CS230

DeepNewsNet: Automated Fake News Classification

Alaukik Aggarwal
Stanford University
alaukika@stanford.edu

Abstract

Fake news has directly or indirectly affected everyone. It has been at core of several instances of inciting mob, causing riots, influencing elections, picking our leaders, and causing failure of justice. This work explores the use of deep learning algorithms to tackle the problem of fake news. Starting with a shallow fully-connected neural network implementation as a baseline model, impact of different models, vectorization process, and annotation with new features are evaluated. We demonstrate that sentiments and statement's context haven significant impact on performance of fake news prediction. The final model performs significantly better than the existing benchmark on the training dataset.

1 Introduction

Fake news is one of the most challenging problems faced by human society today. There are several facets to tackling this problem.

One of the most challenging aspect of this is the lack of structured information. While web, social networks, and messaging applications provide platform for free speech, they are recently being leveraged by malicious actors for lack of need to provide evidence of truth. Amount of data poses another big hurdle in stopping the dissemination of such information. It is no longer feasible to use manual or heuristic based approaches to detect it. In addition, these platforms have increased the rate of spread of information by manifolds. By the time a fake news article is identified, the damage has already been done.

These issues are compounded by the fact that author's intent might be conveyed in some form, which validates the content of articles [1]. Example, the intent might be to present hoax, satirical, or humorous by presenting factually incorrect or outlandish claims in the article. Another aspect of fake news is intentional withholding of information in order to propagate the misbelief [2]. This project explores the scenario where article contains factually incorrect information, independent of the intent of the author.

The input to the model consists of short political statement from news articles, along with metadata like speaker and location. The length of input text can vary based on the article, and metadata is different based on the article. Example, party affiliation might not be available for all speakers. The model predicts whether the statement is true, partially fake, or completely fake.

2 Related Work

There have been attempts to tackle these challenges from various angles. Both regression and classification approaches have been used to address issue of fake news. There have been different approaches based on type of claim, source of data, metadata attached, media, and source of label annotation [4].

Approaches involve use of TF-IDF [7] or word embeddings [6] to represent input text. For classification, there have been context-aware approaches [5] where the emphasis is on the context like location or subject of the statement. Retrieving or annotating context for statements can become an issue for new or unseen statements. Evidence [6], in form of other news articles, has also been used a way for fact checking. One of the biggest challenges of this approach is the performance is directly linked to the number of articles scanned and trained to be linked to claims. There have been attempts to outsource this responsibility to the agency publishing the article [8]. However, there has not been significant work in exploration of direct impact of use of sentiments as input to detection of fake news [11].

3 Dataset and Features

In this project, readily available public dataset has been used for training and benchmarking: LIAR dataset [3]. It comprises of 12,836 short statements collected from PolitiFact [17]. Each statement is labeled as: true, mostly-true, half-true, barely-true, false, and pants-on-fire. It has following attributes:

- 1: the ID of the statement
- 2: the label
- 3: the statement
- 4: the subject(s)
- 5: the speaker
- 6: the speaker's job title
- 7: the state info
- 8: the party affiliation
- 9-13: the total credit history count, including the current statement - barely true counts, false counts, half true counts, mostly true counts, pants on fire counts.
- 14: the context (venue / location of the speech or statement)

The dataset is randomly split between train, validation, and test dataset, comprising of 10238, 1284, and 1283 samples (80 - 10 - 10). Except for 'pants-on-fire' label (~10%), dataset is evenly split between other label types (17-19% for each label type). This is true for each dataset split.

4 Methods

There are two main parts of the problem – feature extraction and model architecture (including hyperparameter tuning).

4.1 Feature Extraction

Input consists of the statement, the subject, the speaker, the speaker's title, the state info, the party affiliation, and the context. A combination of different input feature sets have been tried:

- Statement + All text attributes
- Statement + All text attributes + Sentiment feature
- Statement + All text attributes + Sentiment feature + All non-text attributes

We have tried two ways to vectorize text – using term frequency - inverse document frequency (TF-IDF) and Global Vectors (GloVe) for word representation [12]. Scikit-learn [9] was used for tokenization and vectorization of text, while GloVe word embeddings pretrained on Wikipedia 2014 and Gigaword 5 were used to convert each word into 100-dimensional vector.

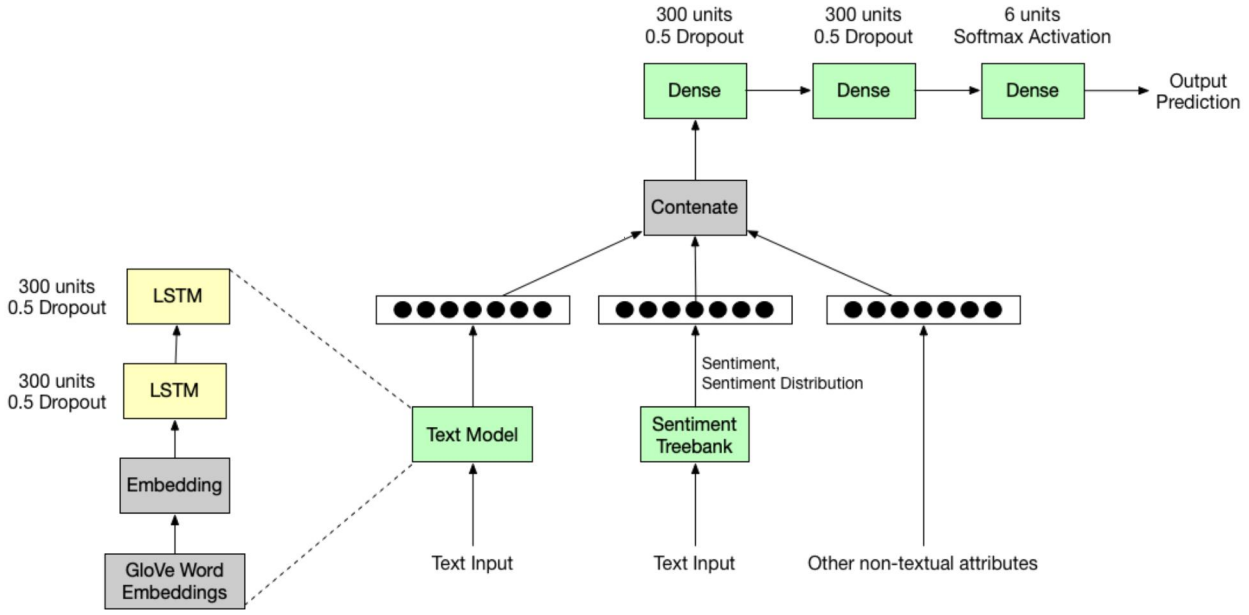
In addition, we also have used sentiment distribution predictions from Sentiment Treebank [18]. Distribution of speaker's history of statements is taken as another feature input.

4.2 Architecture

A fully-connected neural network is used as a baseline model, having three fully-connected layers of 300, 300, and 6 hidden units each. TF-IDF is used for vectorization of bag of words. To be able to perform a clear comparison with the performance achieved by the authors of LIAR dataset [3], 'accuracy' has been used as the metric for measuring models' performance.

After implementing the baseline model, different models are explored:

- Deeper fully-connected neural network having nine fully-connected layers of 300, 300, 300, 200, 200, 200, 100, 100, and 6 hidden units.
- LSTM [16] model having two LSTM layers having 300 hidden units with Dropout, followed by a dense layer having 6 hidden units.
- Hybrid architecture – combines LSTM model for representing text features and combine it with non-textual attributes like sentiment distribution and speaker history using fully-connected layers.



4.3 Training Approach

Being a multiclass classification task, the models use categorical cross-entropy as loss function [13].

$$L_i = -\sum_{j=1}^{k} t_{i,j} \log(p_{i,j})$$

All models use softmax as the final activation function. For regularization of the system, we used L2 for fully-connected models and Dropout for LSTM and Hybrid models. Training was performed using mini-batch gradient descent with Adam optimizer.

All the models have been built and trained using tensorflow [10] and Keras [14]. The models were strictly tuned using validation set. Early stopping was used based on plateauing of loss function and stability of accuracy. After training, test dataset was used only for final performance evaluation.

5 Experiments

5.1 Hyperparameter tuning

The set of hyperparameters tuned in different models, range of values considered, and the final optimal values used are presented in Table 1. Depending on the parameter, the optimal values were searched through some combination of exponential, arithmetic, and heuristic based search in the range of values. In some cases, like number of layers and dimensions of word embeddings, hardware restrictions also forced the model to not able to try bigger values. With very high values, factors like training time, memory issues, and parameter tuning become much more challenging. We talk about some of these in *Lessons Learnt* section below.

Table 1

Hyperparameter Name	Range	Optimal Value
Learning Rate	0.1 – 0.0001	0.01
Epoch Count	10 - 300	50
LSTM Hidden Units	100 - 500	300
L2 Regularization Parameter	0.05 – 0.5	0.1
Dropout Rate	0.2 – 0.8	0.5
Mini-batch size	50 - 500	200
Loss weights	0.2 – 1.	1., 1.

Loss weights are the weights of contributions considered from final layer output and intermediate LSTM layers in Hybrid model, while minimizing the loss value during training.

5.2 Model Evaluation

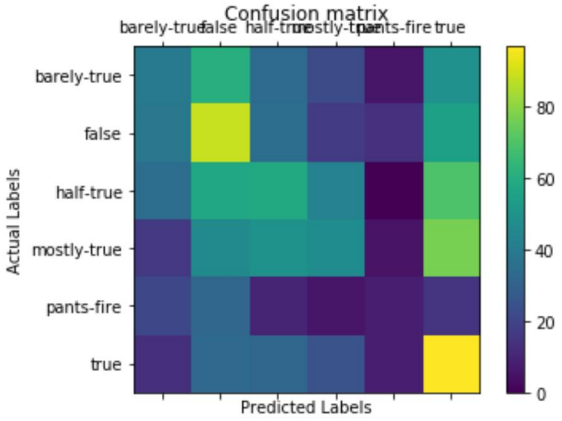
We have used accuracy as the metric to compare performance of our model with the authors of LIAR dataset. The performance for the best combination of input for each model type is presented in Table 2.

Table 2

Model	Features	Accuracy
Existing LIAR Dataset Benchmark (CNN)	Text + non-text attributes	0.274
Shallow fully-connected (baseline)	Text (TF-IDF) only	0.25
Deep fully-connected	Text (TF-IDF) only	0.252
Deep fully-connected	Text (GloVe) only	0.258
LSTM	Text (GloVe) only	0.269
Hybrid	Text (GloVe) + Sentiment	0.294
Hybrid	Text (GloVe) + Sentiment + non-text attributes	0.402

The non-text attributes here represent histogram of count of statements for each label category for a given speaker.

Creating confusion matrix for the model provides with some interesting insights:



1. 'mostly-true' falling under 'true', or vice-verse: An intuitive reasoning would be that a similar error might be observed in human labeling as well. If we make this a binary classification problem of fake news or not, then misclassification might reduce.
2. 'pants-on-fire' label not categorized correctly: As discussed in *Dataset* section before, the count of labels for 'pants-on-fire' are significantly lower than other labels.
3. 'true' and 'false' are classified with high confidence, however it gets fuzzy with other labels like mostly-true and half-true. Turning this into binary classification problem, i.e. is given news article fake or not, may yield higher performance.

5.3 Other Lessons Learnt

Based on model performance, using sentiment distribution as a feature input improves the performance. However, there are couple of issues:

1. Performance of fake news detection model is tied to accuracy of sentiment analysis, which introduces noise in certain cases.
2. Generating sentiment distribution for large input data would take significant investment of time. Sentiment Treebank runs as a server that introduces issues like HTTPConnectionPool exhaustion, too many files open, high latency of 10s-100s of milliseconds for each labeling. This was addressed by caching the results.

Hardware limitations had measurable impact on hyperparameter choices like number of hidden units, size of word embeddings, and number of LSTM layers. At one point, it was taking over 5 hours on Tesla K80 GPU for a single iteration. Also, switching from Tensorflow to Keras, made it much faster to build, evaluate, compare performance, and tune parameters. It provides useful abstractions like summary, mini-batch, early-stopping, and model history.

6 Conclusion and Future Work

Comparison of different models and feature inputs has been presented. Hybrid model performs significantly better previous LIAR dataset benchmark. Improvement of model performance with addition of sentiment features highlights the aspect of strong emotions in fake news. Further, significant boost in accuracy by adding history of speeches for a given speaker show the potential of context-aware models. With more time and hardware resources, it would be interesting to explore use of higher dimension for word embeddings, using deeper networks, and constructing of a pipeline composed of different models trained separately for text and non-textual attributes.

Note: Source code can be found at https://github.com/alaukika/cs230project. An invitation for collaboration has been sent to cs230-stanford group (alt invite link).

8 References

1. Rashkin, Hannah, Eunsol Choi, Jin Yea Jang, Svitlana Volkova and Yejin Choi. *"Truth of Varying Shades: Analyzing Language in Fake News and Political Fact-Checking."* EMNLP (2017).

2. Rubin, Victoria L., Yimin Chen and Niall Conroy. *"Deception detection for news: Three types of fakes."* ASIST (2015).

3. Wang, William Yang. *""Liar, Liar Pants on Fire": A New Benchmark Dataset for Fake News Detection."* ACL (2017).

4. Oshikawa, Ray, Qian Jing, William Yang Wang. *"A survey on natural language processing for fake news detection"*, CoRR abs/1811.00770 (2018).
5. Gao, Lei and Ruihong Huang. *"Detecting Online Hate Speech Using Context Aware Models."* RANLP (2017).

6. Popat, Kashyap, Subhabrata Mukherjee, Andrew Yates and Gerhard Weikum. *"DeClarE: Debunking Fake News and False Claims using Evidence-Aware Deep Learning."* EMNLP (2018).

7. Riedel, Benjamin, Isabelle Augenstein, Georgios P. Spithourakis and Sebastian Riedel. *"A simple but tough-to-beat baseline for the Fake News Challenge stance detection task."* CoRR abs/1707.03264 (2017).

8. Google approach to addressing fake news problem: *https://developers.google.com/search/docs/data-types/factcheck.*

9. F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Pret-tenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, andM. P. E. Duchesnay. *"Scikit-learn: Machine learning in Python."* Journal of Machine LearningResearch (2011).

10. Tensorflow: *https://www.tensorflow.org/*.

11. Conroy, Niall J., Victoria L. Rubin, and Yimin Chen. *"Automatic deception detection: Methods for finding fake news."* Proceedings of the 78th ASIS&T Annual Meeting: Information Science with Impact: Research in and for the Community. American Society for Information Science (2015).

12. Jeffrey Pennington, Richard Socher, and Christopher D. Manning. *"GloVe: Global Vectors for Word Representation*." Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP) (2014).

13. Gulli, Antonio, and Sujit Pal. *"Deep Learning with Keras."* Birmingham: Packt Publishing (2017).

14. Keras: *https://keras.io/.*

15. Manning, Christopher D., Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. "*The Stanford CoreNLP Natural Language Processing Toolkit."* In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations, pp. 55-60 (2014).

16. Sepp Hochreiter and J¨urgen Schmidhuber. *"Long short-term memory."* Neural computation9(8):1735–1780 (1997).

17. PolitiFact API: *http://static.politifact.com/api/v2apidoc.html.*

18. Socher, Richard, et al. *"Recursive deep models for semantic compositionality over a sentiment treebank."* Proceedings of the 2013 conference on empirical methods in natural language processing (2013).