5644–5651, 2019.
15. Xinyi Zhou, Reza Zafarani, Kai Shu, and Huan Liu. Fake news: Fundamental theories, detection strategies and challenges. In Culpepper et al. [4], pages 836–837.