Università di Pisa

Fake News Detection: Four Neural Networks challenge the State of the Art

Annachiara Aiello[‡] and Veronica Pistolesi[‡]

Abstract

The purpose of this work is to make a contribution to the challenge of Fake News Detection: the task of correctly classifying news as fake or real. In particular, this study compares four different neural networks trained on the pre-processed Fake News Dataset [1] with the state of the art. All the architectures are implemented with a pre-trained word embedding method, namely GloVe or fastText. The proposed models are a Convolutional Neural Network, two types of gated Recurrent Neural Networks - Bidirectional LSTM and GRU - and a combination of Convolutional Neural Network with Bidirectional LSTM. The first experiments show that Bidirectional LSTM with fastText word embedding is the model with the best results. Finally, the best model is put into a Transformer architecture, with BERT (bert-base-uncased) used as embedding layer, further exceeding previous performances.

1 Introduction

Fake news is misleading information passed off as true, with malicious intent [2]. Although this kind of information has always existed, its spread massively increased with the recent rise of social networks. Studies state that factors implicated in the diffusion of fake news are political polarization, post-truth politics, motivated reasoning, confirmation bias, and social media algorithms [3]. One of the most dangerous effects of fake news is that they can reduce impact of real ones, undermining people trust in serious media coverage.

Our study aims to make a contribution to the task of binary classification of news, starting from the work of I.K. Sastrawan, I.P.A. Bayupati and D.M.S. Arsa [4], on the Fake News Dataset [1]. Specifically, we implemented our own Convolutional Neural Network (CNN) and Bidirectional LSTM (Bi-LSTM), with the addition of Gated Recurrent Units (GRU) and a combination of CNN with Bi-LSTM (CNN+Bi-LSTM). One of the purposes of the project was also to see the effect of the two pre-trained word embedding models, namely GloVe [5] and fastTex [6]. Furthermore, the best performing model was put on top of a BERT encoder stack, since Transformers currently produce the most interesting results in many NLP tasks. The paper is organised into several main sections. Section 2 explains in details the connection with related works. Section 3 is further divided into three subsections: the first one and the second one describe the dataset and the word embedding methods used, respectively; the third one presents the proposed deep learning models. Section 4 illustrates the experiment setup, Section 5 shows all the results and discussions, and eventually Section 6 contains the conclusions of the work.

2 Related Works

So far, several methods have been explored to detect fake news. Our contribution to the research is to compare the performance of four neural networks implemented with methods developed in similar studies using the same dataset, namely Fake News Dataset [1].

Among the various experiments, Kaliyar et al. [7] chose a pre-trained word embedding called GloVe, which was later combined with a CNN. As a result, the accuracy, precision, recall, and F1-score were 98.36%, 99.40%, 96.88%, and 98.12%, respectively.

Sastrawan et al. [4] compared the effect of some pre-trained word embeddings, namely Word2Vec [8], GloVe, and fastText, with three deep learning methods (CNN, Bi-LSTM and ResNet). They obtained 98.65%, 98.64%, 98.66%, and 98.65% results for accuracy, precision, recall, and F1-score,

[‡] These authors are students of the master degree in Computer Science, Artificial Intelligence curriculum, University of Pisa and they contributed equally to this work.

respectively, using fastText with a Bi-LSTM.

In our work, we trained our CNN and Bi-LSTM architectures, with GloVe and fastText word embedding respectively, to compare our results with those of Kaliyar et al. [7] and of Sastrawan et al. [4] In addition, we implemented and tested two new models, that are GRU and a combination of CNN with Bi-LSTM, both with GloVe and fasText methods. After that, a new experiment was made by combining the best performing model with a BERT encoder. The results are shown in Section 5.

3 Research Method

The first part of this section describes the data pre-processing and the analysis; the second one introduces the pre-trained language representation methods; the last one illustrates the architectures of the implemented models.

3.1 Dataset

The dataset used to train and test our proposed neural networks is the Fake News Dataset [1], which was also used in previous studies (Section 2). The dataset consists of three files, namely `train.csv`, `test.csv` and `submit.csv`.

The first file (`train.csv`) is a full training dataset with the following attributes:

- id: unique id for a news article;
- title: the title of a news article;
- author: author of the news article;
- text: the text of the article; could be incomplete;
- label: a label that marks the article as potentially unreliable (1 = unreliable, 0 = reliable).

The second file (`test.csv`) is a testing dataset with all the same attributes as `train.csv` without the labels. The third file (`submit.csv`) contains the labels of `test.csv`. In particular, we chose to use a version of this dataset that had already been cleaned, augmented and pre-processed, provided by the best performing research team [4] of previous studies.

In the data cleansing process, data with no content or label were deleted to remove low-quality data from the dataset. Data augmentation process was performed applying the back-translation English-German [9] to balance the dataset. Finally, data went through pre-processing where tokenization, case folding, characters non alphabet and stopword removal and lemmatization were carried out.

At the end of these steps, the current dataset consists of two files, namely `train.csv` and `test.csv`, in which the *id* column has been dropped and the columns named *title*, *author* and *text* have been merged into a single column named *text*. Furthermore, in the case of `test.csv`, the *label* column from the `submit.csv` file was added.

Our study begins with the data analysis of the current dataset, where we considered the `test.csv` file as a black box and chose to focus our analysis on the `train.csv` file. Table 1 shows the examined values.

Exploring the data distribution is useful since neural networks need to have the inputs with similar shape and size. Furthermore, this work uses texts as input for the models and not all the sequences have the same length in terms of tokens. This is why padding the final configurations of the inputs became a necessary step. Figure 1 shows the Boxplot distribution of the news length. The standard deviation of the news length showed too much dispersion in the data. Therefore, the median length of the news (dotted black line in Figure 2), which is less sensitive to outliers (3.37%), turned out to be the most representative value for our input data. For these reasons, we chose the latter as the maximum length of the padded sequences instead of the simple mean.

train.csv	
number of unique words	123659
number of news	16646
max length news	10211
min length news	2
mean length news	426
standard deviation news	443.92
median length news	317
number of outliers	561
percentage of outliers	3.37%

Table 1 Examined values in `train.csv` file

Final configurations of the sets, used during the model selection phase, were obtained by splitting the `train.csv` file: 80% of the data was used as training set and the remaining 20% as validation set (data in `test.csv` were clearly used as test set). Table 2 shows the composition of the main sets.

Data	Fake News	Real News	Total
`train.csv`	8323	8323	16646
Training Set	6658	6658	13316
Validation Set	1665	1665	3330

Table 2 Training set, validation set and full dataset composition.

3.2 Language Representation

When words with similar meanings are visualized, they will appear next to one another thanks to a technique called word embedding or distributed word representa-

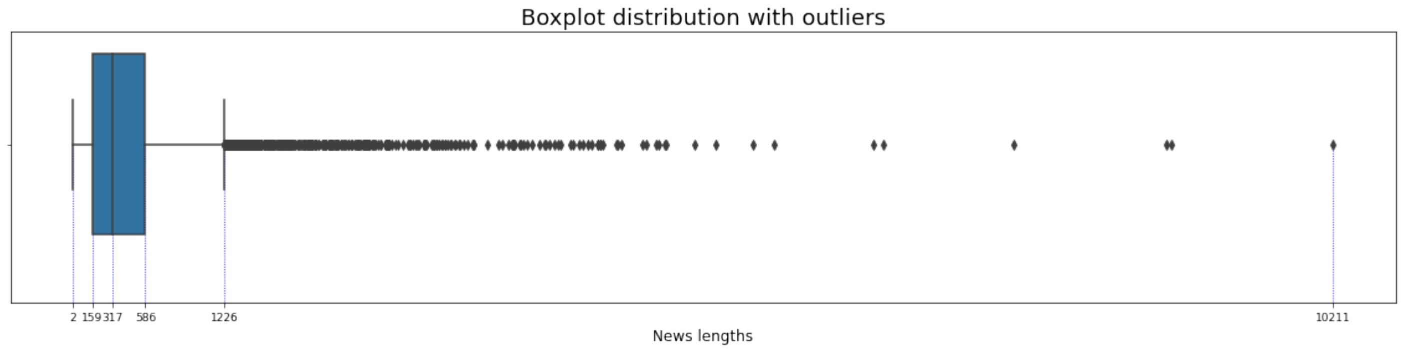


Fig. 1 Boxplot Distribution with outliers.

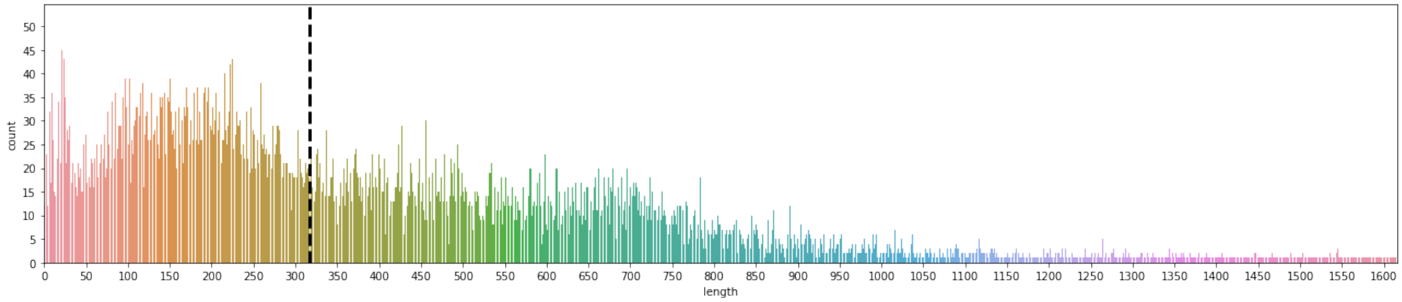


Fig. 2 Countplot Distribution of the news length.

tion[10]. This is possible due to word embedding's ability to capture a word's semantic and syntactic details in a sizable corpus[11]. Because it produces promising results, word embedding is being utilized more frequently in text analysis-based research on sentiment analysis, entity recognition, part of speech tagging, and other topics[12]. GloVe and fastText are two popular context free word embedding examples, while BERT is a contextual representation. Context-free models generate a single word embedding representation for each word in the vocabulary, where BERT takes into account the context for each occurrence of a given word.

3.2.1 GloVe

Global Vector (GloVe)[5] is a widely used pre-trained word embedding. The idea behind is that it creates a vector representation of a word factorizing the matrix of the number of word occurrences in a corpus[13].

3.2.2 FastTesxt

fastText[6] is a pre-trained word embedding that makes use of the upgraded CBOW Word2Vec architecture. Word order and the use of more effective computational procedures contributed to the improvement[14]. Since the bag of character n-gram has a positive effect on vectors representing words that are uncommon or misspelled, fastText also includes subword information[15].

3.2.3 BERT

BERT (Bidirectional Encoder Representations from Transformers) is the first deep bidirectional language representation, pre-trained from unlabeled text (BookCorpus[16] and English Wikipedia) by jointly conditioning on both left and right context in all layers. It improves the so-called fine-tuning approach, which introduces minimal task-specific parameters, and is trained on the downstream tasks by simply fine-tuning all pre-trained parameters. As a result, the pre-trained BERT model can be finetuned with just one additional output layer to create state-of-the-art models for a wide range of tasks. In particular, BERT uses masked language model to randomly masks some of the tokens from the input, and the objective is to predict the original vocabulary id of the masked word based only on its context[17].

3.3 Models

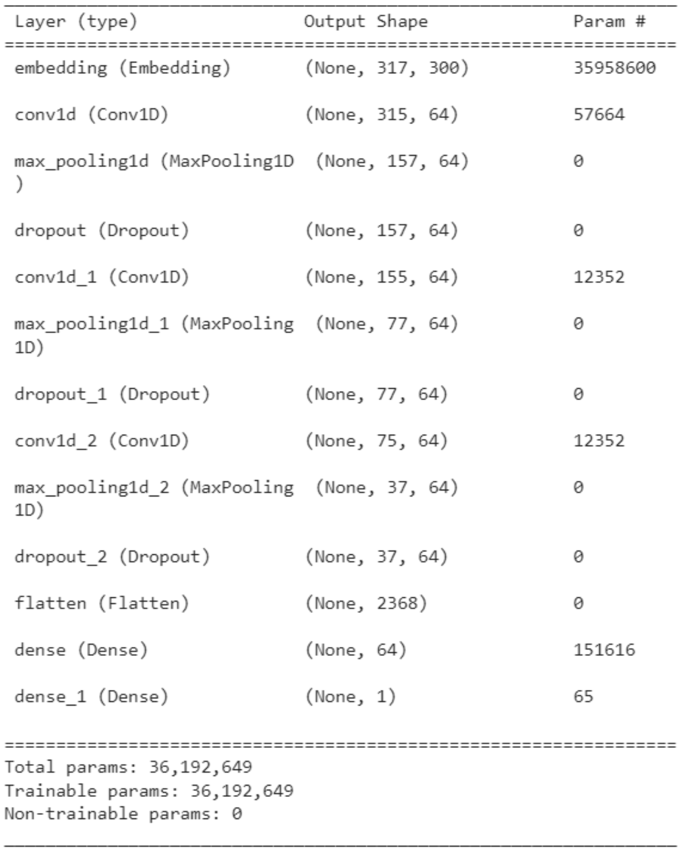
As already mentioned, our study tested four different deep learning methods: Convolutional Neural Network (CNN), Bidirectional LSTM (Bi-LSTM), Gated Recurrent Units (GRU) and a combination of CNN with Bi-LSTM (CNN+Bi-LSTM).

CNNs are a specialized kind of neural network for data that has a grid-like topology[18]. Since images can be thought of as a 2-D grid of pixels, they are typically used for Computer Vision problems. Nevertheless, more recently

CNNs have been applied to Natural Language Processing tasks and have proven to be very effective. The idea is that the input are sentences represented as a matrix, where each row of the matrix is a vector (in our case, a word embedding) corresponding to one token[19].

To have a direct comparison to the model proposed by Kaliyar et al.[7], we used pre-trained word embedding GloVe. We now describe our CNN architecture, which can be seen in Figure 3.

First, an embedding layer takes as input the vocabulary size, the output dimension and the input length, which is set to the median length of all news in `train.csv` (Table 1). The embedding layer is connected to a convolutional layer, with a Rectified Linear Unit (ReLu) activation function, followed by a max-pooling layer and a dropout layer. The choice of ReLu has several advantages: it counteracts gradient vanishing, it is cheap to compute and favors sparsity[20]. Pooling reduces the output dimensions while maintaining the most salient features[19]; finally, dropout is a powerful regularization technique[18]. The above three layers are repeated two more times up to the fully connected layer and the output layer.



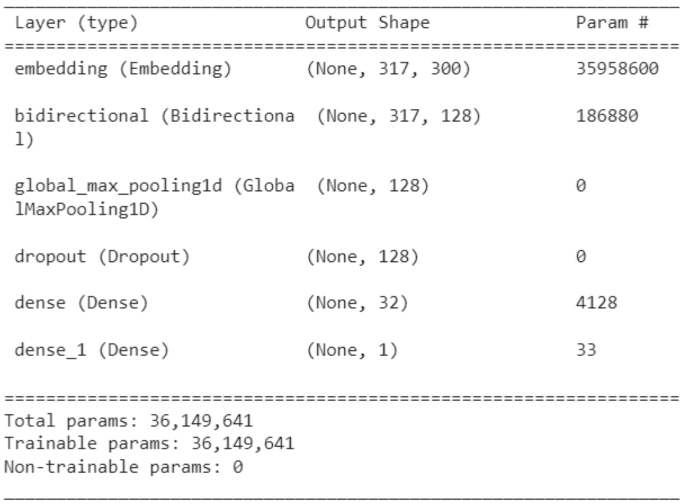
```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 317, 300)          35958600

 conv1d (Conv1D)             (None, 315, 64)           57664

 max_pooling1d (MaxPooling1D  (None, 157, 64)          0
 )

 dropout (Dropout)           (None, 157, 64)           0

 conv1d_1 (Conv1D)           (None, 155, 64)           12352

 max_pooling1d_1 (MaxPooling  (None, 77, 64)           0
 1D)

 dropout_1 (Dropout)         (None, 77, 64)            0

 conv1d_2 (Conv1D)           (None, 75, 64)            12352

 max_pooling1d_2 (MaxPooling  (None, 37, 64)           0
 1D)

 dropout_2 (Dropout)         (None, 37, 64)            0

 flatten (Flatten)           (None, 2368)              0

 dense (Dense)               (None, 64)                151616

 dense_1 (Dense)             (None, 1)                 65

=================================================================
Total params: 36,192,649
Trainable params: 36,192,649
Non-trainable params: 0
_________________________________________________________________
```

Fig. 3 CNN architecture.

Recurrent Neural Networks (RNNs) are a family of neural networks for processing sequential data[18]. Among them, Long Short-term Memory (LSTMs) are devised to solve the challenge of learning long-term dependencies. In this project, two variants of LSTM were proposed: Bidirectional LSTM (Bi-LSTM) and Gated Recurrent Units (GRU).

Bi-LSTM is basically two LSTMs positioned in opposite directions and connected to an output layer. This provides the complete past and future context information for each point in the input layer, improving memory capability of the network[21]. To have a direct comparison to the model proposed by Sastrawan et al.[4], we used pre-trained word embedding fastText. Our architecture, shown in Figure 4, consists of an embedding layer, followed by the Bi-LSTM layer, a global max-pooling and a dropout layer. It finishes as usual with the fully connected layer and the output layer.

In the last two architectures, we tried both GloVe and fastText word embeddings, to find out which was best suited for each. Furthermore, we experimented batch normalization, which is an adaptive reparameterization method that reduces the problem of coordinating updates across many layers of deep models[18].

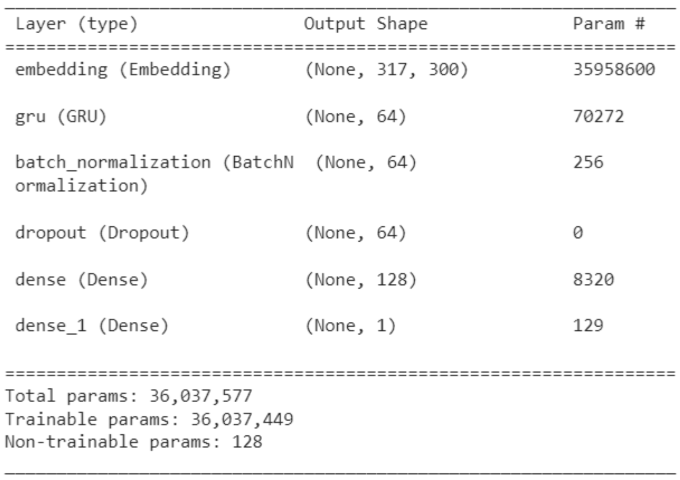


```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 317, 300)          35958600

 bidirectional (Bidirectiona  (None, 317, 128)         186880
 l)

 global_max_pooling1d (Globa  (None, 128)              0
 lMaxPooling1D)

 dropout (Dropout)           (None, 128)               0

 dense (Dense)               (None, 32)                4128

 dense_1 (Dense)             (None, 1)                 33

=================================================================
Total params: 36,149,641
Trainable params: 36,149,641
Non-trainable params: 0
_________________________________________________________________
```

Fig. 4 Bi-LSTM architecture.

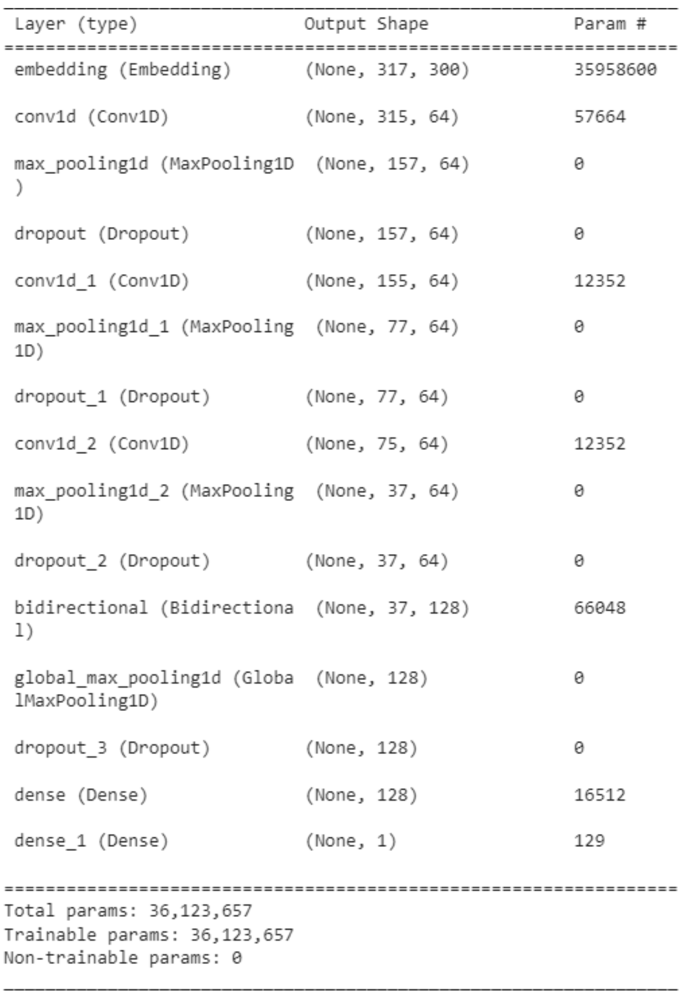
GRUs were born to answer the question of which gates of the LSTMs are really essential[18]. Indeed, GRUs are a simplification of LSTMs, so they have far fewer parameters and require less computational time. The architecture can be seen in Figure 5 and it has an embedding layer, the GRU layer, dropout, the fully connected layer and the output layer.

At the end, we implemented a hybrid model made up of CNN and Bi-LSTM (CNN+Bi-LSTM), each of which has the same structure already described (Fig. 6).



```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 317, 300)          35958600

 gru (GRU)                   (None, 64)                70272

 batch_normalization (BatchN  (None, 64)               256
 ormalization)

 dropout (Dropout)           (None, 64)                0

 dense (Dense)               (None, 128)               8320

 dense_1 (Dense)             (None, 1)                 129

=================================================================
Total params: 36,037,577
Trainable params: 36,037,449
Non-trainable params: 128
_________________________________________________________________
```

Fig. 5 GRU architecture.



```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 317, 300)          35958600

 conv1d (Conv1D)             (None, 315, 64)           57664

 max_pooling1d (MaxPooling1D  (None, 157, 64)          0
 )

 dropout (Dropout)           (None, 157, 64)           0

 conv1d_1 (Conv1D)           (None, 155, 64)           12352

 max_pooling1d_1 (MaxPooling  (None, 77, 64)           0
 1D)

 dropout_1 (Dropout)         (None, 77, 64)            0

 conv1d_2 (Conv1D)           (None, 75, 64)            12352

 max_pooling1d_2 (MaxPooling  (None, 37, 64)           0
 1D)

 dropout_2 (Dropout)         (None, 37, 64)            0

 bidirectional (Bidirectiona  (None, 37, 128)          66048
 l)

 global_max_pooling1d (Globa  (None, 128)              0
 lMaxPooling1D)

 dropout_3 (Dropout)         (None, 128)               0

 dense (Dense)               (None, 128)               16512

 dense_1 (Dense)             (None, 1)                 129

=================================================================
Total params: 36,123,657
Trainable params: 36,123,657
Non-trainable params: 0
_________________________________________________________________
```

Fig. 6 CNN+Bi-LSTM architecture.

3.3.1 Transformers

Modern machine learning encoder-decoder architectures can achieve strong performance on sequence-to-sequence tasks such as machine translation, language modeling, etc[22]. Many of these architectures make use of attention[23], a mechanism that allows the network to focus on a specific part of the input most relevant to the current prediction step[22].
Vaswani et al.[24] proposed a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely, typical of encoder-decoder configurations.
Our next experiment was considering the best performing model and using a pre-trained Transformer (BERT[17]) as its embedding layer. In particular, BERT returns the embeddings for the whole sequence as well as a pooled output. Since the pooled output is not considered a good summary of the semantic content of the input, it was discarded. These embeddings are then be fed into Bi-LSTM to produce a prediction for the classification of the input news. The final Transformer architecture can be seen in Figure 7.

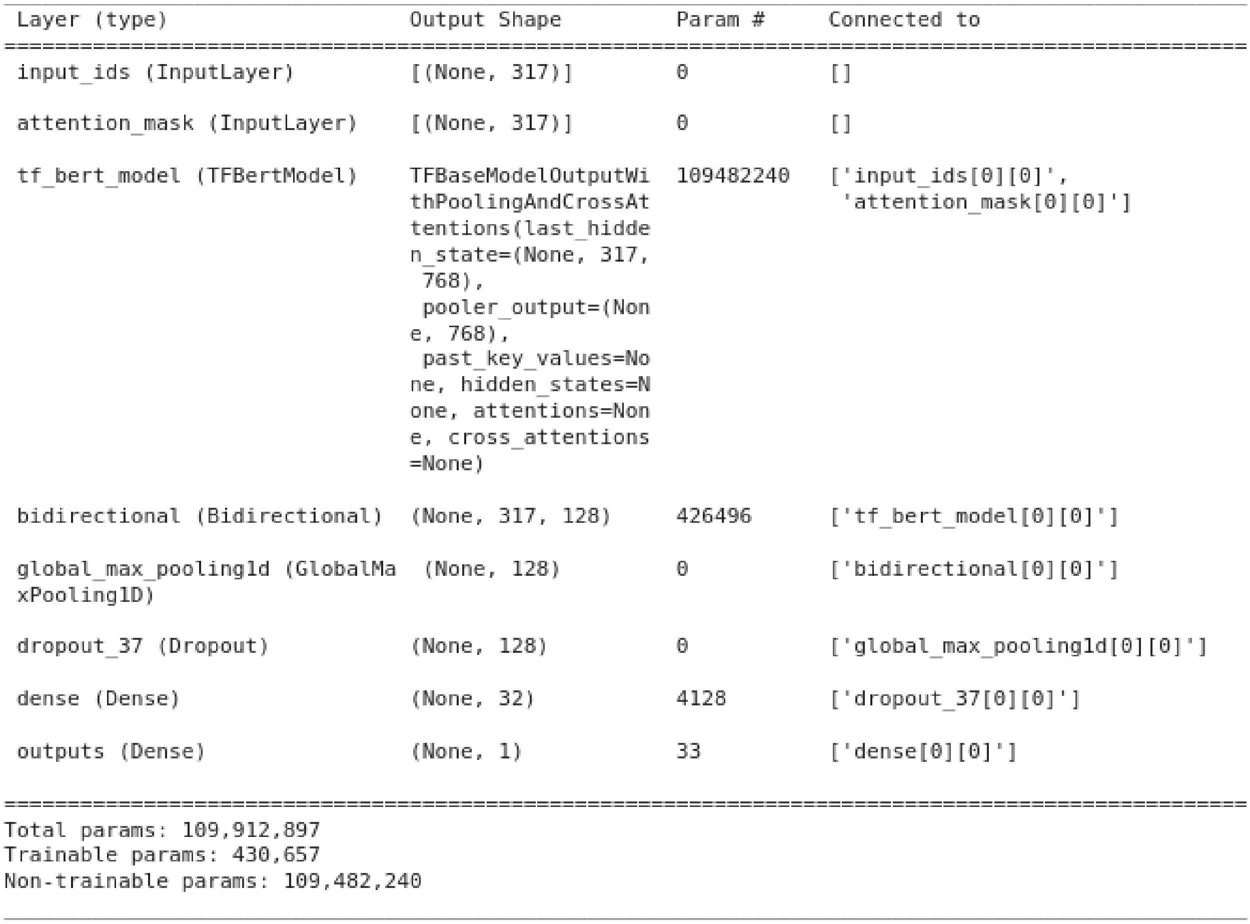
4 Experiment Setup

The project is entirely developed on Google Colab, in order to exploit its GPU. It is organized into different .ipynb files containing data processing and analysis, model selections and retraining of the best final models. Several python libraries were used, such as Tensorflow, Keras Tuner, NLTK, Numpy, Seaborn, Sklearn, Pandas and Matplotlib.

For every Deep Neural Network, a model selection was made on many hyper-parameters, to select the values giving the highest performance on the validation set. A random search was run using Keras Tuner[25], with 60 trials and validation accuracy as objective. According to "Random Search for Hyper-parameter Optimization" by Bergstra and Bengio, "*randomly chosen trials are more efficient for hyper-parameter optimization than trials on a grid*" and they obviously "*require less computational time*"[26]. Furhtermore, it has been proven that "*for any distribution over a sample space with a finite maximum, the maximum of 60 random observations lies within the top 5% of the true maximum, with 95% probability*"[27].

Since we noticed that all the models performed well in a few epochs, we set the search with an upper bound of 20 epochs. Furthermore, we introduced an early stopping criterion using Keras, with monitor on validation accuracy and patience equals to 3 epochs. Every model has been optimized with Adam method, using binary cross entropy loss.

Once model selection was done, the winning model of each architecture was retrained with the same optimizer and the batch size selected by the random search. The retraining was performed on the entire `train.csv` file, that is, rejoining the training and validation sets used for model



```
 Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
 input_ids (InputLayer)          [(None, 317)]        0           []

 attention_mask (InputLayer)     [(None, 317)]        0           []

 tf_bert_model (TFBertModel)     TFBaseModelOutputWi  109482240   ['input_ids[0][0]',
                                 thPoolingAndCrossAt               'attention_mask[0][0]']
                                 tentions(last_hidde
                                 n_state=(None, 317,
                                  768),
                                  pooler_output=(Non
                                 e, 768),
                                  past_key_values=No
                                 ne, hidden_states=N
                                 one, attentions=Non
                                 e, cross_attentions
                                 =None)

 bidirectional (Bidirectional)   (None, 317, 128)     426496      ['tf_bert_model[0][0]']

 global_max_pooling1d (GlobalMa  (None, 128)          0           ['bidirectional[0][0]']
 xPooling1D)

 dropout_37 (Dropout)            (None, 128)          0           ['global_max_pooling1d[0][0]']

 dense (Dense)                   (None, 32)           4128        ['dropout_37[0][0]']

 outputs (Dense)                 (None, 1)            33          ['dense[0][0]']

==================================================================================================
Total params: 109,912,897
Trainable params: 430,657
Non-trainable params: 109,482,240
```

Fig. 7 Bi-LSTM with BERT architecture.

selection. At this stage, we set the upper limit of epochs to 30. The stopping criterion used was instead based on the training loss of the best model of the random search.

Table 3 shows the values of the hyper-parameters considered for all model selections.

hyper-parameter	values
`filters`	32, 64, 128
`LSTM/GRU_units`	32, 64, 128
`rate_dropout`	0.1, 0.2, 0.3, 0.4, 0.5
`dense_units`	32, 64, 128
`batch_size`	16, 32, 64, 128
`learning_rate`	0.0001, 0.0005, 0.001, 0.005, 0.01
`clipnorm`	1, 3, 5

Table 3 Values of hyper-parameters in random search.

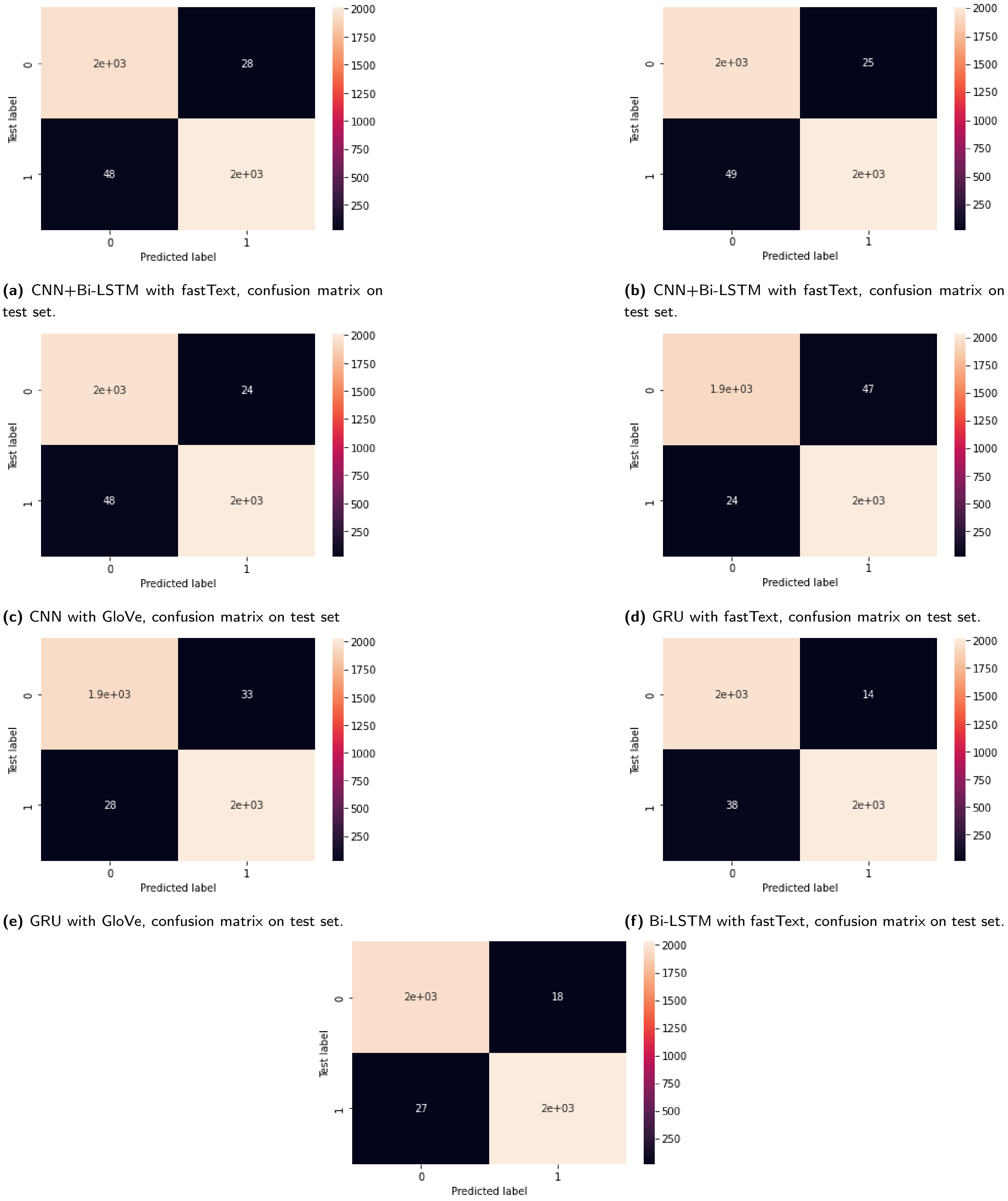
4.1 Transformers Approach

The Transformers library[28] was used to get pre-trained BERT and use it as embedding layer. We decided to freeze BERT layer, with its default parameters, and only train the remainder of the model (Bi-LSTM), which learns from the representations produced by the Transformer. Due to very high computational costs, we started from the winning Bi-LSTM configuration avoiding to perform a random search. Rather, we performed a manual model selection, changing only the value of the learning rate, which affects the model convergence rate the most. We tried different values for the learning rate hyper-parameter (0.005, 0.001, 0.0008, 0.0005), where the first three of them led to too low convergence on the validation set. At this phase, 90% of the original set was used as training set and the remaining 10% as validation. The model selection was run with 30 epochs and no patience in the stopping criterion. Finally, the best configuration - with learning rate 0.0005 - was retrained on the whole training dataset, with stopping criterion based on the training loss again.

5 Results and Discussions

Table 4 shows the results of the proposed and state of the art models on the test set.

Regarding the architectures we made a direct comparison of, it emerges that our Bi-LSTM with fastText outper-



(a) CNN+Bi-LSTM with fastText, confusion matrix on test set.

(b) CNN+Bi-LSTM with fastText, confusion matrix on test set.

(c) CNN with GloVe, confusion matrix on test set

(d) GRU with fastText, confusion matrix on test set.

(e) GRU with GloVe, confusion matrix on test set.

(f) Bi-LSTM with fastText, confusion matrix on test set.

(g) Bi-LSTM with BERT, confusion matrix on test set.

Fig. 8 Confusion matrices on test set of the proposed models.

Author	Embedding Layer	Model	Accuracy	Precision	Recall	F1-score	Training time (ms/step)
Sastrawan et al.	fastText	CNN	96.92%	96.86%	96.98%	96.91%	-
Sastrawan et al.	fastText	ResNet	98.11%	98.11%	98.09%	98.1%	-
Proposed model	fastText	CNN+Bi-LSTM	98.12%	98.11%	98.13%	98.12%	36
Proposed model	GloVe	CNN+Bi-LSTM	98.17%	98.16%	98.18%	98.17%	31
Proposed model	GloVe	CNN	98.22%	98.21%	98.23%	98.22%	25
Proposed model	fastText	GRU	98.24%	98.26%	98.23%	98.24%	32
Kaliyar et al.	GloVe	CNN	98.36%	99.4%	96.88%	98.12%	-
Proposed model	GloVe	GRU	98.49%	98.49%	98.49%	98.49%	40
Sastrawan et al.	fastText	Bi-LSTM	98.65%	98.64%	98.66%	98.65%	-
Proposed model	fastText	Bi-LSTM	98.71%	98.71%	98.72%	98.71%	62
Proposed model	**BERT**	**Bi-LSTM**	**98.88%**	**98.88%**	**98.89%**	**98.88%**	**3000**

Table 4 Comparison of the proposed models (highlighted in blue) with the state of the art on Fake News Dataset.

forms Sastrawan et al. (our 98.71% of accuracy on their 98.65%). Different is the case of CNN with GloVe, where we obtained 98.22% of accuracy over 98.36% by Kaliyar et al. The latter result is consistent with the fact that the architecture we implemented for CNN is only a simplification of theirs, while Bi-LSTM is the same with different hyperparameters and the addition of the dropout layer.

As concerns the other two models, GRU and CNN+Bi-LSTM, the comparison between the two word embeddings shows that GloVe helped both models to perform better than fastText. In particular, the experiment with the hybrid model did not show improvement over the individual CNN and Bi-LSTM themselves (98.12% and 98.17% with fastText and GloVe respectively).

Hence, the RNNs seem to be the most effective networks for this task, with GRU reaching third place in the ranking (98.49% of accuracy), even with a basic architecture. Note that, as might be expected, training GRU has a shorter computational time than Bi-LSTM (32-40 ms/step against 62).

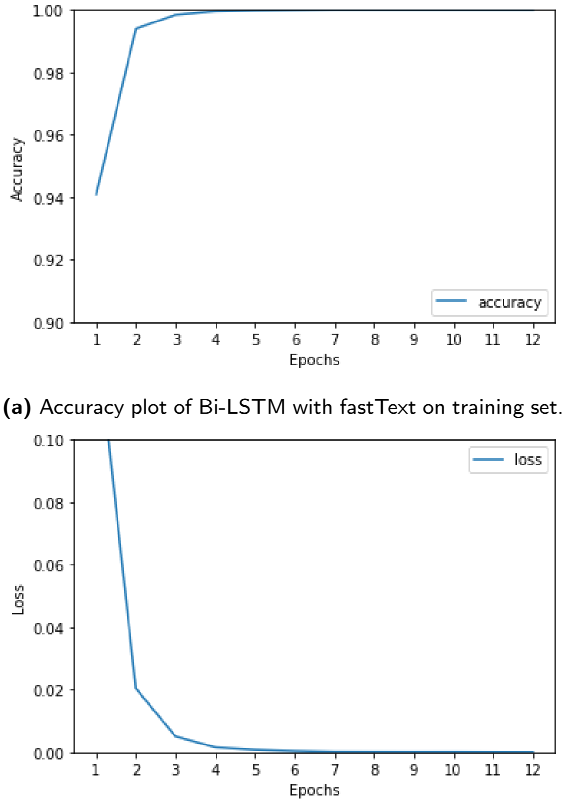
Finally, the Bi-LSTM and BERT combination performed as predicted, exceeding all previous models with an accuracy of 98.88% while having the highest computational cost (3000 ms/step).

Figure 8 shows the confusion matrix[29] of each model. The light colors on the main diagonal indicate that almost all news in the test set were classified correctly. The dark color in the opposite diagonal shows that there are few false positives and negatives.

5.1 Focus on Best Model

5.1.1 fastText

Before considering the Transformers, the architecture Bi-LSTM with fastText word embedding is the one who reaches the best test scores (98.71% of accuracy), with a relative high computational cost (62 ms/step). The accuracy and loss plots of model training are in Figure 9.



(a) Accuracy plot of Bi-LSTM with fastText on training set.

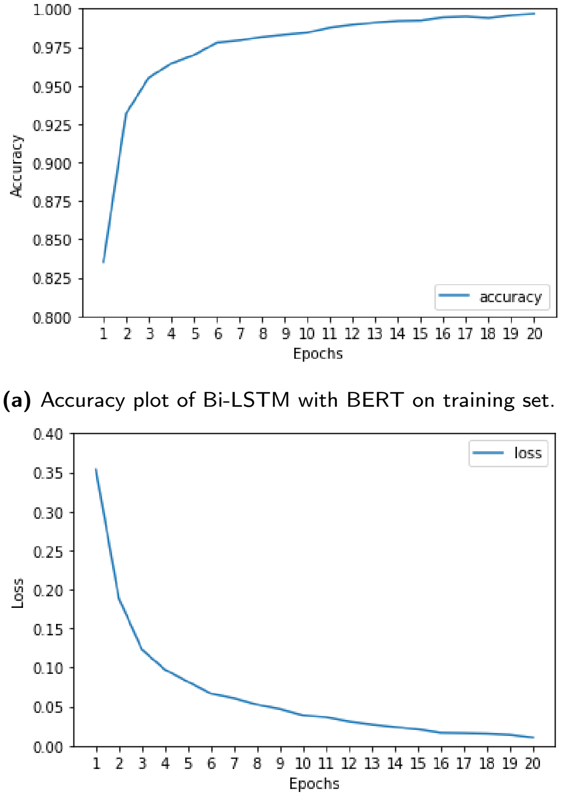
(b) Loss plot of Bi-LSTM with fastText on training set.

Fig. 9 Accuracy and loss plots of Bi-LSTM with fastText on training set.

To further challenge the winning model of the first exper-

Threshold	Accuracy	Precision	Recall	F1-score
0.50	98.71%	98.71%	98.72%	98.71%
0.80	98.54%	98.54%	98.55%	98.54%
0.90	98.44%	98.44%	98.46%	98.44%
0.95	98.36%	98.37%	98.39%	98.36%
0.99	98.12%	98.14%	98.15%	98.12%

Table 5 Test results of Bi-LSTM with fastTesxt on different classification thresholds.



(a) Accuracy plot of Bi-LSTM with BERT on training set.

(b) Loss plot of Bi-LSTM with BERT on training set.

Fig. 10 Accuracy and loss plots of Bi-LSTM with BERT on training set.

iments phase, we decided to test it by varying the classification threshold. The results, which can be seen in Table 5, prove that even with a 0.99 threshold it still achieves good performances (98.12% of accuracy).

5.1.2 Bert

With Transformers, which require an additional rise in computational cost, Bi-LSTM with BERT as its embedding layer is the top-scoring model overall. It reached 98.88% of accuracy although it was not possible to carry out an exhaustive search of its hyper-parameters. Its accuracy and loss plots can be found in Figure 10.

As before, this model has been further tested with different classification thresholds (see Table 6). According to the results, it performs worse than the fastText combination with higher classification thresholds.

Threshold	Accuracy	Precision	Recall	F1-score
0.50	98.88%	98.88%	98.89%	98.88%
0.80	98.64%	98.63%	98.65%	98.64%
0.90	98.54%	98.54%	98.56%	98.54%
0.95	98.29%	98.30%	98.32%	98.29%
0.99	97.87%	97.91%	97.91%	97.87%

Table 6 Test results of Bi-LSTM with BERT on different classification thresholds.

6 Conclusions

This study compares four neural networks with the state of the art on the Fake News Dataset[1], in the task of Fake News Detection. With two of them, namely CNN and Bi-LSTM, it directly compares itself with the work of the best performing research teams[4,7]; with the other two, namely GRU and CNN+Bi-LSTM, it attempts to place itself in the current ranking of results.

A pre-trained word embedding is present in all the architectures. In the case of direct comparisons, the same word embedding as in the state of the art was used, in particular Bi-LSTM was tested with fastText and CNN with GloVe. On the other side, GRU and CNN+Bi-LSTM were implemented with both fastText and GloVe to determine which was better for them.

Based on the test results, our combination of Bi-LSTM and fastText outperformed all the other non-tranformers based models with an accuracy of 98.71%. The updated ranking of results (Table 4) also shows that GloVe performs better with both GRU and CNN+Bi-LSTM than fastText, and that RNNs seem to be more suitable models for the task.

In the conclusion, the experiments demonstrated that Bi-LSTM with BERT, which has an accuracy of 98.88%, is the final winning model combination of our entire study.

The current fake news detection system still has much room for improvement. In the future, we would like to extend our methods by trying to expand the training dataset to cover as many topics as possible, or by using data from another language, such as Italian. Regarding models, it would be interesting to test other Transformer variations, such as XLNet or ROBERTa. Once we have obtained high-performance models, we would like to make them user-friendly by creating, for example, a web extension or a mobile app, to help people to properly inform themselves.

References

1 I Kadek Sastrawan, Bayupati, I Putu Agung, Arsa, and Dewa Made Sri. Fake news dataset, 2021. URL https://data.mendeley.com/datasets/945z9xkc8d/1.

2 Robert Schlesinger. Fake news in reality. *News & World Report*, April 14, 2017.

3 Marju Himma-Kadakas. Alternative facts and fake news entering journalistic content production cycle. *Cosmopolitan Civil Societies*, 9:25–41, July 2017.

4 I.P.A. Bayupati I.K. Sastrawan and D.M.S. Arsa. Detection of fake news using deep learning cnn–rnn based methods. *ICT Express*, 2021. doi:https://doi.org/10.1016/j.icte.2021.10.003.

5 Jeffrey Pennington, Richard Socher, and Christopher Manning. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, October 2014. Association for Computational Linguistics. doi:10.3115/v1/D14-1162. URL https://aclanthology.org/D14-1162.

6 Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. Advances in pre-training distributed word representations, 2017. URL https://arxiv.org/abs/1712.09405.

7 Rohit Kumar Kaliyar, Anurag Goswami, Pratik Narang, and Soumendu Sinha. Fndnet – a deep convolutional neural network for fake news detection. *Cognitive Systems Research*, 61:32–44, 2020. ISSN 1389-0417. doi:https://doi.org/10.1016/j.cogsys.2019.12.005. URL https://www.sciencedirect.com/science/article/pii/S1389041720300085.

8 Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space, 2013. URL https://arxiv.org/abs/1301.3781.

9 Rico Sennrich, Barry Haddow, and Alexandra Birch. Improving neural machine translation models with monolingual data, 2015. URL https://arxiv.org/abs/1511.06709.

10 Waldemar López, Jorge Merlino, and Pablo Rodríguez-Bocca. Learning semantic information from internet domain names using word embeddings. *Engineering Applications of Artificial Intelligence*, 94:103823, 2020. ISSN 0952-1976. doi:https://doi.org/10.1016/j.engappai.2020.103823. URL https://www.sciencedirect.com/science/article/pii/S0952197620301974.

11 Siwei Lai, Kang Liu, Shizhu He, and Jun Zhao. How to generate a good word embedding. *IEEE Intelligent Systems*, 31(6):5–14, 2016. doi:10.1109/MIS.2016.45.

12 Abu Bakr Soliman, Kareem Eissa, and Samhaa R. El-Beltagy. Aravec: A set of arabic word embedding models for use in arabic nlp. *Procedia Computer Science*, 117:256–265, 2017. ISSN 1877-0509. doi:https://doi.org/10.1016/j.procs.2017.10.117. URL https://www.sciencedirect.com/science/article/pii/S1877050917321749. Arabic Computational Linguistics.

13 Marwa Naili, Anja Habacha Chaibi, and Henda Hajjami Ben Ghezala. Comparative study of word embedding methods in topic segmentation. *Procedia Computer Science*, 112:340–349, 2017. ISSN 1877-0509. doi:https://doi.org/10.1016/j.procs.2017.08.009. URL https://www.sciencedirect.com/science/article/pii/S1877050917313480. Knowledge-Based and Intelligent Information & Engineering Systems: Proceedings of the 21st International Conference, KES-20176-8 September 2017, Marseille, France.

14 Roger Alan Stein, Patricia A. Jaques, and João Francisco Valiati. An analysis of hierarchical text classification using word embeddings. *Information Sciences*, 471:216–232, 2019. ISSN 0020-0255. doi:https://doi.org/10.1016/j.ins.2018.09.001. URL https://www.sciencedirect.com/science/article/pii/S0020025518306935.

15 Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 06 2017. ISSN 2307-387X. doi:10.1162/tacl_a_00051. URL https://doi.org/10.1162/tacl_a_00051.

16 Yukun Zhu, Ryan Kiros, Richard Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books, 2015. URL https://arxiv.org/abs/1506.06724.

17 Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018. URL http://arxiv.org/abs/1810.04805.

18 Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

19 Marc Moreno Lopez and Jugal Kalita. Deep learning applied to NLP. *CoRR*, abs/1703.03091, 2017. URL http://arxiv.org/abs/1703.03091.

20 Xavier Glorot, Antoine Bordes, and Yoshua Bengio.

Deep sparse rectifier neural networks. In Geoffrey Gordon, David Dunson, and Miroslav Dudík, editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pages 315–323, Fort Lauderdale, FL, USA, 11–13 Apr 2011. PMLR. URL https://proceedings.mlr.press/v15/glorot11a.html.

21 Guozheng Rao, Weihang Huang, Zhiyong Feng, and Qiong Cong. Lstm with sentence representations for document-level sentiment classification. *Neurocomputing*, 308:49–57, 2018. ISSN 0925-2312. doi:https://doi.org/10.1016/j.neucom.2018.04.045. URL https://www.sciencedirect.com/science/article/pii/S092523121830479X.

22 Kyle Aitken, Vinay V Ramasesh, Yuan Cao, and Niru Maheswaranathan. Understanding how encoder-decoder architectures attend, 2021. URL https://arxiv.org/abs/2110.15253.

23 Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate, 2014. URL https://arxiv.org/abs/1409.0473.

24 Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017. URL https://arxiv.org/abs/1706.03762.

25 Tom O'Malley, Elie Bursztein, James Long, François Chollet, Haifeng Jin, Luca Invernizzi, et al. Kerastuner. https://github.com/keras-team/keras-tuner, 2019.

26 James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(10):281–305, 2012. URL http://jmlr.org/papers/v13/bergstra12a.html.

27 Alice Zheng, Nicole Shelby, and Ellie Volckhausen. Evaluating machine learning models. *Machine Learning in the AWS Cloud*, 2019.

28 Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, October 2020. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/2020.emnlp-demos.6.

29 Stephen V. Stehman. Selecting and interpreting measures of thematic classification accuracy. *Remote Sensing of Environment*, 62(1):77–89, 1997. ISSN 0034-4257. doi:https://doi.org/10.1016/S0034-4257(97)00083-7. URL https://www.sciencedirect.com/science/article/pii/S0034425797000837.