Czech Technical University in Prague

Faculty of Information Technology

Fake News Detection
With the Use of Machine Learning Techniques

Anastasiia Solomiia Hrytsyna

December 31, 2022

B221 - Winter semester 2022/2023

The Main Task

The main goal of this project is to understand the concept of real and fake news. For this some analysis of each news category has to be made (a comparison of title or text length, average word length, most common words, publication category or date etc.). Afterwards, we used a few different classification methods (LSTM, BERT, Logistic Regression, Random Forest, Naive Bayes Classifier) to recognize fake and real news according to its content. Effectiveness of all methods were compared and the results are displayed below.

Working With The Data

For our examination the dataset from Kaggle on this link was used. It contains 2 '.csv' tables for real (53.58 MB) and fake (62.79 MB) english news. Each of these tables has 5 columns: title, text, subject and publication date of the article (I also added our target column that marks 1 for real news and 0 for the fake one). The whole dataset of news has approximately 44.9 thousand entries.

During the analysis I found out that real and fake news contain approximately the same amount of entries (see Fig. 1).

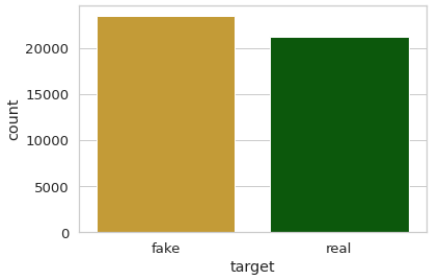

Fig. 1: Class distribution of real and fake news

The dataset also contains at about 209 duplicated rows that were dropped. I also divided the date column into 3 separate columns (year, month and day) - the last two of them were encoded with sin or cos function, but have never been used during classification. Additionally, 10 rows contain https links (not date), so we have also deleted them. Class distribution depending on the publication date is shown on Fig. 2.

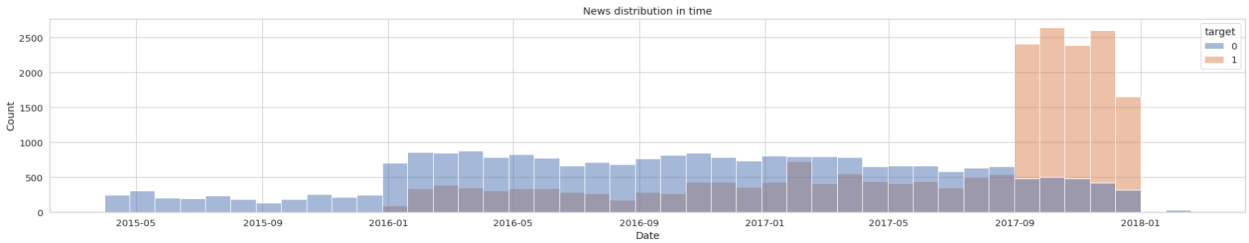

Fig. 2: Class distribution of real and fake news in time

After that I preprocessed article content: combined title and main text, removed stop words, links and punctuation, changed some of the short word forms to the long ones (for example 'I'm' to 'I am') and so on. Some parts of content analysis are shown below.

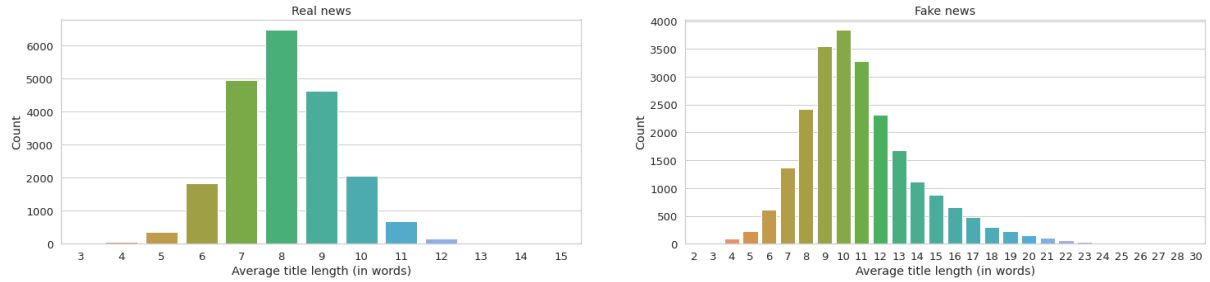

Fig. 3: Average title length distribution depending on the class

As we may notice, in most cases fake news contains longer titles and text, using more complicated words. In addition, I checked the most popular common words for both real and fake classes (see Fig. 4) and analyzed 3-grams for each class (see Fig. 5). All of the additional graphs and information may be found in the notebook[1].

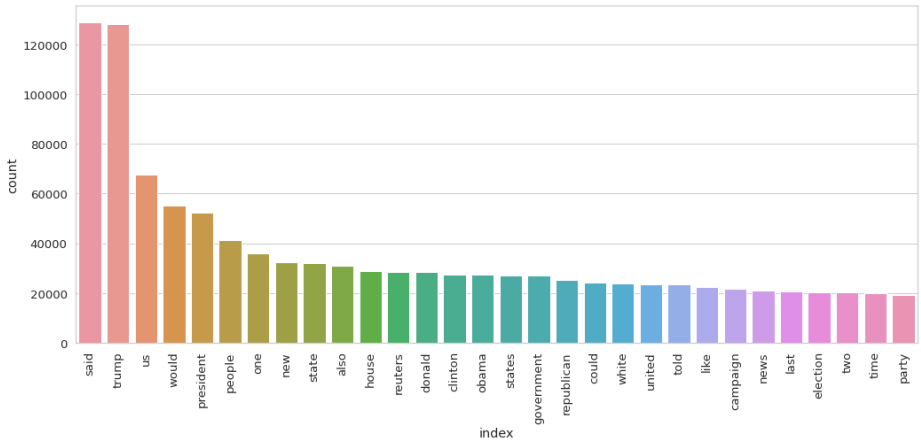

Fig. 4: Common words distribution in both classes

After the EDA part and small preprocessing I tokenized my data (divided the whole news text into separate parts - tokens) and performed word embedding. I used a few different methods for this (see section **Methods**) depending on the classification model.

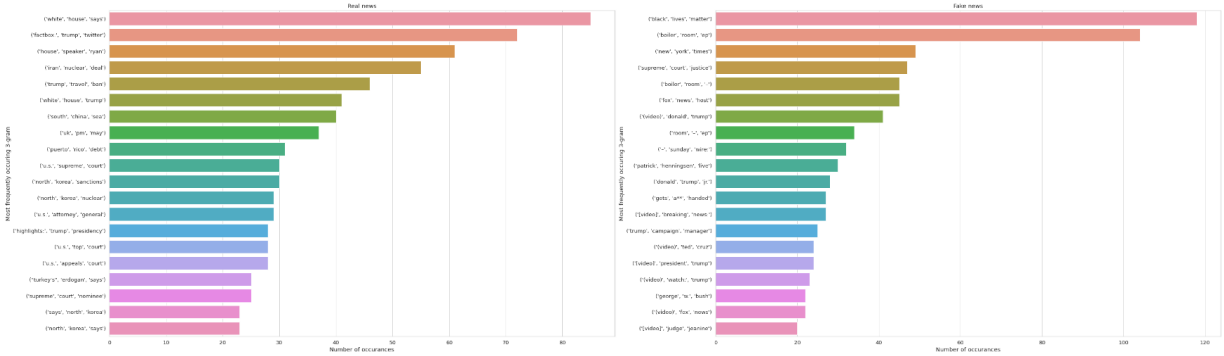

Fig. 5: Most frequently occuring 3-gram depending on the class

[1] Link to the notebook: *https://gitlab.fit.cvut.cz/hrytsana/semestral-project-ni-mvi/-/blob/master/program.ipynb*

Methods

As I have already mentioned in my first report - for the classification I used 3 different methods:

1. Logistic Regression, Random Forest and Naive Bayes Classifier

In this part I performed GridSearch for the best parameters tuning for each classification. Before that I preprocessed news content with TfidfVectorizer and converted it to the vectors. Unfortunately, later due to the low processing capabilities I was made to smaller my database up to 3000 samples. That is why obtained results (see section **Results**) may not be precise.

2. Bidirectional Long-Short Term Memory (LSTM)

Next I performed classification with the LSTM model (implemented from scratch). It consists of one embedding layer (which is trained simultaneously with the whole NN model), two bidirectional layers, max-pooling layer, and a set of 5 dense layers with Relu activation functions + dropout. Finally, I added the last (sixth) dense layer with the Sigmoid activation function, cross-entropy loss function and Adam optimizer (see NN model in the notebook) to control model performance. In addition, I have implemented early stopping (in case validation accuracy does not improve in more than 4 epochs) and vectorized my input text with tf.keras.preprocessing.text.Tokenizer.

3. Bidirectional Encoder Representations from Transformers (BERT)

In the last part I used transfer learning with the BERT model and BertTokanizer of sequences. I also used a custom data loader for labeled input data. Learning rate was set to 0.01, as an optimizer I chose an Adam one, but the same as in the first method I was made to reduce the number of input samples, so I can perform my experiment. The whole model I trained at about 10 epochs. Since training and validation loss was constant I performed evaluation.

Results

1. Logistic Regression, Random Forest and Naive Bayes Classifier

All of the classificators showed similar metrics results: accuracy, precision, recall and F1-score at about 50%. Confusion matrices (see Fig.6), ROC and Precision-Recall curves don't show promising results. The best performance was with Random Forest classification (see tuned parameters in the notebook) and gained results of 51% for accuracy, 67% for precision, 0.68 for recall and 1.34 for F1-score. Obviously, these results may not be considered as successful ones, but they may be slightly improved by training on the bigger dataset.

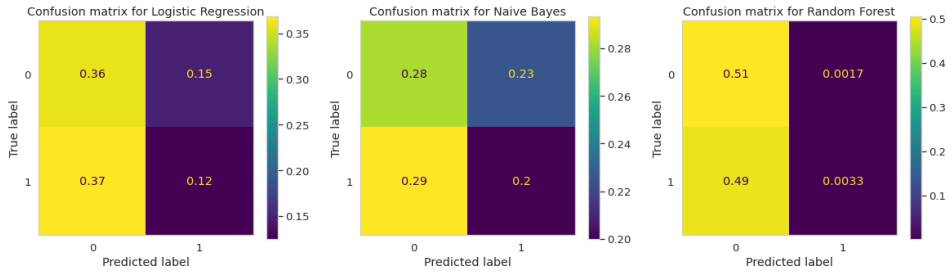

Fig. 6: Confusion matrices for each classificator

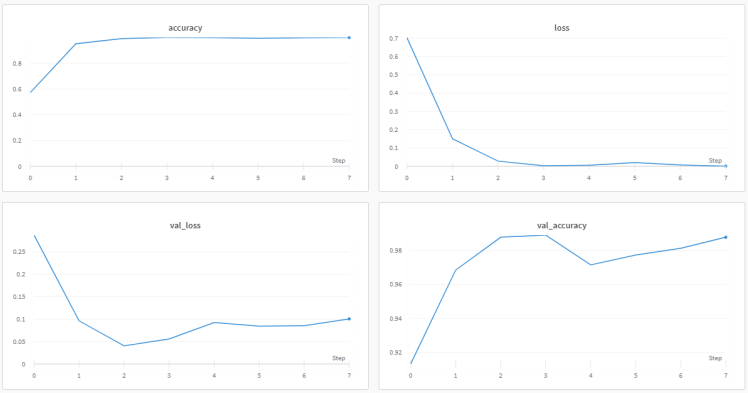

Fig. 7: LSTM training progress

2. Bidirectional Long-Short Term Memory (LSTM)

The results for the training phase are shown on Fig.7 The whole training was stopped after 7 epochs with early stopping. For performance evaluation I used a trained model on its second epoch (while validation loss does not grow and model is not pretrained). Final results seems to be promising: 98% for accuracy, 97% for precision, 98% for recall and 98% for F1-score.

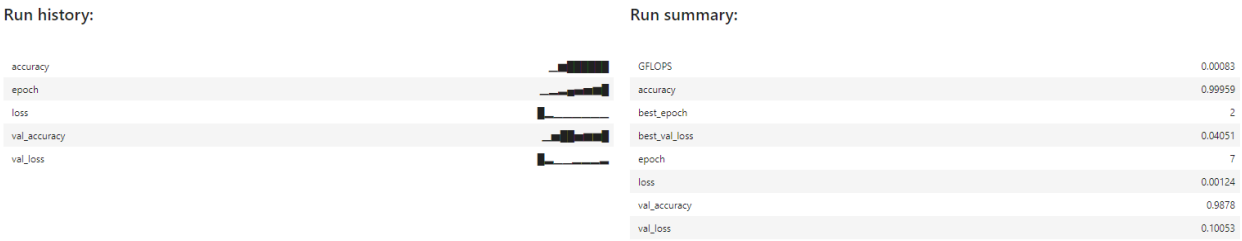

Fig. 8: LSTM training summary

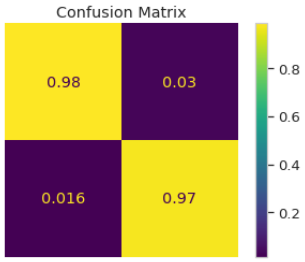

Fig. 9: Confusion matrix (in %) for LSTM model

3. Bidirectional Encoder Representations from Transformers (BERT)

The last method achieved results of 52% for accuracy, 52% for precision, 100% for recall and 68% for F1-score. Confusion matrix shows us that during evaluation we do not use fake text for classification, just the real news, which were classified almost with 50/50% chance. Such a bad output may be explained by the low variety of unbalanced input samples. To better model performance we should train it with more data, but it is time consuming, as the NN model is deeper than the previous one.

Conclusion

After performing all of my experiments I may say that the LSTM model was the most successful one (accuracy - 97%) - I would definitely choose this one for a future improvement. Even with the fact that obtained results are slightly invalid (as each model was trained with different input data), we may notice that the selected model gains its results much faster than the second NN model and is more precise than usual classificators (even with parameters tuning).

In the future it would be great to use even more balanced input data and train selected model longer with the smaller learning rate. Also, it would be beneficial to use other columns (such as subject or date) in our classification.

References

[1] https://medium.com/swlh/fake-news-detection-using-machine-learning-69ff9050351f
[2] https://towardsdatascience.com/fake-news-detector-with-deep-learning-approach-part-i-eda-757f5c052
[3] https://realpython.com/python-keras-text-classification/#what-is-a-word-embedding
[4] https://www.geeksforgeeks.org/fake-news-detection-using-machine-learning/
[5] https://towardsdatascience.com/text-classification-with-pytorch-7111dae111a6
[6] https://towardsdatascience.com/lstm-text-classification-using-pytorch-2c6c657f8fc0
[7] https://jalammar.github.io/a-visual-guide-to-using-bert-for-the-first-time/
[8] https://towardsdatascience.com/bert-text-classification-using-pytorch-723dfb8b6b5b
[9] https://data-flair.training/blogs/advanced-python-project-detecting-fake-news/
[10] https://www.analyticsvidhya.com/blog/2020/07/transfer-learning-for-nlp-fine-tuning-bert-for-text-classification/
[11] https://towardsdatascience.com/fake-news-detection-with-machine-learning-using-python-3347d9899ad1
[12] https://pytorch.org/tutorials/beginner/text_sentiment_ngrams_tutorial.html
[13] https://coderzcolumn.com/tutorials/artificial-intelligence/pytorch-simple-guide-to-text-classification
[14] https://www.kaggle.com/code/ishandutta/ag-news-classification-lstm
[15] https://developer.nvidia.com/blog/three-approaches-to-encoding-time-information-as-features-for-ml-models/
[16] https://www.kaggle.com/code/ishandutta/ag-news-classification-lstm
[17] https://iq.opengenus.org/binary-text-classification-bert/?fbclid=IwAR0W7RkUckONBfYEowEyilWV9eojCf-zmt4yzNQyOQlyaKK2zwfkCd8W5Lo