Faculty of computer science and engineering

Fake news detection using deep learning

Research project from the subject Natural language and processing

Done by :
Laze Gjorgiev 161094

Mentor by:
prof. Sonja Gieveska

Abstract

Since the popularization and massive adoption of the online social networks, there is a significant increase of fake news generation. Fake news is written and published usually with the intent to mislead in order to damage an agency, entity, or person, and/or gain financially or politically, often using sensationalist, dishonest, or outright fabricated headlines to increase readership. For instance, during the US president election in 2016, there was enormous fake news placed on the social network. These fake news usually had the intention to damage the reputation, create false rumors about the political opponent or to create false positive claims about the favored politicians. These fake news can cause enormous damage to firstly the misguided readers and secondly the modern democratic society. Here arises a need for a method that can help or give induction for successfully classifying the given news to true or fake. In this paper we tried several approaches for news modeling and classification using deep learning techniques.

Introduction

Fake information is news, stories or hoaxes created to deliberately misinform or deceive readers. Usually, these stories are created to either influence people's views, push a political agenda or cause confusion and can often be a profitable business for online publishers. False information can deceive people by looking like trusted websites or using similar names and web addresses to reputable news organisations. The fake news aroused the attention and concerns of the public after the US presidential election in 2016 because there is a probability that these fake news might have been pivotal in the election of President Trump [1].

Since 2016 there has been an actualization of the fake news detection problem and its potential for damage to both the democracy and prospering societies. This was a reason for researches and the large tech companies like Google, Facebook, Twitter and etc. (some of them are also endangered by the fake news problem because they are online social network based companies) to try to solve the problem by creating natural language processing (NLP) based algorithms for automatic detection of fake news.

Natural language processing (NLP) is a subfield of linguistics, artificial intelligence concerned with algorithms and methods that enable computers to process and analyze large amounts of natural language human data. NLP is not something new in the world of computer science and artificial intelligence with beginnings from the late 1940s till now. NLP is a very vast research field covering many topics like machine translation [2 - 5], ontology [7], information extraction [7], natural language understanding [8], text summarization [9] and etc. But its applicability and success has grown significantly in the last 10-15 years as the rapid progress of the information technology and advances in deep learning techniques.

Deep learning is a subset area of the broader machine learning area. In the past years it has been a subject of extensive research and improvements, primarily because of their robustness and excellent performance in both classification and regression. Deep learning's nested hierarchy is able to extract and learn high-dimensional features and correlations that exist in the input data.

Over the last 10 years the deep learning architectures and algorithms have achieved superior results over the traditional machine learning models on various NLP tasks. This superiority has been brought by some major breakthrough achievements in the NLP and applying deep learning architectures and algorithms on NLP tasks. Tom Young et al. [12] have given a detailed overview and summarization

of the deep learning techniques and achievements in NLP over the last few years. One of these achievements are the word embeddings algorithms such as Word2Vec [10,11], in which authors purpose an idea and method for representing the words in a continuous vectors space that preserves the relationship between words and their linear regularities such as differences in syntax and semantics, allowing operations like computing analogies via vector addition and cosine similarity. Another great progress in the field of NLP was done by introducing BERT (Bidirectional Encoder Representations from Transformers) [8]. BERT is a new language representation model and it is designed to pre-train deep bidirectional representations from unlabeled text by learning both left and right context. As a result, the pre-trained BERT model can be easily fine-tuned with just one additional output layer to create state-of-the-art models for a wide range of tasks, such as question answering and sequence classification without additional architecture modification or large training dataset nor high computational power for training.

In [12] The authors discuss the use and results of convolutional neural networks in NLP tasks together with word embeddings, which are natural choices for deep learning architectures in computer vision. This combination (CNN and word embeddings) has shown great results on various NLP tasks such as documents summarization, information retrieval, semantics and etc. Furthermore, they discuss and analyze the use and results of the Recurrent neural networks such as simple RNN, GRU[13] and LSTM[14,15] for various NLP tasks, using word embeddings. Due the ability for capturing relationships and dependencies that exists in sequential data as it is the natural language the LSTM models have are very common and successful for modeling and solving NLP tasks and problems such as encoder-decoder architecture for sequence-to-sequence models, language generation, semantic modeling and capturing and etc [12]. Wang et al. [16] purpose a LSTM approach for social media text classification in context of sentiment using word embeddings from Word2Vec, yielding comparable results.

The rest of this paper is organized as follows. Section II reviews the related work. Section III introduces the proposed models and describes the training and architecture of the network. Section IV shows the experimental results of our model on the test sample. Section V conclusions.

2 Related work

Most of the approaches for fake news detection are based on deep learning but there are also researches who proposed traditional machine learning models for this problem. Ahmet H. et al [17] used several traditional machine learning models for fake news detection problems. They modeled the problem as a binary classification task and used bag-of-words combined with tf-idf method [18] for feature creation. One of the two datasets on which we base our experiments is the same as dataset-2 in [17]. Álvaro I. et al. [19] purpose three different neural network architectures for fake news classification problem, one of them based on BERT [8], the other two are LSTM [14,15] based model and Convolutional neural network based model, while the feature extraction process is based on creating sequence of indexes of the words in the corresponding vocabulary in order to be mapped to Word2Vec [11] representations later which is very similar to our approach. Of particular interest to the present work is the approach and model proposed by [20] in which they purpose C-LSTM model for sentiment classification consisting of stacked convolutional and LSTM neural network, we used the same model architecture in one of our approaches.

Another approach is proposed by Yang Y. et al [21] where they proposed a model named as TI-CNN (Text and Image information based Convolutional Neural Network) which tries to separate fake and true news by learning the latent relationship between the text itself in the news and the image with which the news was published. Considering the feature extraction of the text data, they also used a

similar approach as in [4] and mapped the words in the sequences to Word2Vec 100 dimension representation. Both the text data (sequence of vector representation) and image data, before concatenation of the latent representation, are modeled by Convolutional neural networks. We use the same dataset as [21] for training and performance evaluation on our models. Similar approach was proposed by Dhruv K. et al. [22] in their paper in which they purpose MVAE (Multimodal Variational Autoencoder for Fake News Detection) in which they use Variational Autoencoder and binary classification for detection fake news. They model the text data with bi-directional LSTM networks while using VGG-19 (deep convolutional neural network with 19 layers) for the image data. The VAE is trained with text and image data, learning to encode the given information into laten probability distribution and later the sampled vector from this distribution is fed into a binary classification, yielding state of art results.

3 Datasets

We conducted our experiments on two datasets:

- Dataset 1 - contains 38729 news, i.e., 20826 fake news anr 17903 real news. It is available online [1]. Due to computational concerns we randomly pick 5127 fake news and 5125 real news which are published from the period of 01:01:2016 - 31:12:2017 and subject labelled as politics content. This dataset is also used in [17].
- Dataset 2 - contains 15831 English written news, i.e., 10955 fake news and 4876 real news. It is available online[2]. The fake news are scraped from more than 240 websites by Megan Risdal on Kaggle[3], while the real news are crawled from more trusted websites such as the New York Times, Washington Post and etc. This dataset contains multiple information, such as title, text, image ref, website, author, sentiment, label (fake or real). But in our experiments we used only the text and title columns. This dataset is used for experiments in [21]

3.1 Data pre-processing

The same procedure for data preprocessing was used for both of the datasets. The first thing was to parse the words of each news text using regular expressions, so the result is a list of words without any punctuation nor number. After we used WordNet lemmatization for lemmatizing each of the parsed words. The same procedure is used with the news titles.

Next thing is to load the pre trained Word2Vec model and check the number of words from the news that do not exist in the model's vocabulary. These words are divided by checking which are valid English words. We replace the non-existing English word with the special word “uknown_word”. We use fine-tuning for updating the pre-trained Word2Vec[4] model with the new words and sequences. Through the training process the new model will learn representations for the words in the sequences that do not exist in the loaded model at first (words from the news text bodies and news titles). After the training, we add vector representation for the word “unknown_word” which is an addition of the vector representations for word “unknown” and “word”.

Now when we have vector representation for every word in the sequences representing the news, we create embedding matrix and convert the sequences of words into sequences of indexes, while the index of the every word in the sequences corresponds to the corresponding Word2Vec vector representation of the same word at the embedding matrix at that index position.

---

[1] https://www.kaggle.com/clmentbisaillon/fake-and-real-news-dataset/data#
[2] https://drive.google.com/open?id=0B3e3qZpPtccsMFo5bk9Ib3VCc2c
[3] https://www.kaggle.com/mrisdal/fake-news
[4] https://drive.google.com/file/d/0B7XkCwpI5KDYNlNUTTlSS21pQmM/edit

The labels are represented as a matrix of size news_size x 2, where every element in the matrix is either [0,1] or [1,0] for fake or real/true news corresponding.

Considering the fact that the models can be fed only with fixed size data, we choose appropriate length for the sequence in both of the datasets. In the first one we chose 434 for length which means that 86.8% of the sequences are padded, since they have less or equal length than 434 and 13.2% & will be cut. For the dataset-2 we chose the length 618, meaning that 78% of the sequences are padded, since they have less or equal length than 618 and 22% & will be cut. Considering the titles, we choose a fixed length of 11 for Dataset-1 and 10 for Dataset-2. For the BERT models the maximum sequence length for Dataset 1 is the same as for the other models, but for Dataset 2 it is 512, since this is a limitation of the pre-trained model.

4 Architectures

This section presents the proposed architectures and gives a brief intuition of the foundations of each one. We trained and evaluated 8 models, 6 of them were based on 2 core neural network architectures and 2 of them were fine-tuning on BERT pre-train.

4.1 LSTM

Long Short-Term Memory (LSTM) networks are a modified version of recurrent neural networks, which makes it easier to remember past data in memory, resolving the vanishing gradient problem of RNN. The LSTM neural networks have demonstrated high performances for modeling sequential data like time series and natural language. Their ability to capture long term dependencies in the data comes from the way the output at each step is computed. The LSTM neural network uses four gates for computing the output at each step considering the previous hidden state and cell state from the previous step and the current input as it is shown on Picture 1. The four gates are: the forget gate Ft , the input gate It , and the output gate Ot. These gates collectively decide how to update the current memory cell Ct and the current hidden state Ht. The activation functions in one LSTM module are shown below:

$It = \sigma(Wi \cdot [Ht-1, xt\ ] + Bi)$

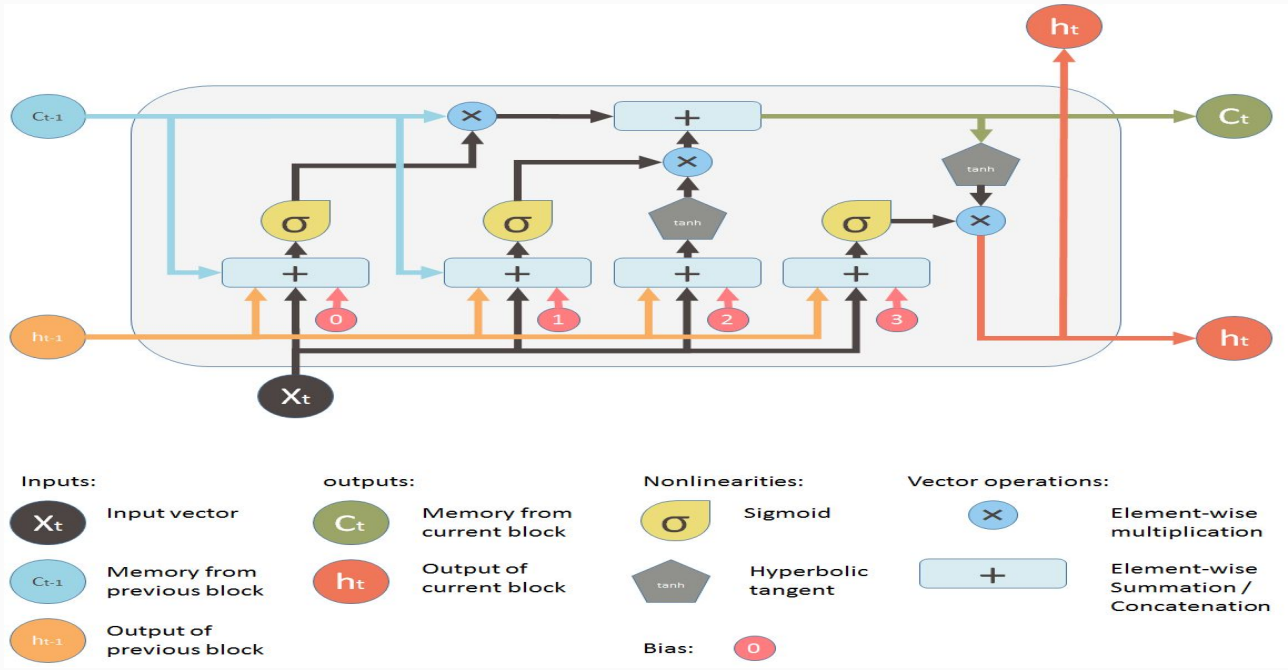
$Ft = \sigma(Wf \cdot [Ht-1, Xt\ ] + Bf\ )$

$Wt = \tanh(Wq \cdot [Ht-1, Xt\ ] + Bq)$

$Ot = \sigma(Wo \cdot [Ht-1, Xt\ ] + Bo)$

$Ct = Ft \odot Ct-1 + It \odot Qt\ Ht = Ot \odot \tanh(Ct)$

Here, σ is the logistic sigmoid function that has an output in [0, 1], tanh denotes the hyperbolic tangent function that has an output in [−1, 1], and ⊙ denotes the element wise multiplication.

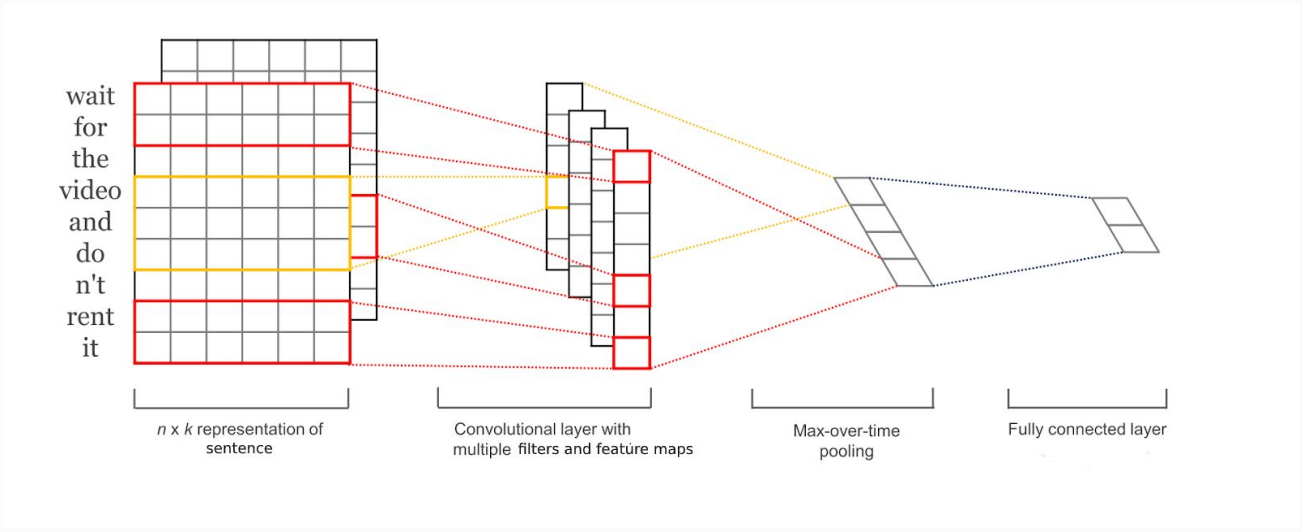


Picture 1: LSTM architecture

4.2 Convolutional neural network

Convolutional Neural Networks (ConvNets or CNNs) are a category of Neural Networks that have proven very effective in areas such as image recognition and classification. For these tasks the 2 dimensional convolutional networks are being used, demonstrating high ability for capturing spatial dependencies and characteristics in the input data. But the 2D CNN did not find very practical applications on data such as time series and natural language, because they are designed to work on 2D data such as images and videos.

As an alternative to 2D CNN, a 1d CNN has been proposed [23] for modeling 1D data and has shown great performance in features extraction from sequence data and mapping the internal features of the sequence. A 1D CNN is very effective for deriving features from a fixed-length segment of the overall dataset, where it is not so important where the feature is located in the segment [24]. Considering their ability for modeling sequential data , the 1D CNN can be applied to NLP tasks as it is proposed in [Ti-CNN, BERT]. Picture 2 shows how CNN works with NLP sequential data. On the picture the first matrix is actually sequence of vectors representation of the words (usually Word2Vec or Glove) with shape n x k, we slice the matrix using fixed sized sliding windows (kernels or region size) and on each of them a certain number of filters are applied, producing a feature map with shape (n - kernel_size) * filters (the second step in the picture). Then a pooling layer is applied taking the maximum or average from each feature map, this output can be fed to another network which can be a LSTM or fully connected neural network for example.



Picture 2: 1D convolutional neural network for NLP

4.3 BERT

BERT [8] architecture is a multi layer bidirectional Transformer encoder. It is a language representation model made by researchers at Google, with a very specific training process and therefore specific input representations.

The input representation can be both a single sentence and a pair of sentences (e.g., Question, Answer) in one token sequence. For tokens embedding, WordPiece (Wu et al., 2016) are used which have a 30,000 token vocabulary. The first token of every sequence is always a special classification token [CLS] which is later used for classification tasks, aggregating the needed information about the input sequence. In case the input is two sequences then the separation token [SEP] is used together with adding a learned embedding to every token indicating whether it belongs to sentence A or sentence B.

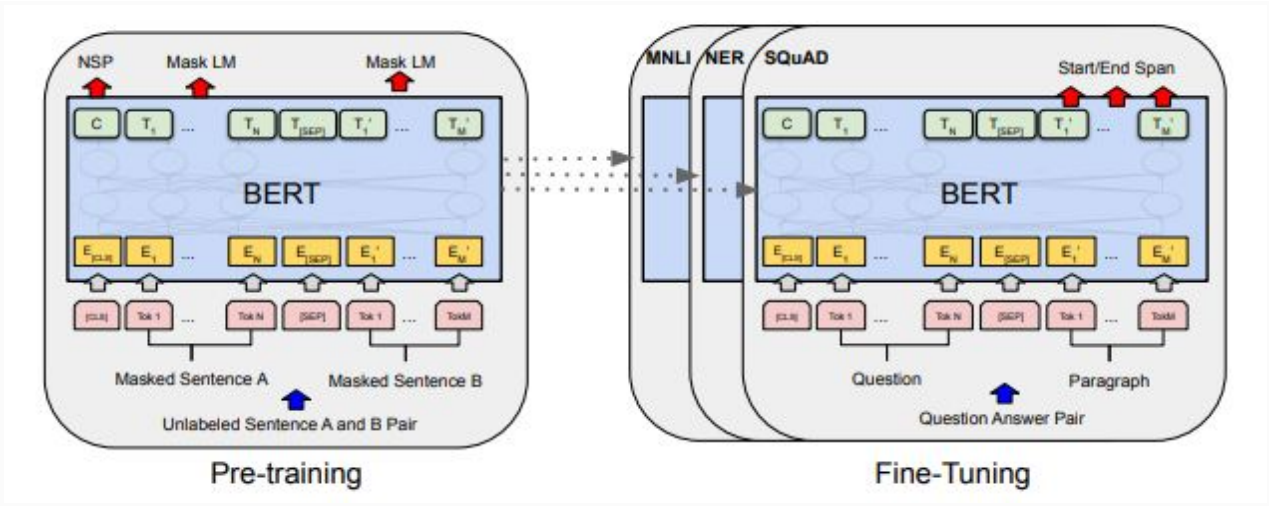
BERT is pre-trained using biderectional approach using two unsupervised tasks:

- Masked input - In order to enable biderectional training and representation, some percentage of the input tokens is masked at random using the special token [MASK], and then those masked tokens are predicted at output. This procedure is referred to as a “masked LM” (MLM). The masked tokens are predicted using a softmax over the vocabulary. According to [8] only 15% of the tokens were masked randomly during training.
- Next sentence prediction - In order to train a model that understands sentence relationships, the model is pre-trained with a binary next sentence prediction task. For example, given a corpus of text, if A and B are the chosen pair of sentences (B follows A) then in 50% of the cases B is the actual sentence that follows A (label isNext) and in the other 50% a random sentence is taken from the corpus (label notNext).

The BERT’s architecture (self attention mechanism in the Transformer) makes It’s fine-tuning very straightforward, inexpensive (considering the resource needed) and suitable for many NLP tasks (question answering, sequence classification, text degeneration and etc...).

For BERT’s pre-training BooksCorpus and English Wikipedia ware used. Google provides several pretrained models, “BASE” (L=12, H=768, A=12, Total Parameters=110M) and “LARGE” (L=24, H=1024,

A=16, Total Parameters=340M), which differ in the size (L stands for layers (transformer blocks), H for the hidden size, and A for the number of self-attention heads). In our models we use the “BASE” variant.



Picture 3: Example for BERT pre-training and fine-tuning

5 Models

Motivated by the approaches used in [19,20,21,23]] we created eight models based on the previously defined architectures. A short description of each of the six models follows:

LSTM-Model - A LSTM based model consisting of embedding layer (based on the fine-tuned Word2Vec described in section 3) connected to two stacked LSTM layers (the first layer return sequences, the second return just the output of the last step), connected to two stacked fully connected layers from with the last one is the output layer. The hyperparameters of the corresponding layers are the following:

- Embedding layer : Trainable = False, input_length = sequence length according to the dataset (413 for Dataset-1 and 618 for Dataset-2), the corresponding vocabulary size which again depends on Word2Vec, and embedding dimension = 300.
- LSTM layers - hidden_units = 64, dropout and recurrent dropout is 0.15, activation = tanh.
- Dense layer with 16 hidden units and relu activation function.
- Dense layer (the output layer) has 2 hidden units and sigmoid activation function.

CNN-Model - A CNN based model consisting of embedding layer (based on the fine-tuned Word2Vec) connected to one CNN layer, dropout layer MaxPool1D layer, GlobalMaxPool1D and fully connected layer with sigmoid activation.
The hyperparameters of the corresponding layers are the following:

- The embedding layer is the same as in LSTM-Model.
- CNN layer : Number of filters (depends on the dataset for example 8 for Dataset 1 and 16 for Dataset 2), kernel size is 12 and 18 for Dataset 1 and Dataset 2 corresponding, l2 regularization parameter (0.09).
- Dropout layer with parameter 0.2.
- MaxPool1D layer with a pool size of 15.

- GlobalMaxPool1D - no parameters
- Dense layer (the output layer) has 2 hidden units and sigmoid activation function.

CNN-LSTM-Model - A CNN and LSTM based model, which uses CNN to extract spatial features and then feds these features into the LSTM layer. The model has an embedding layer (based on the fine-tuned Word2Vec) connected to one CNN layer, dropout layer and MaxPool1D layer. The MaxPool1D layer outputs the extracted latent features into a LSTM layer which is connected to fully connected layer with sigmoid activation at the end.
The hyperparameters of the corresponding layers are the following:

- The embedding layer is the same as in LSTM-Model.
- CNN layer : Number of filters (depends on the dataset for example 14 for Dataset 1 and 25 for Dataset 2), kernel size is 12 and 18 for Dataset 1 and Dataset 2 corresponding, l2 regularization parameter (0.09).
- Dropout layer with parameter 0.2.
- MaxPool1D layer with a pool size of 12.
- LSTM layers - hidden_units = 5, dropout and recurrent dropout is 0.15, activation = tanh.
- Dense layer (the output layer) has 2 hidden units and sigmoid activation function.

LSTM-T-Model - This model is similar to the LSTM-Model, with the difference that here we ingest latent information about the titles of the news before the last layer (the output layer). We create a separate 'mini' LSTM-Model for modeling and capturing the essential information in the titles and then we concatenate these information with the information obtained from modeling the text (the output of the last LSTM layer) with the 'mini' LSTM-Model which models the titles data.
The hyperparameters of the corresponding layers from mini LSTM-Model are the following:

- Embedding layer : Trainable = False, input_length = sequence length according to the dataset (- for Dataset-1 and 10 for Dataset-2), the corresponding vocabulary size which again depends on Word2Vec, and embedding dimension = 300.
- LSTM layer - hidden_units = 4, dropout and recurrent dropout is 0.15, activation = tanh.

CNN-T-Model - Similarly to LSTM-T-Model (explained above) but instead of LSTM-Model and LSTM layers we use the CNN-Model for modeling the text data and 'mini' CNN-Model for capturing the local dependencies in the titles and then combine these latent features from the two model before feeding it into the last output layer. The hyperparameters of the corresponding layers from mini CNN-Model are the following:

- Embedding layer : Trainable = False, input_length = sequence length according to the dataset (- for Dataset-1 and 10 for Dataset-2), the corresponding vocabulary size which again depends on Word2Vec, and embedding dimension = 300.
- CNN layer : Number of filters (depends on the dataset for example - for Dataset 1 and 2 for Dataset 2), kernel size is - and 4 for Dataset 1 and Dataset 2 corresponding, l2 regularization parameter (0.01).
- Dropout layer with parameter 0.2.
- MaxPool1D layer with a pool size of 2.
- GlobalMaxPool1D - no parameters

We combine the output from the GlobalMaxPool1D layer from both models and feed this data into the last output fully connected layer with 2 hidden units and sigmoid activation function.

CNN-LSTM-T-Model - Similar to CNN-LSTM-Model, but here we have the same model for modeling the titles data as in CNN-T-Model, and these latent features about the titles are concatenated with the output

of the LSTM layer in the model and fed into the fully connected layer which is connected to the output layer which is also fully connected neural network with 2 hidden units and sigmoid activation function.

BERT-Model - This model is using the pre-trained BERT "BASE" version. We fine-tuned this version for sequence classification**.** Google's uncased pre-trained BERT model was fine-tuned for fake news classification using only the text from the articles. The fine-tuning was done by adding an additional fully connected layer with 2 neurons with linear activation function plus a softmax function on top to allow the interpretation of the result as a probability as it shown in [19]. The loss we used during training was Cross-Entropy loss.

BERT-T-Model - Similar to BERT-Model with only difference that we additionally use the titles from the articles. The title is concatenated with the text from the corresponding article and is fed to the model (after applying the needed transformations).

5.1 Training the model and metrics for performances evaluation

The dataset is splitted into training, validation and test sets. The training and validation set are 80% of the dataset and the other 20% is for testing. The split is done randomly. It is worth to note that considering the imbalanced distribution of fake and true news in Dataset-2 (10955 fake news and 4876 real news) we made sure that in the test set the two labels will be equally distributed (50% from each). The models were trained using 256 batch size and 15-30 epochs depending on the model. We used binary cross entropy as a loss function.

Considering the metrics for performance evaluation we used accuracy, precision, recall, f1-measure, with the false news being the positive class and ROC score (for Dataset-2).

6 Discussion of results

For every dataset we will choose our best BERT based and non-BERT model and compare their performances with the other's models performances proposed by other authors for the same datasets.

6.1 Dataset-1

On the table 1 are shown the results obtained from the experiments with Dataset-1. According to the results shown in the table, the best performing model considering all metrics is **BERT-Model**. This model gave perfect results for this classification task, with highest as possible accuracy. This means that the model using the already learned knowledge from the pre-train process managed to perfectly learn the difference between the true and fake news. **Bert-T-Model** also showed perfect performances with just slightly lower than **BERT-MODEL**. This means that modeling the titles together with the news text added some noise during the training and slightly lowered the performances.

About the other six models that are not BERT based, the best model was **CNN-LSTM-Model** which yielded almost perfect results**.** It seems that this model through the CNN layers were able to capture the spatial and local features in the input word-vector sequences and then the LSTM layer managed to learn the sequential high-level features and dependencies in the sequences, managing to capture and learn high-level and latent differences between the fake and true news.

The current findings demonstrate a bit lower performance achieved by the LSTM-Model, the LSTM layers were not able to capture and learn the high-level and latent differences between the fake

and true news as it was performed by the other models. The reason for that may be the small size of the sampled dataset.

The CNN-Model yielded almost perfect results, but just a bit lower than the CNN-LSTM-Model. The reason for that probably is the inability of CNN to capture sequential and long term dependencies. So in this case the CNN-Model was able to capture the local and spatial characteristics, but still demonstrated almost perfect performances.

The LSTM-T-Model demonstrated better performances than LSTM-Model, which is expected since this model is also modeling and using data about the title of the news.

Interesting results were yielded by CNN-T-Model, which had very good performances but a slight lower than CNN-Model, although the CNN-T-Model is fed by both text and title data. This might mean that the title data brought some noise and is making it more difficult for the model to learn the differences between fake and true news, stopping it from yielding as good or better results than the CNN-Model.

Little or no evidence has supported the advantages of using CNN-LSTM-T-Model, which is interesting and unexpected. This model had the lowest performances and the reason for this is probably the small sample of dataset, which made it harder for the model to converge into global or acceptably good local optimum.

Model	Accuracy	Precision	Recall	F1-measure
LSTM-Model	0.918	0.894	0.951	0.921
CNN-Model	0.994	0.995	0.992	0.993
CNN-LSTM-Model	**0.996**	**0.995**	**0.997**	**0.996**
BERT-Model	**1.00**	**1.00**	**1.00**	**1.00**
LSTM-T-Model	0.928	0.967	0.889	0.926
CNN-T-Model	0.993	0.994	0.993	0.994
CNN-LSTM-T-Model	0.830	0.834	0.830	0.832
BERT-T-Model	0.999	0.999	0.999	0.999

Table 1. Proposed model's performance comparison

On table 2 are shown the performances of our best BERT based model and the best model from the other six models - **CNN-LSTM-Model,** compared to the models proposed in [17]. It is worth noting that in [17] researchers sampled 12600 articles from each label, while we sampled around 5100 from each, but due to the random sampling we will assume that the sample size did not affect the results. We can see that our model shows superior results compared to other models. Actually even six of our proposed models and approaches (considering also the feature engineering part) show superior performances compared to the performances of the traditional models and features based in n-grams and tf-idf proposed in [17]. We think that the difference in the performance was made by two main factors:

1. The use of tf-idf features with bag-of-words did not provide enough information for the traditional models to capture the relations, between the words, the semantics and context in the sequences (the news). On the other hand, in our approach we used Word2Vec word representations and

mapped the word sequences into vector sequences and by that we kept the word relations, the contextual and semantic information that exist in the text.

2. The traditional models were not able or were able but not as good to capture local or sequential relationships between the data as is done by CNN and LSTM.

Model name	Accuracy
BERT-Model	**1.00**
CNN-LSTM-Model	**0.996**
SVM (unigram, 5000 features TF-IDF)	86.0
LSVM (unigram, 50 000 features TF-IDF)	92.0
KNN (unigram, 5000 features TF-IDF)	83.0
DT (unigram, 50 000 features TF-IDF)	89.0
SGD (unigram, 50 000 features TF-IDF)	89.0
LR (unigram, 50 000 features TF-IDF)	89.0

Table 2. Ours best model- **BERT-Model** and best non-BERT model - **CNN-LSTM-Model** compared to the best models in [17]

6.2 Dataset-2

On the table 3 are shown the results obtained from the experiments with Dataset-2. Based on the results, the best performing model across all metrics is **BERT-T-Model.** The model managed to successfully learn almost all differences between the true and fake news during the fine-tuning process using the knowledge and natural language understanding obtained during the pre-train phase. Additionally, based on the comparison with the **BERT-Model's** performances, we can conclude that using the titles together with the text from the news articles improved the model performances and helped the model to do better separations between the true and the fake news. Both BERT based models have shown batter performances than the other six, non-BERT, models, which is expected, since the BERT based models, are a fine-tuning of already pre-trained BERT model over a large data corpus, thus this model have significantly more knowledge and natural language representation than the other models trained only on this dataset.

Considering the other six non-BERT models, the best performing model considering almost all metrics (except ROC) is the **CNN-T-Model** which yielded good results**.** It seems that this model, through the CNN layers, were able to capture the spatial and local features in the input word-vector sequences representing the text body and the local features from the titles data, managing to capture and learn high-level and latent differences between the fake and true news.

The LSTM-Model and LSTM-T-Model showed solid performances, and interestingly as opposed to the results with the previous dataset, here LSTM-T-Model has poorer performances considering the accuracy and ROC metric, while it has better performances considering recall, precision and f1-measure obviously.

The CNN-Model showed very good performances, but not as good as CNN-T and CNN-LSTM-Model. The reason for that probably is the inability of CNN to capture sequential and long term dependencies. So in this case the CNN-Model was able to capture the local and spatial characteristics and it did not receive any info about the title data, but still demonstrated almost perfect performances.

Also, good results were yielded by CNN-LSTM-T-Model. It had excellent performances and the best ROC score, but in all the other metrics it had a slightly lower performance than CNN-Model, although the CNN-LSTM-T-Model is fed by both text and title data and uses both CNN and LSTM. This might mean that the sequential and long term dependencies in the words (news) are not as important as the local and neighborhood information in the sequences. This might also be the reason why the CNN-LSTM-Model had poorest performances among the other models, but second best on recall score.

Method	Accuracy	Precision	Recall	F1-measure	ROC
LSTM-Model	0.927	0.895	0.968	0.930	0.926
CNN-Model	0.939	0.900	0.986	0.941	0.939
CNN-LSTM-Model	0.897	0.835	0.990	0.906	0.898
BERT-Model	0.984	0.990	0.991	0.990	0.985
LSTM-T-Model	0.913	0.911	0.970	0.940	0.870
CNN-T-Model	**0.951**	**0.918**	**0.991**	**0.953**	0.949
CNN-LSTM-T-Model	0.950	0.917	0.989	0.952	**0.950**
BERT-T-Model	**0.991**	**0.994**	**0.996**	**0.995**	**0.991**

Table 3. Performances comparison of proposed models

On table 2 are shown our best BERT based and non-BERT models performances compared to the two best models proposed in [17]. Based on this comparison, we can see that our best model-BERT-T-Model outperformed all other models considering all metrics. The reasons for the superiority of the BERT-T-Model are probably the powerful architecture of BERT, the bidirectional approach to natural language understanding, the use of a transformer and the larger corpus on which it was pre-trained by Google.

Additionally, we can see that our best, non-BERT, **CNN-T-Model** model outperforms TI-CNN model and LSTM-text-400 model in terms of recall and f1-measure. The TI-CNN model is similar to our **CNN-T-Model** in terms of processing the text using CNN, but also process the image which is associated with the news it is modeling. The image can bring some additional useful information sometimes, but in general it brings too random information and ends up ingesting noise into the model. TI-CNN showed some better precision score than our method, but fails to detect as much of the fake news as our method did. Also LSTM-text-400 showed poorer performance than our method in terms of all metrics. The reason for the superior performances of our best non-BERT method probably comes both from the approach we used towards the feature engineering and the model itself. Namely we used 300 dimensions for word representation in contrast with 100 dimensions used with TI-CNN and LSTM-text-400. In addition we replaced the non-existing English word with the ‘unknown_word’ or represented by a vector which is a result from the addition between the vector representing ‘unknown’ and ‘word’. We believe that these

approaches allowed our method to capture more complex semantic, contextual information as well as relations that exist between the words in the sequences (news). Another difference (although it is not specified how the title data is used in [21]] but we assume it was concatenated with text body data) is the way we incorporated the information from the titles, which based on the results on Table 3 brought some useful information towards the classification. Additionally, we used 618 fixed size for the sequences in contrast with 1000 fixed size used in TI-CNN, which means that more sequences were padded with zero vectors. Considering the differences in the model, as we said above, we believe that the image information is too random in general case and they do not bring as much useful information towards the classification. The reasons are similar for the superior results of our model over LSTM-text-400.

Method	Precision	Recall	F1-Measure
BERT-T-Model	**0.994**	**0.996**	**0.995**
CNN-T-Model	0.918	**0.991**	**0.953**
TI-CNN	**0.922**	0.923	0.921
LSTM-text-400	0.9146	0.8704	0.8920

Table 4. BERT-T-Model and CNN-T-Model compared to the best models in [21]

In table 5 are shown our best performing BERT based and non-BERT based models results in terms of accuracy ROC score compared with the proposed methods in [19]. Here we need to note that just a part of the used dataset in [19] is the same as our dataset-2, because it is augmented with additional data with larger size as described in [19]. Beside this core difference, we compared our method with the methods in [19]. Our best BERT based model- BERT-T-Model showed superior performances to all models proposed in [19].

About our best non-BERT model, we can see that our method outperforms both the LSTM-Model and CNN model from [19] in terms of accuracy, but has poorer performance in terms of ROC score. In terms of accuracy it was outperformed only by BERT, which can be expected due to the use of pretrained Google model and the fine-tuning with larger dataset.

Considering the feature engineering approach, we use a similar approach by using both the text and title data with Word2Vec embedding vectors. Still our approach showed better performances, although it was trained with a smaller dataset.

Method	Accuracy	ROC
BERT-T-Model	**0.99**	**0.99**
CNN-T-Model	**0.95**	0.949
LSTM model (from [19])	0.91	0.97
CNN (from [19])	0.937	**0.98**
BERT (from [19])	0.98	0.97

Table 5. BERT-T-Model and CNN-T-Model compared to the models used in [19]

7 Conclusion

The present research proposed a specific feature engineering approach and explores several approaches towards deep neural network architectures on the task of fake news detection. We proposed eight different methods (models) for fake news detection, four of them used just the text body data from the news and the other four incorporated the titles data also. Two of these eight models are BERT based, which means they are a fine-tuning of the pre-trained BERT model. Considering the other six, non-BERT models, three of these models were purely LSTM and CNN based (LSTM-Model, CNN-Model, CNN-T model) and the other three were a result of a combination of CNN and LSTM (LSTM-T-Model, CNN-LSTM-Model, CNN-LSTM-T-Model). We used two datasets for training and evaluating the proposed models. Based on the obtained results, for both datasets, the BERT based models showed superior performances over the other six models, which is expected, considering the powerful BERT architecture, It's bidirectional approach for learning, the use of a transformer and the fact that the model was already pre-trained with large corpus of data by Google. From the other six models, the model CNN-LSTM-Model yielded best results on Dataset-1 while CNN-T and CNN-LSTM-T showed best performances on Dataset-2. Our approaches showed excellent results outperforming the previously proposed models on the same datasets. However testing on benchmark datasets and better cross validation is needed to fully examine the potential of the proposed approaches.

References:

1. Hunt Allcott and Matthew Gentzkow. Social media and fake news in the 2016 election. Technical report, National Bureau of Economic Research, 2017.
2. Mikel Artetxe, Gorka Labaka, Eneko Agirre, and Kyunghyun Cho. Unsupervised neural machine translation. arXiv preprint arXiv:1710.11041, 2017.
3. Peter Brown, John Cocke, S Della Pietra, V Della Pietra, Frederick Jelinek, Robert Mercer, and Paul Roossin. A statistical approach to language translation. In Proceedings of the 12th conference on Computational linguistics-Volume 1, pages 71–76. Association for Computational Linguistics, 1988.
4. Peter F Brown, Vincent J Della Pietra, Stephen A Della Pietra, and Robert L Mercer. The mathematics of statistical machine translation: Parameter estimation. Computational linguistics, 19(2):263–311, 1993.
5. S. P. Singh, A. Kumar, H. Darbari, L. Singh, and S. Jain, "Machine translation using deep learning: an overview," in Proceedings of the 2017 International Conference on Computer, Communications and Electronics (Comptelix), IEEE, Jaipur, India, July 2017.
6. Mario A. Paredes-Valverde, José A. Noguera-Arnaldos, Cristian Aarón, Rodríguez-Enríquez, Rafael Valencia-García, Giner Alor-Hernández, A Natural Language Interface to Ontology-Based Knowledge Bases, January 2015, Advances in Intelligent Systems and Computing 373:3-10
7. Sonit Singh , Natural Language Processing for Information Extraction, 2018, arxiv:1807.02383
8. Jacob Devlin and Ming-Wei Chang and Kenton Lee and Kristina Toutanova, BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, 2018, arxiv:1810.04805

9. Mehdi Allahyari, Seyedamin Pouriyeh, Mehdi Assef, Saeid Safae, Elizabeth D. Trippei, Juan B. Gutierrez, Krys Kochut, Text Summarization Techniques: A Brief Survey (2017), arxiv:1707.02268
10. Tomas Mikolov, Kai Chen, Greg Corrado, Jeffrey Dean, Efficient Estimation of Word Representations in Vector Space (2013), arxiv:1301.3781v3
11. Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, Jeffrey Dean, Distributed Representations of Words and Phrases and their Compositionality (2013), arxiv:1310.4546
12. "Recent Trends in Deep Learning Based Natural Language Processing" — Tom Young, Devamanyu Hazarika, Soujanya Poria, and Erik Cambria. IEEE Computational Intelligence Magazine, 2018.
13. K. Cho, B. Van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase ¨ representations using rnn encoder-decoder for statistical machine translation," arXiv preprint arXiv:1406.1078, 2014.
14. S. Hochreiter and J. Schmidhuber, "Long short-term memory," Neural computation, vol. 9, no. 8, pp. 1735–1780, 1997.
15. F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: Continual prediction with lstm," 9th International Conference on Artificial Neural Networks, pp. 850–855, 1999.
16. Jenq-Haur Wang, Ting-Wei Liu, Xiong Luo, Long Wang, An LSTM Approach to Short Text Sentiment Classification with Word Embeddings, The 2018 Conference on Computational Linguistics and Speech Processing, ROCLING 2018, pp. 214-223
17. Ahmed H, Traore I, Saad S. (2017) "Detection of Online Fake News Using N-Gram Analysis and Machine Learning Techniques. In: Traore I., Woungang I., Awad A. (eds) Intelligent, Secure, and Dependable Systems in Distributed and Cloud Environments. ISDDC 2017. Lecture Notes in Computer Science, vol 10618. Springer, Cham (pp. 127-138).
18. Ramos, Juan Enrique. "Using TF-IDF to Determine Word Relevance in Document Queries." (2003).
19. Álvaro Ibrain Rodríguez and Lara Lloret Iglesias, Fake news detection using Deep Learning, (2019), arXiv:1910.03496
20. Zhou, C.; Sun, C.; Liu, Z.; Lau, F. C. M. 2015. A C-LSTM Neural Network for Text Classification. CoRR Volume abs/1511.08630. Available online: https://arxiv.org/abs/1511.08630.
21. Yang Yang and Lei Zheng and Jiawei Zhang and Qingcai Cui and Zhoujun Li and Philip S. Yu, TI-CNN: Convolutional Neural Networks for Fake News Detection, 2018, axriv:1806.00749
22. Dhruv Khattar, Jaipal Singh Goud, Manish Gupta, Vasudeva Varma, MVAE: Multimodal Variational Autoencoder for Fake News Detection, The Web Conference-2019, Report No: IIIT/TR/2019/-1
23. S. Kiranyaz, T. Ince, M. Gabbouj, Real-Time Patient-Specific ECG Classification by 1-D Convolutional Neural Networks, IEEE Trans. Biomed. Eng. 63 (2016) 664–675. doi:10.1109/TBME.2015.2468589.
24. https://missinglink.ai/guides/keras/keras-conv1d-working-1d-convolutional-neural-networks-keras/