Fake news classification

Hamid Dimyati
dimyati@kth.se

Sri Janani
sjre@kth.se

Ryan Yared
rcyared@kth.se

Abstract

In this course project, we attempt automatic fake news detection using deep learning on the Liar dataset, a labelled fake news dataset taken from PolitiFact.com comprising of statements and metadata. We add metadata by performing sentiment analysis, dependency parsing (DEP), and part-of-speech (POS) tagging and examine their effects on performance. CNN, Bi-LSTM, and hybrid CNN-BiLSTM models based off of the architectures presented in the original Liar paper and a report paper are explored. Additionally, the effects of the augmented metadata are tested. It is found that the additional features could increase the fake news classifier performance.

1 Introduction

The Internet has provided a new way for people to consume news. However, the ease of accessing news comes with the danger of fake news spreading. This fake news could mislead people's perception of reality, and, on a larger scale, it could even yield to a chaotic society [1]. Fake news detection arises as a new branch of the Natural Language Processing research since it cannot be identified only by a human. In recent years, many researchers tried to create models to detect fake news using deep learning algorithms such as Wang [2], Bajaj [3], and Ruchansky et al. [4].

In this project, we train several deep learning algorithms to classify fake news from Liar datasets [2]. In order to provide enough information to the models, apart from the text of the statements, we append some additional features such as the meta information (speaker, subject, speaker's party, speaker's job title, and others), dependency parsing, part-of-speech tagging, and sentiment of the statement. Our results show that adding those features could statistically significantly increasing the accuracy of the baseline model which is using only the text of the statements.

2 Related Work

A plethora of works related to fake news classification are available. We have referred to a few of those and have based our project on the following literature: Wang [2] published a benchmark dataset (LIAR) for fake news detection. The dataset consists of short political statements and connected metadata. Each datapoint has been manually categorized into six truthfulness ratings (pants-fire, false, barely-true, half-true, mostly-true and true). Wang [2] proposes and evaluates their proposed fake news detection on a hybrid model, consisting of CNNs and a Bi-LSTMs, by testing it on the LIAR dataset and was able to achieve an accuracy of around 27% using all the features provided in the native dataset. In a paper by Ajao et al [5], the authors start with a hypothesis that there exists a relation between fake news and sentiment expressed by the news and propose a sentiment aware fake news classification model. Their hypothesis is mainly based on the belief that fake news generally tends to have some negative connotation. PHEME - labelled twitter dataset was used for the experiment, sentiment score was computed based on the emotional ratio obtained for each text corpus. Emotional ratio is the ratio between the count of number of negative emotional words and positive emotional words. After several experiments with different models and other meta features, it was shown that using emotional score has improved the performance of fake news classification. Both

of these works serve as the foundation on which the proposed project is developed upon. In other works, Bajaj [3] performs a binary classification of fake news or not on news articles. The articles were embedded using GLoVe. The author experiments with various models - Two-layer feed forward Neural Network, Logistic Regression, RNN, LSTM, Gated Recurrent Units, Bidirectional RNN with LSTMs, CNN with max pooling, Attention-Augmented CNN. In comparison to other methods, Logistic Regression and CNN did not perform well. The conclusion that was drawn from this result is that CNN requires additional features, not just the text corpus. Usage of dense vectors like GLoVe embeddings require nonlinear kernels for good results. From the literature, it was observed that common architectures experimented with, are CNN and Bi-LSTM or some hybrid model involving them. So for this project, it was chosen to have CNNs and Bi-LSTMs as the main architectures. Regarding word vectors, there are a myriad of pre-trained models, GloVe [6], Google's word2vec from news [1] being some of the more frequently used ones within the literature.

3 Methodology

In this project, three different Neural Network architectures are implemented and tested. These architectures were inspired by Wang [2] and Roy et al. [7]. Wang [2] presents a hybrid architecture by combining a Bi-LSTM and a CNN architecture, whereas Roy et al. [7] presents two models - a CNN model and a Bi-LSTM model, both utilizing metadata. Each architecture explored has multiple inputs and a single output.

3.1 Datasets

In this project, we are using the "LIAR" dataset published by Wang [2]. The dataset consists of 12.8K short statements from PolitiFact.com. Each statement is manually labelled using the labels: *pants-fire*, *false*, *barely-true*, *half-true*, *mostly-true*, and *true*. The dataset is pre-split into training, validation, and testing datasets by the author. It is also considered to be a benchmark dataset for fake news detection. The columns of the dataset contains the following information:

1. Statement ID
2. Label of the statement
3. The statement itself
4. Subject(s)
5. Speaker (person who made the statement)
6. Job title of the speaker
7. Speaker's State (one of 52 American states)
8. Political party to which the speaker is affiliated
9. barely true counts
10. false counts
11. half true counts
12. mostly true counts
13. pants on fire counts
14. The context (venue / location of the speech or statement)

Columns 9-13 represent the total credit history of the speaker. For example, column 9 indicates the number of times a *barely true* statement has made by the speaker (in column 5), this count includes the statement from column 3 (statement to be classified) as well. Columns from 4-14 are called metadata whereas column 2 is the output categories.

3.2 Data Preprocessing

Data preparation plays an essential role when it comes to the development of deep learning applications. In this project, five different pre-processing and feature extraction methods are employed and they are described below.

Statement preprocessing: Input to a neural network are usually vectors containing real numbers. However, the main input to the network is a statement – a string of words. These words are converted into vectors using Google news word2vec [1]. Google word2vec is a pre-trained word2vec model that was trained on 100 billion words from the Google News dataset and converts each word into

[1] https://code.google.com/archive/p/word2vec/

a 300-dimensional vector. All the network architectures that were experimented with in this paper have the same pre-processing for *Statement* input. In the pre-processing step, statements from training, validation, and test datasets are fed to the Tokenizer API from Keras. The tokenizer creates a vocabulary dictionary of unique words from these statements. Each unique word is assigned an integer key. The pre-trained Google word2vec model is fit on this vocabulary obtained from the datasets and a word2vec embedding matrix is created, where each row represents a word. Each word in the vocabulary is represented by a 300-dimensional vector of real numbers.

Metadata preprocessing: Metadata such as – Speakers, Subject, Job, State, Political party, and Context are pre-processed as categorical data for network architectures implemented using CNN or Bi-LSTMs. In the hybrid architecture (CNN + Bi-LSTM), each category of metadata is passed through a trainable embedding layer except for the history and sentiment values which are passed into a dense layer.
Categorical representation: For the speaker category from the training data, the 15 most frequently appearing speaker values are chosen. These frequent values are labelled from 0 to 14, the rest of the speaker values are clubbed under the label 15. Each value from the "Speaker" column is represented as a 15-dimensional one-hot vector. The procedure is followed for all the other metadata columns.

Part-of-Speech tagging (POS): In addition to the data provided by the dataset, we append to the metadata by performing POS tagging to the statements through the spaCy [2] library. POS tagging consists of categorizing each word with a part of speech tag (e.g. noun, verb, adjective, etc.). Words that are categorized with the same tag tend to be grammatically similar.

Dependency parsing (DEP): In addition to POS tagging we also perform Dependency parsing on the statements. Dependency parsing defines the arcs/relationship between words of a sentence. The spaCy [3] library was used for dependency parsing.

Sentiment analysis: We evaluate several models in producing sentiment results of given statements. Since we cannot train a sentiment classification with the LIAR dataset, we then choose the First 2016 GOP Presidential Debate Twitter Sentiment dataset [4] for choosing which model has the best accuracy in predicting the sentiment of the political statement as in the LIAR dataset. We mainly adopt a work by Rao [8] which compares six different models comprising TextBlob [5], Vader [6], Logistic Regression, Support Vector Machine (SVM), FastText [7] and Flair [8]. TextBlob and Vader models are based on sentiment lexicon to assign scores for each word, while the logistic regression and SVM use the TF-IDF representations as the features of the models. We also apply a SMOTE oversampling to handle the imbalance problem in the dataset. Lastly, FastText and Flair are embedding-based methods for text classification. The Flair model could also be used along with other different embeddings such as GloVe, ELMo, or BERT. In the final results, Logistic Regression, SVM, and Flair with BERT embedding come as the top three performing models with testing accuracy 0.6561, 0.6309, and 0.6128, respectively. We then choose these models to be used in the fake news classification.

3.3 CNN Architecture

The overall model layout is presented in Figure 1. The *Statement* input is a vector of integers where each integer points to a word from the vocabulary dictionary created by the Tokenizer API. This vector is passed on to the embedding layer, which consists of an embedding matrix. The embedding matrix serves as a lookup, each column of the embedding matrix is word2vec representation. The input to this layer is a list of integers denoting the column numbers of the embedding matrix. The output of the embedding layer is filtered using one-dimensional filter kernels and global pooling. Here, a multi-channel CNN is used where the same input is filtered using three different filter kernels.

[2] https://spacy.io/
[3] https://spacy.io/usage/linguistic-features
[4] https://www.kaggle.com/crowdflower/first-gop-debate-twitter-sentiment
[5] https://textblob.readthedocs.io/en/dev/
[6] https://github.com/cjhutto/vaderSentiment
[7] https://fasttext.cc/
[8] https://github.com/flairNLP/flair

This method allows the text to be processed at different resolutions [9]. *POS* and *DEP* inputs are processed similarly to the *Statement*.

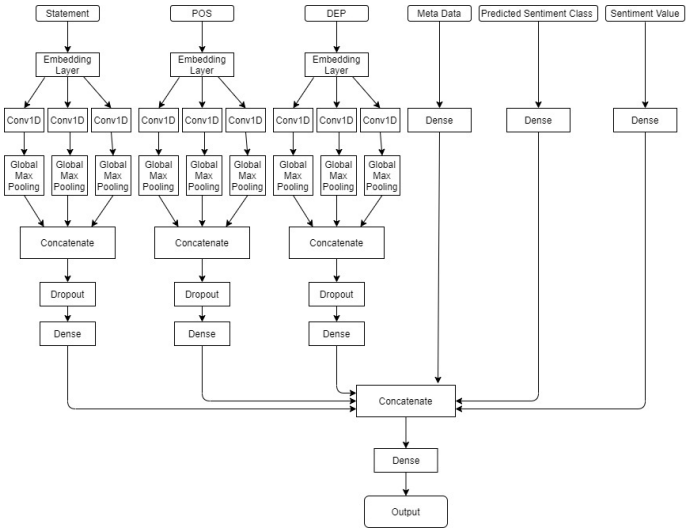

Figure 1: CNN Architecture

3.4 Bi-LSTM Architecture

Bi-LSTM architecture is explained in Figure 2. The *Statement* input is a vector of integers where each integer points to a word from the vocabulary dictionary created by the Tokenizer API. This vector is sent to the embedding layer, which consists of an embedding matrix. This embedding layer serves similarly as in the CNN architecture but the output of the embedding layer is then fed to the Bidirectional LSTM layer. *POS* and *DEP* inputs are processed similarly to the *Statement*. While, the metadata, the sentiment class and values are treated differently by sending those features directly to the dense layers. At the end of the model, all these features are combined together to classify the output.

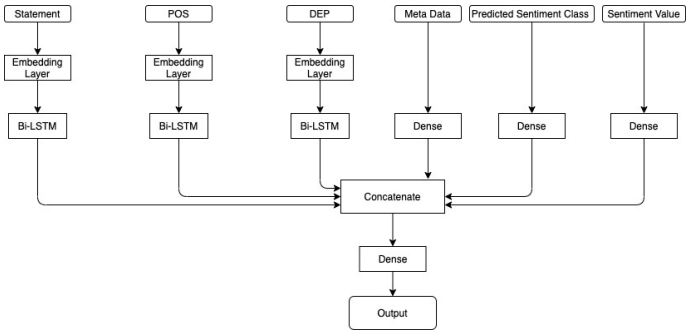

Figure 2: Bi-LSTM Architecture

3.5 Hybrid CNN-BiLSTM Architecture

The Hybrid CNN-BiLSTM [Figure 3], originally proposed in [2], consists of combining a CNN and a Bi-LSTM networks. The Hybrid CNN-Bi-LSTM architecture builds upon the CNN network by adding passing in metadata through a CNN network followed by a Bi-LSTM network. The embedded statements go through a CNN network as in the CNN architecture (c.f. above) before being max pooled. The embedded metadata goes through a CNN network followed by a Bi-LSTM, the result of which is concatenated to the global max pooling output of the CNN that processed the statement data. The concatenated layer is then passed through a dense layer with a softmax activation in order

to obtain the different classification probabilities. In this paper, the original architecture was modified by adding a dense layer where the sentiment and history features are passed through and the output of the dense layer is concatenated before the dropout and final dense layer. In our results, we opted not to use the history data in order to compare with the other architectures and the addition of the history input did not yield better accuracies.

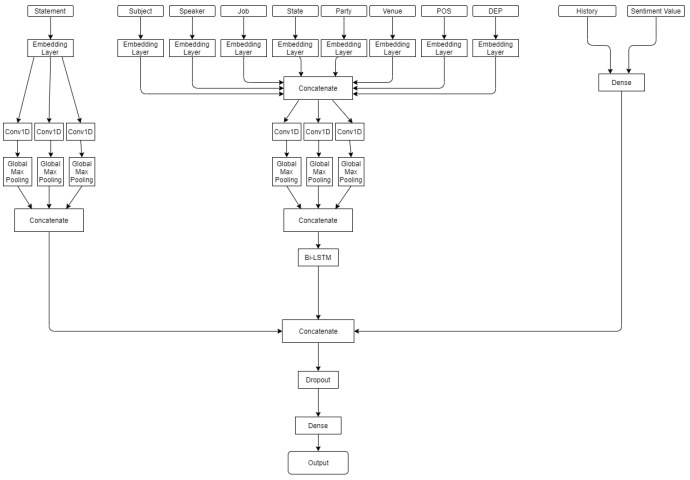

Figure 3: Hyrbid CNN-BiLSTM Architecture

4 Evaluation

For the experiments, we limit only using two different sentiment analyses by removing the sentiment results from the logistic regression since it does not give as good results as the other two models. The logistic regression model could yield 0.2027 accuracy of classifying the fake news, while the SVM and Flair models result in 0.2060 and 0.2073 accuracy, respectively.

4.1 Parameter Tuning for Baseline Model

In this step, we train baseline CNN and Bi-LSTM models using only the text of the statement as the only feature. We then perform a hyper-parameter tuning using grid search methods with a list of possible values for each parameter described in Table 1. From the parameter tuning, we choose 256 as the number of hidden unit in the CNN dense layer and Bi-LSTM layer, 0.5 as the dropout rate for both models, 7 as the CNN kernel size, 256 as the CNN filter size and 0.01 as the learning rate for the SGD optimization. The other parameters we used for training these two models are 64 hidden units on the dense layer for the metadata and sentiment features, 32 mini-batch size, and 30 epochs.
Due to time constraints hyper parameter tuning for the Hybrid model was performed using limited permutations of feature values and would ideally, more tuning would have to be done. The best values found are as follows: for the statement CNN, 128 filters with width (2, 3, 4). For the metadata CNN-BiLSTM, the number of dimensions for the metadata embedding layer is 300, the number of filters for the CNN is 20 with width (3,5,6) and the size of the BiLSTM is 6. Overall, an epoch count of 30 and the dropout probabilities of 0.5 was found to be optimal.

4.2 Results

We investigate which additional features that make significant improvement to the baseline models. In order to achieve more reliable and accurate results, we perform 5-fold cross-validation to gain the average accuracy of each model with different features and two-tailed paired t-test to compare the modified models with the baseline models. The average accuracy gained by the CNN baseline model is 0.2320, the Bi-LSTM baseline model is able to yield 0.2459 and the hybrid model is able to yield 0.2483 accuracy. When we add one additional feature, the metadata generated the highest increment in accuracy. However, we cannot conclude its contribution is statistically significant since its p-value of the paired t-test with the baseline model is higher than 0.05. When we add three

Table 1: Hyper-parameter tuning

Feature	Model		List of values	Best value
	CNN	Bi-LSTM		
Number of hidden units	✓	✓	[64, 128, 256]	256
Dropout rate	✓	✓	[0.2, 0.3, 0.4, 0.5, 0.6]	0.5
Kernel size	✓	×	[3, 5, 7]	7
Filter size	✓	×	[64, 128, 256]	256
Learning rate	✓	✓	[0.01, 0.001, 0.0001]	0.01

Table 2: Features Contribution to the Fake News Classifier

Model	Features	Avg. CV accuracy			p-value		
		CNN	Bi-LSTM	Hybrid	CNN	Bi-LSTM	Hybrid
Baseline	Statement	0.2320	0.2459	0.2483	-	-	-
Model 1	Statement + DEP	0.2440	0.2431	0.2436	0.1442	0.6195	0.1156
Model 2	Statement + POS	0.2387	0.2448	0.2468	0.3487	0.7729	0.6259
Model 3	Statement + Meta	0.2453	0.2550	0.2478	0.0883	0.0501	0.8725
Model 4	Statement + Sentiment by SVM	0.2409	0.2434	0.2422	0.2785	0.5509	0.1146
Model 5	Statement + Sentiment by Flair	0.2418	0.2444	0.2470	0.2395	0.7199	0.6816
Model 6	Statement + DEP + POS	0.2485	0.2380	0.2407	0.0725	0.1235	0.0718
Model 7	Statement + DEP + POS + Meta	0.2489	0.2580	0.2437	0.0659	0.0594	0.3211
Model 8	Statement + DEP + POS + Meta + Sentiment by SVM	0.2505	0.2556	0.2417	**0.0410**	0.0567	0.1045
Model 9	Statement + DEP + POS + Meta + Sentiment by Flair	**0.2619**	**0.2586**	0.2466	**0.0028**	**0.0307**	0.7194

additional features (DEP, POS, and metadata), Model 7, the average accuracy increases much yet still statistically insignificant. Only when we add all features (DEP, POS, metadata, and sentiment) to the Model 9, the conclusion of the modified model gives better performance than the baseline model is statistically significant in which its p-values are less than 0.05. However, in the Bi-LSTM model, only the sentiment result from Flair model could produce significant improvement, while the sentiment result from SVM model is failed. Moreover, it could be concluded that the CNN model slightly outperforms the Bi-LSTM for the Model 9. In this case, our hybrid CNN-BiLSTM model cannot give better accuracy when more features are added to the baseline model. The full results are provided in Table 2. Furthermore, Figure 4 show visually how the distribution of the results differ to the baseline models.

5 Conclusions

In this project report, we tested three different model architectures and have compared their performance given different input features. It is concluded that additional features: POS, DEP, metadata, and sentiment of the statement could significantly increase the performance of the baseline fake news classifiers with using only the statement.

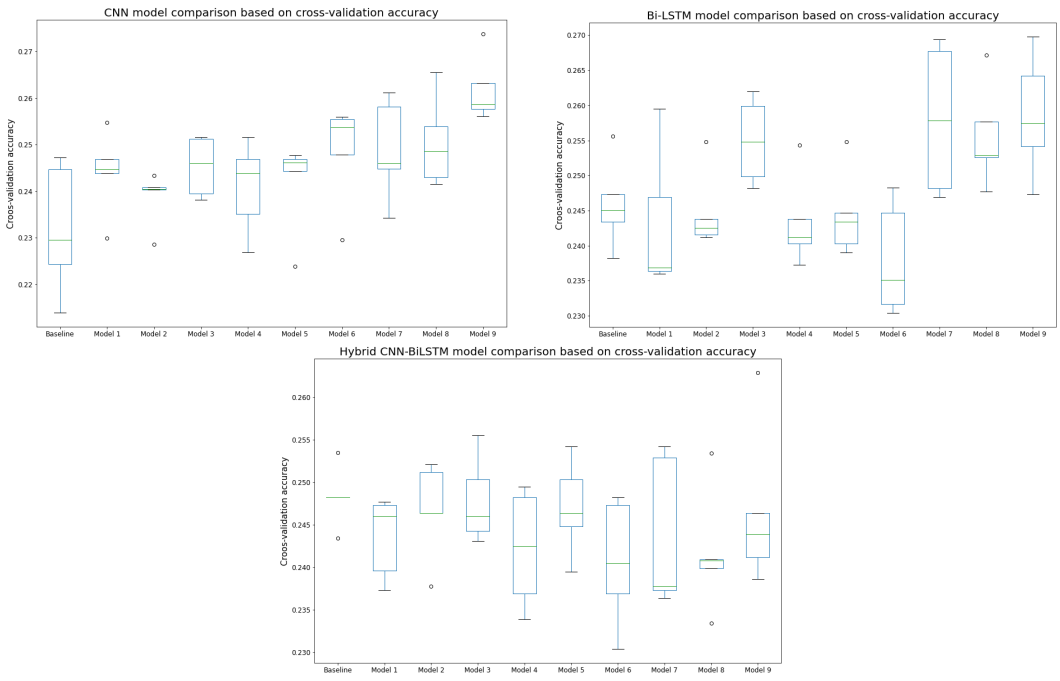

Figure 4: CNN (top-left), Bi-LSTM (top-right) and Hybrid CNN-BiLSTM (bottom) results

References

[1] Adam Burston, John Cano Barrios, Daniel Gomez, Ingmar Sturm, Anagha of Uppal, Yoori of Yang, and Joseph B. Walther. A Citizen's Guide to Fake News | Center for Information Technology and Society - UC Santa Barbara. URL `https://www.cits.ucsb.edu/fake-news`.

[2] William Yang Wang. " liar, liar pants on fire": A new benchmark dataset for fake news detection. *arXiv preprint arXiv:1705.00648*, 2017.

[3] Samir Bajaj. The pope has a new baby! fake news detection using deep learning, 2017.

[4] Natali Ruchansky, Sungyong Seo, and Yan Liu. Csi: A hybrid deep model for fake news detection. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 797–806, 2017.

[5] O. Ajao, D. Bhowmik, and S. Zargari. Sentiment aware fake news detection on online social networks. In *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2507–2511, 2019.

[6] 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.

[7] Debjyoti Roy, Priyanka Bedarkar, and Siddarth Harinarayanan. NLP-Fake-News_detection: Implementation of a deep learning model that does fake news classification based on Liar-Liar Dataset, 2019. URL `https://github.com/siddarthhari95/NLP-Fake-News{_}detection`.

[8] Prashanth Rao. fine-grained-sentiment: A comparison and discussion of different NLP methods for 5-class sentiment classification on the SST-5 dataset., 2019. URL `https://github.com/prrao87/fine-grained-sentiment`.

[9] Yoon Kim. Convolutional neural networks for sentence classification. *CoRR*, abs/1408.5882, 2014. URL `http://arxiv.org/abs/1408.5882`.