UAICS at CheckThat! 2021: Fake news detection

Ciprian G. Cusmuliuc, Matei A. Amarandei, Ioana Pelin, Vlad I. Cociorva and Adrian Iftene

"Alexandru Ioan Cuza" University, Faculty of Computer Science, Iasi, Romania

Abstract

Social media growth in recent years has facilitated an enhancement in human communication. Platforms such as Facebook and Twitter are now ever-present in our lives, influencing how we speak, think and act. The growth of fake news greatly impacts this phenomenon as it lowers one's trust in the content presented. One such example is related to the 2016 U.S. presidential election campaign where fake news was a deciding factor in tipping the balance of power. It is hence of critical importance to develop tools that detect and combat such destructive content. CLEF 2021 CheckThat! Task 3 tries to address the problem of fake news, posing a challenge to develop systems that could detect if the main claim made in an article is true, partially true, false, or other. Our team participated in this task with 5 models, ranking 6th place with an F1-macro of 0.44 and a model based on Gradient Boosting; in this paper we will present our methods, runs and results but also discuss future work.

Keywords

Fake news detection, LSTM, Bi-LSTM, BERT, RoBERTa, Random Forest, Gradient Boosting, Naïve Bayes, KNN.

1. Introduction

Recent advances in computing, that date at the beginning of the millennium, have drastically changed human interaction, people no longer tend to meet in real life to maintain contact with friends; furthermore, the COVID-19 pandemic has accelerated this movement by forcing everybody to dialogue via digital means for months at a time. The main facilitators of this movement are social media platforms, that have seen massive usage spikes in the past decade, radically changing how we speak, read news, watch videos and so on, this freedom however comes at a cost. Allowing everybody almost unlimited reachability and free hand to post however they please is a big advantage, but it is also very dangerous; the classical example is related to the 2016 U.S. presidential election campaign where a mixture of social profiling and fake news have led to surprising electoral results (this result contrasts with the 2020 U.S. elections where social media platforms have banned many ads[2]). Considering the previous argument, it is obvious that we need automated methods that analyze the posts and flag them for fake or misleading content.

CLEF CheckThat! 2021 Task 3a [1] [2] [17] [18] has exactly the goal expressed in the previous section; the task definition been that: "given the text of a news article, determine whether the main claim made in the article is true, partially true, false, or other (e.g., claims in dispute) and also detect the topical domain of the article". In the competition we submitted 5 different models and overall ranked 6th.

This paper describes the participation of team UAICS, from the Faculty of Computer Science, "Alexandru Ioan Cuza" University of Iasi, in Task 3a at CLEF 2021. The remaining of this paper was organized as follows: Section 2 details the models we developed and the submitted runs and then

[1]CLEF 2021 – Conference and Labs of the Evaluation Forum, September 21–24, 2021, Bucharest, Romania
EMAIL: gabriel.cusmuliuc@info.uaic.ro (Ciprian G. Cusmuliuc)
ORCID: 0000-0003-0758-3061 (Ciprian G. Cusmuliuc)


[2] https://www.bbc.com/news/technology-54369303

Section 3 details the results we obtained and finally Section 4 concludes this paper and presents future work.

2. Methods and runs

In this section we will detail the submitted models; 5 models have been developed in the search of finding the best one, we relied on state-of-the-art methods such as LSTM, Bi-LSTM, BERT, RoBERTa but also experimented with a few novel methods based on more traditional techniques such as Gradient Boosting, Naïve Bayes, KNN and Random Forest. In future sections we will take a look at state-of-the-art techniques, analyze the dataset as well as discuss our models and preprocessing.

2.1. State of the art

Research interest in fake news classification has grown exponentially in just a few years. Identification efforts have been very diverse but they all can be summarized in 3 big categories as [3] outlines: *creator and user analysis*, *social context analysis and news content analysis*.

Creator and user analysis focuses on extensive analysis of user accounts in order to identify malicious behaviors. Malicious user accounts behave differently from authentic users; thus, identification is possible. Different user categorization can be achieved using different techniques: *user profiling analysis* [4][5], *temporal and posting behavior analysis* [6], *credibility-related analysis* [7], *and sentiment-related analysis* [8]. Considering user information was not available in the CheckThat! dataset, these techniques would not have been possible to apply.

Social context analysis tries to study how the news disseminates in the social environment, meaning how quick and wide the data is share/distributed and how users interact with each other, having 2 big research areas: *user network analysis* (users with high interaction with the news creator can be used to predict the truthfulness of the news) [9] and *distribution pattern analysis* (analysis of the information spread in the network) [10]. Just like *creator and user analysis*, *social analysis* is not feasible on this task, not to mention that this technique is not used often. Many approaches choose to analyze the news itself.

News content analysis in contrast to *creator and user analysis* does not focus on *who* posts rather on *what* they post. In [11] they used a multitude of neural networks in combination with GloVE embedding to predict the label of a news article; the best result was with a Bi-LSTM, accuracy of 0.91, but notable results were obtained with CNN (0.90) and vanilla RNN (0.78). [12] takes a different approach based on machine learning, implying Naïve Bayes, Gradient Boosting and Random Forest in order to identify a series of 10000 tweets collected in August 2012, concluding that Random Forest is the best algorithm with an accuracy of 96%. Finally [13] uses the most novel techniques at this time, BERT [14]; they start off by tokenizing the input string, then padding after which feeding it to a pre-trained large cased BERT model to perform the classification which yields an accuracy of 0.69 on a test dataset.

Knowing thus what the best models are but also what their limitations were we proceeded with training them in order to see a result.

2.2. Training and test dataset analysis

The training and test dataset have been provided by the organizers and examples can be seen in Table 1 and 2. The training dataset consisted of 945 labeled articles and the test dataset had 365 unlabeled articles. This small number of articles proved to be a disadvantage to neural network models as we did not use any other additional datasets.

Table 1
Training dataset example

public_id	text	title	our rating
c7ea6a6e	New evidence ties COVID-19 creation to research funded by Fauci?	Flooding of Coast, Caused by Global Warming, Has Already Begun.	False

Table 2
Training dataset example

public_id	text	title
58bea1db	Second patient cured of HIV, say doctors.	Lisa Page Squeals: DNC Server Was Not Hacked By Russia.

In Figure 1 a dataset analysis is done; taking the Task 3a batches we plot them in order to gain some insight in the collection. In the left size of the figure a word cloud view of most frequent words in the dataset has been build, with the biggest topics being related to politics and COVID-19. The right part of the figure also confirms the latter assumption as there we can see the most frequent words, such as "trump", "covid19" and so on (the plots have been done with tokenized data).

A problem that was identified early on and will greatly impact the results is relat-ed to label imbalance. Figure 2 shows in different representations how many articles are available with a certain label, unfortunately since False is the most common one, automatically the algorithms will be biased in that direction (0-False, 1-Other, 2-Partially False, 3-True).

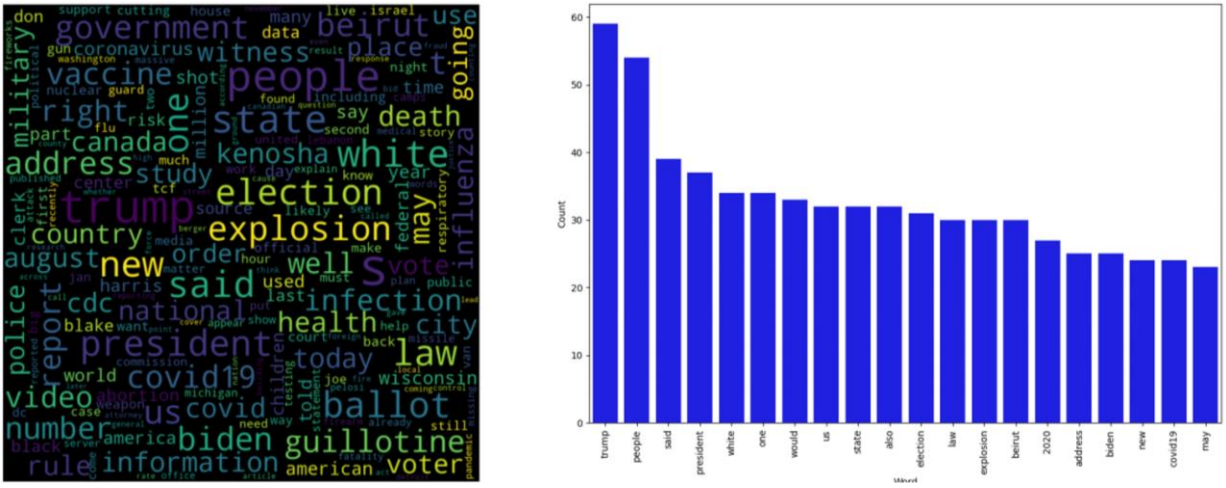

Figure 1: Left – word cloud view of most frequent words in the dataset; Right – Bar Plot of most frequent words in the dataset.

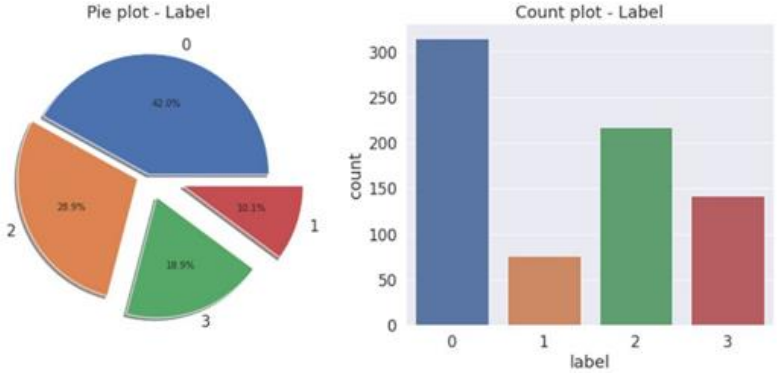

Figure 2: Label distribution in the dataset (0 - False, 1 - Other, 2 - Partially False, 3 - True).

2.3. Models

2.3.1. 3Layer Model

The first model, and the one which proven to be the most performant, has been named "*3Layer Model*" because of its use of 3 different preprocessing methods and 3 different Machine Learning algorithms used.

In the data preparation phase, there have been a series of alterations over the dataset. The *public_id* field has been removed, the two training batches have been combined as well as the *title* field and *text*, punctuation signs have been removed as well as stop-words, dashed and underscores and lastly the text has been lowercased and lemmatized.

The feature extraction phase consisted of three approaches:

- ***Clean text*** is a bigram (a contiguous sequence of n items, where n is 2), the training column will be called ***clean_text***;
- ***POS Tagging*** on *text* column using spaCY[3] to obtain the POS form), the training column will be called ***POS_text***;
- ***Semantic Analysis*** is done using Stanford's Empath Tool[4] [15] to categorize the words in the articles by their lexicon and approximate which articles that are fake predominantly use a certain lexicon (this column was named ***semantics_text***). An example can be seen in *appendix A.*

Besides the three aforementioned techniques we created a fourth one by weighting them as follows: clean_text: 0.5, POST_tagging: 0.15 and semantic_text: 0.35 (these values have been determined experimentally).

On the columns mentioned earlier, clean_text, POS_text and semantics_text, in order to feed the data to the M.L. algorithms we applied TF-IDF.

As for the models used, they consisted of Naïve Bayes, KNN, Random Forest and Gradient Boosting. In the results section we will discuss the hyperparameter tuning in relationship to the result; in the end the most performant variant consisted of Gradient Boosting combined with the weighted representation of clean text, POS tagging and semantic analysis.

2.3.2. BERT

Another model developed is based on BERT which yielded great results in many state of the art systems [13].

Data preparations for this method consisted of shuffling the training articles, concatenation of the batches, merging the title and text columns and eliminating *public_id* (it was redundant to training). Other operations have consisted of punctuation signs removal, lemmatization, mandatory text padding and a special BERT tokenizing process.

As for the model, we used *bert-large-uncased* (24-layer, 1024 hidden dimensions, 16 attention heads, 336M parameters) from HuggingFace[5] and begun the fine-tuning process. A problem immediately apparent was the size of the dataset as BERT requires many training entities. We used AdamW Optimizer (fine-tuned the learning rate as well as possible, 6e-6 yielding the best results), 3 epochs and a batch size of 3.

Figure 2 presents the Training and Validation loss over the epochs; training set contained 70% of the data, 20% for testing and 10% for validation. Apendix B shows a snippet of the BERT classifier.

[3] https://spacy.io/
[4] https://github.com/Ejhfast/empath-client
[5] https://huggingface.co/transformers/model_doc/bert.html

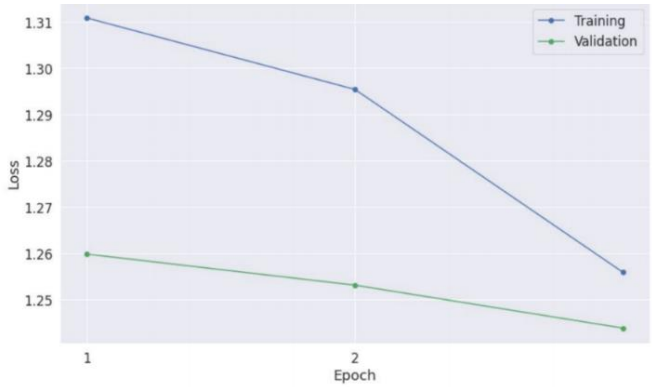

Figure 3: Training and Validation loss of BERT.

2.3.3. RoBERTa

Since RoBERTa [16] proves to be better than BERT in some scenarios, we were eager to use it and compare the results. The pre-trained RoBERTa has been taken from HuggingFace as well, we used the model *'roberta-base'*[6].

The data processing is similar to BERT. The dataset has been split as follows: 70% of data is for training, 20% testing and 10% validation. Hyperparameters used are: text sequence is 256, batches are of 32 elements. Code samples are available in appendix C.

2.3.4. LSTM

The fourth implemented model is LSTM. Training and testing have been done on an 80-20 split. The data processing involves combining the title and text columns and then applying SnowballStemmer[7] from NTLK[8] to stem the text. The text has also been tokenized using Keras's Tokenizer.

Feature extraction uses Word2Vec as it preserves semantic meaning of words in documents, using the embedding matrix resulted we fed it to the model.

The model is built with Tensorflow and it's a combination of the following layers:

- Embedding layer;
- Dropout layer with a dropout rate of 0.3;
- LSTM layer with 100 units with a recurrent dropout (fraction of the units to drop for the linear transformation of the recurrent state) of 0.2 and a dropout of 0.2 (fraction of the units to drop for the linear transformation of the inputs);
- Dense layer with 4 units (because we predict 4 labels) and using SoftMax activation function.

The loss function used was sparse categorical cross entropy with Adam optimizer. The total params of the model were 2,648,304. The optimum number of epochs found were 8 and the batch size 16. We used callback functions such as ReduceLROnPlateau[9] to reduce the learning rate if the accuracy does not improve and early stopping to halt training if the model does not improve.

2.3.5. Bi-LSTM

The fifth and final implemented model is an improvement effort on the previous LSTM network. The dataset split was: 90% training and 10% validation.

---

6 https://huggingface.co/roberta-base
7 https://www.nltk.org/_modules/nltk/stem/snowball.html
8 https://www.nltk.org/
9 https://keras.io/api/callbacks/reduce_lr_on_plateau/

The *title* and *text* columns were merged in a single column, just like all the models. The newly formed *total* column was then processed by removing every stop word and lemmatizing it using NLTK. Finally, the sentences were converted to lowercase and had their whitespaces removed.

The text was tokenized using the Keras Tokenizer. The word index generated length was 27401. For extracting the features, we used GloVe embedding (Global Vectors for Word Representation) with 100 dimensions. Training is performed on aggregated global word-word co-occurrence statistics from a corpus, and the resulting representations show case interesting linear substructures of the word vector space.

For building the model we used Tensorflow. The model was build using the Bidirectional LSTM architecture. We experimented with a lot of combinations of layers but the one that gave the best results during the validation stage was the following (in order):

- Embedding layer with the input dimension equaling the word index length (27401), output dimension equaling the number of embedding dimensions (100) and the input length equaling the maximum sentence length from the training test.
- Bidirectional LSTM layer with 64 units and return sequences set to true.
- Bidirectional LSTM layer with 32 units.
- Dropout layer with dropout rate equaling 0.25 to better handle the overfitting due to the small dataset.
- Dense layer with 4 units (because it predicts 4 labels) and softmax.

The loss function we used was sparse categorical cross entropy with Adam optimizer. The total params of the model was 2,866,156. We experimented with many variations of values for the number of epochs and batch sizes, but the best performing was setting the number of epochs to 5 and the batch size to 32.

3. **Results**

3.1.3Layer Model

In this section we will discuss the results of the 3Layer model as well as parameter tuning on the models. Throughout Table 3 to 6 there have been experiments with each of the 3 feature extraction methods (clean text, POS tagging and semantic tags) as well as a weighted approach of the three; what worked best in the end is the weighed approach combined with Gradient Boosting, this combination earned us 6th place with a F1-macro of **0.44**.

Table 3

TF-IDF Vectorization on Cleaned Text

Classifier	Parameters	Accuracy	Macro Average
Multinomial Naive-Bayes	alpha = 0.0	0.57	0.48
K-Nearest Neighbors	p=2, n_neighbors = 29, leaf_size = 45	0.61	0.41
Random Forest	n_estimators = 1000, max_features = 'sqrt', max_depth = 50, min_samples_split = 2, min_samples_leaf = 2	0.47	0.25
Gradient Boosting	n_estimators = 200	0.57	0.43

Table 4

TF-IDF Vectorization on POS Tags

Classifier	Parameters	Accuracy	Macro Average
Multinomial Naive-Bayes	alpha = 0.0	0.48	0.23
K-Nearest Neighbors	p=2, n_neighbors = 29, leaf_size = 45	0.52	0.37
Random Forest	n_estimators = 400, max_features = 'sqrt', max_depth = 30, min_samples_split = 10, min_samples_leaf = 2	0.54	0.35
Gradient Boosting	n_estimators = 200	0.58	0.44

Table 5

TF-IDF Vectorization on Semantic Tags

Classifier	Parameters	Accuracy	Macro Average
Multinomial Naive-Bayes	alpha = 0.1	0.49	0.29
K-Nearest Neighbors	p=2, n_neighbors = 27, leaf_size = 12	0.35	0.24
Random Forest	n_estimators = 200, max_features = 'sqrt', max_depth = 30, min_samples_split = 10, min_samples_leaf = 1	0.52	0.32
Gradient Boosting	n_estimators = 200	0.52	0.42

Table 6

TF-IDF Vectorization on All Three Representations, using a sparse matrix form

Classifier	Parameters	Accuracy	Macro Average
Multinomial Naive-Bayes	alpha = 0.0	0.62	0.45
K-Nearest Neighbors	p=2, n_neighbors = 19, leaf_size = 6	0.51	0.34
Random Forest	n_estimators = 1000, max_features = auto, max_depth = 30, min_samples_split = 10, min_samples_leaf = 2	0.57	0.39
Gradient Boosting	n_estimators = 200	0.59	0.48

3.2. BERT

Table 6 highlight the performance of BERT; it is clear from this table that the best setup is with 3 epochs, yielding an F1 of **0.5** on the training dataset split.

Table 7
Validation accuracy of BERT on the training dataset split.

Epoch	Training loss	Validation loss	Validation accuracy	Validation F1	Training Time	Validation Time
1	1.31	1.26	0.50	0.50	0:00:44	0:00:02
2	1.29	1.25	0.48	0.48	0:00:47	0:00:03
3	1.25	1.24	0.50	0.50	0:00:50	0:00:03

3.3. RoBERTa

RoBERTa accuracy is very different, depending on the label; the F1-macro is **0.37**. In Table 9 we can see a confusion matrix of the model, unfortunately the imbalance of label has left the system unable to predict 'other' label, it is only good at 'false' and 'partially false'.

Table 8
Classification report for RoBERTa on training data

RoBERTa	Precision	Recall	F1	Support
False	0.65	0.85	0.74	97
True	0.35	0.20	0.26	30
Partially false	0.50	0.49	0.49	47
Other	0.00	0.00	0.00	15
Accuracy			0.59	189
Macro avg	0.38	0.38	0.37	189
Weighted avg	0.51	0.59	0.54	189

Table 9
Confusion matrix for RoBERTa on training data.

Other	0	3	2	10
Partially False	0	23	4	20
True	0	10	6	14
False	0	10	5	82
	Partially False	False	True	Other

3.4. LSTM

The accuracy and loss measured for this model are **0.563157** and **1.405469**.

Table 10
Confusion matrix for LSTM on training data.

Partially false	23	25	5	0
False	10	78	4	0
True	9	12	6	0
Other	3	14	1	0
	Partially False	False	True	Other

3.5. Bi-LSTM

The results were not the best, mainly to the fact that, the dataset was small, the F1-macro for this model has been measured at **0.33**.

Table 11
Classification report for Bi-LSTM on training data.

Bi-LSTM	Precision	Recall	F1	Support
False	0.58	0.73	0.64	92
True	0.43	0.11	0.18	27
Partially false	0.46	0.58	0.52	53
Other	0.00	0.00	0.00	18
Macro avg	0.37	0.36	0.33	190
Weighted avg	0.47	0.53	0.48	190

Table 12
Confusion matrix for Bi-LSTM on training data.

False	67	1	24	0
True	14	3	10	0
Partially false	21	1	31	0
Other	14	2	2	0
	False	True	Partially False	Other

3.6. Results conclusions

To conclude the results section, we had 5 models, the best approach seems to be the **3Layer weighted method that officially has an F1-macro of 0.44**. We were unable to calculate the other scores with the gold label and the organizers did not provide a ranking. Mostly the results seem to revolve around a score of 0.5 which is in part related to the small dimension of the dataset and the fact that many of our models relied on neural network which require large training sets.

4. Conclusions

To conclude, in this paper we presented our run at the CLEF 2021 Task 3a; our best method had a F1-macro of 0.44 ranking us 6th. We proposed multiple mod-els based on different methods, for future work we plan on increasing the dataset as well as create a system based on inference so that the article content will be verified using different ontologies.

5. Acknowledgements

Special thanks go to: Smau Adrian-Constantin, Mosor Andre, Radu Rares-Aurelian, Gramescu George-Rares, Filipescu Iustina-Andreea without whom this work would not have been possible. This work was supported by project REVERT (taRgeted thErapy for adVanced colorEctal canceR paTients), Grant Agreement number: 848098, H2020-SC1-BHC-2018-2020/H2020-SC1-2019-Two-Stage-RTD.

6. References

[1] Preslav Nakov and Giovanni Da San Martino and Tamer Elsayed and Alberto Barr'on- Cedeño and Rubén M'iguez and Shaden Shaar and Firoj Alam and Fatima Haouari and Maram Hasanain and Nikolay Babulkov and Alex Nikolov and Gautam Kishore Shahi and Julia Maria Struss and Thomas Mandl (2021). The CLEF-2021 CheckThat! Lab on Detecting Check-Worthy Claims, Previously Fact-Checked Claims, and Fake News. In Advances in Information Retrieval - 43rd European Conference on IR Research, ECIR 2021, Virtual Event, March 28 - April 1, 2021, Proceedings, Part II (pp. 639–649). Springer.

[2] Shahi, G., Dirkson, A., & Majchrzak, T. (2021). An exploratory study of covid-19 misinformation on twitter. Online Social Networks and Media, 22, 100104.

[3] X. Zhang, and A.A. Ghorbani, "An overview of online fake news: Characterization, detection, and discussion." In Information Processing & Management, vol. 57 (2), 2020, 102025, ISSN 0306-4573, https://doi.org/10.1016/j.ipm.2019.03.004.

[4] Emilio Ferrara, Onur Varol, Clayton Davis, Filippo Menczer, and Alessandro Flammini. 2016. The rise of social bots. Commun. ACM 59, 7 (July 2016), 96–104. DOI:https://doi.org/10.1145/2818717

[5] J. Zhao, N. Cao, Z. Wen, Y. Song, Y. Lin and C. Collins, "#FluxFlow: Visual Analysis of Anomalous Information Spreading on Social Media," in IEEE Transactions on Visualization and Computer Graphics, vol. 20, no. 12, pp. 1773-1782, 31 Dec. 2014, doi: 10.1109/TVCG.2014.2346922.

[6] Ghosh, Rumi, Tawan Surachawala, and Kristina Lerman. "Entropy-based classification of'retweeting'activity on twitter." arXiv preprint arXiv:1106.0346 (2011).

[7] Alan Mislove, Massimiliano Marcon, Krishna P. Gummadi, Peter Druschel, and Bobby Bhattacharjee. 2007. Measurement and analysis of online social networks. In Proceedings of the 7th ACM SIGCOMM conference on Internet measurement (IMC '07). Association for Computing Machinery, New York, NY, USA, 29–42. DOI:https://doi.org/10.1145/1298306.1298311

[8] J. P. Dickerson, V. Kagan and V. S. Subrahmanian, "Using sentiment to detect bots on Twitter: Are humans more opinionated than bots?," 2014 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2014), 2014, pp. 620-627, doi: 10.1109/ASONAM.2014.6921650.

[9] Carlos Castillo, Marcelo Mendoza, and Barbara Poblete. 2011. Information credibility on twitter. In Proceedings of the 20th international conference on World wide web (WWW '11). Association for Computing Machinery, New York, NY, USA, 675–684. DOI:https://doi.org/10.1145/1963405.1963500

[10] Diakopoulos, N., Naaman, M., Kivran-Swaine, F. 2010. Diamonds in the rough: Social media visual analytics for journalistic inquiry. In VAST 10 - IEEE Conference on Visual Analytics Science and Technology 2010, Proceedings, art. no. 5652922, pp. 115-122. DOI: 10.1109/VAST.2010.5652922

[11] Pritika Bahad, Preeti Saxena, Raj Kamal, Pritika Bahad, Preeti Saxena, Raj Kamal, Fake News Detection using Bi-directional LSTM-Recurrent Neural Network, Procedia Computer Science, Volume 165, 2019, Pages 74-82, ISSN 1877-0509, https://doi.org/10.1016/j.procs.2020.01.072.
[12] C.G. Cusmuliuc, L.G. Coca, and A. Iftene, “Identifying Fake News on Twitter using Naive Bayes, SVM and Random Forest Distributed Algorithms.” In Proceedings of The 13th Edition of the International Conference on Linguistic Resources and Tools for Processing Romanian Language (ConsILR-2018), 2018, pp. 177-188.
[13] Jwa, H., Oh, D., Park, K., Kang, J. M., & Lim, H. (2019). exBAKE: automatic fake news detection model based on bidirectional encoder representations from transformers (bert). Applied Sciences, 9(19), 4062.
[14] Devlin, Jacob, et al. "Bert: Pre-training of deep bidirectional transformers for language understanding." arXiv preprint arXiv:1810.04805 (2018).
[15] Fast, Ethan, Binbin Chen, and Michael S. Bernstein. "Empath: Understanding topic signals in large-scale text." Proceedings of the 2016 CHI conference on human factors in computing systems. 2016.
[16] Liu, Yinhan, et al. "Roberta: A robustly optimized bert pretraining approach." arXiv preprint arXiv:1907.11692 (2019).
[17] Shahi, Julia Maria and Thomas Mandl. "Overview of the CLEF-2021 CheckThat! Lab Task 3 on Fake News Detection." Working Notes of CLEF 2021–-Conference and Labs of the Evaluation Forum.
[18] P. Nakov, G. Da San Martino, T. Elsayed, A. Barrón-Cedeño, R. Míguez, S. Shaar, F. Alam,F. Haouari, M. Hasanain, N. Babulkov, A. Nikolov, G. K. Shahi, J. M. Struß, T. Mandl, S. Modha, M. Kutlu, Y. S. Kartal, Overview of the CLEF-2021 CheckThat! Lab on Detecting Check-Worthy Claims, Previously Fact-Checked Claims, and Fake News, in: Proceedings of the 12th International Conference of the CLEF Association: Information Access Evaluation Meets Multiliguality, Multimodality, and Visualization, CLEF ’2021, Bucharest, Romania(online), 2021

Apendix A

lexicon.analyze("he hit the other person", normalize=True)
# => {'help': 0.0, 'office': 0.0, 'violence': 0.2, 'dance': 0.0, 'money': 0.0, 'wedding': 0.0, 'valuable': 0.0, 'domestic_work': 0.0, 'sleep': 0.0, 'medical_emergency': 0.0, 'cold': 0.0, 'hate': 0.0, 'cheerfulness': 0.0, 'aggression': 0.0, 'occupation': 0.0, 'envy': 0.0, 'anticipation': 0.0, 'family': 0.0, 'crime': 0.0, 'attractive': 0.0, 'masculine': 0.0, 'prison': 0.0, 'health': 0.0, 'pride': 0.0, 'dispute': 0.0, 'nervousness': 0.0, 'government': 0.0, 'weakness': 0.0, 'horror': 0.0, 'swearing_terms': 0.0, 'leisure': 0.0, 'suffering': 0.0, 'royalty': 0.0, 'wealthy': 0.0, 'white_collar_job': 0.0, 'tourism': 0.0, 'furniture': 0.0, 'school': 0.0, 'magic': 0.0, 'beach': 0.0, 'journalism': 0.0, 'morning': 0.0, 'banking': 0.0, 'social_media': 0.0, 'exercise': 0.0, 'night': 0.0, 'kill': 0.0, 'art': 0.0, 'play': 0.0, 'computer': 0.0, 'college': 0.0, 'traveling': 0.0, 'stealing': 0.0, 'real_estate': 0.0, 'home': 0.0, 'divine': 0.0, 'sexual': 0.0, 'fear': 0.0, 'monster': 0.0, 'irritability': 0.0, 'superhero': 0.0, 'business': 0.0, 'driving': 0.0, 'pet': 0.0, 'childish': 0.0, 'cooking': 0.0, 'exasperation': 0.0, 'religion': 0.0, 'hipster': 0.0, 'internet': 0.0, 'surprise': 0.0, 'reading': 0.0, 'worship': 0.0, 'leader': 0.0, 'independence': 0.0, 'movement': 0.2, 'body': 0.0, 'noise': 0.0, 'eating': 0.0, 'medieval': 0.0, 'zest': 0.0, 'confusion': 0.0, 'water': 0.0, 'sports': 0.0, 'death': 0.0, 'healing': 0.0, 'legend': 0.0, 'heroic': 0.0, 'celebration': 0.0, 'restaurant': 0.0, 'ridicule': 0.0, 'programming': 0.0, 'dominant_heirarchical': 0.0, 'military': 0.0, 'neglect': 0.0, 'swimming': 0.0, 'exotic': 0.0, 'love': 0.0, 'hiking': 0.0, 'communication': 0.0, 'hearing': 0.0, 'order': 0.0, 'sympathy': 0.0, 'hygiene': 0.0, 'weather': 0.0, 'anonymity': 0.0, 'trust': 0.0, 'ancient': 0.0, 'deception': 0.0, 'fabric': 0.0, 'air_travel': 0.0, 'fight': 0.0, 'dominant_personality': 0.0, 'music': 0.0, 'vehicle': 0.0, 'politeness': 0.0, 'toy': 0.0, 'farming': 0.0, 'meeting': 0.0, 'war': 0.0, 'speaking': 0.0, 'listen': 0.0, 'urban': 0.0, 'shopping': 0.0, 'disgust': 0.0, 'fire': 0.0, 'tool': 0.0, 'phone': 0.0, 'gain': 0.0, 'sound': 0.0, 'injury': 0.0, 'sailing': 0.0, 'rage': 0.0, 'science': 0.0, 'work': 0.0, 'appearance': 0.0, 'optimism': 0.0, 'warmth': 0.0, 'youth': 0.0, 'sadness': 0.0, 'fun': 0.0, 'emotional': 0.0, 'joy': 0.0, 'affection': 0.0, 'fashion': 0.0, 'lust': 0.0, 'shame': 0.0, 'torment': 0.0, 'economics': 0.0, 'anger': 0.0, 'politics': 0.0, 'ship': 0.0, 'clothing': 0.0, 'car': 0.0, 'strength': 0.0, 'technology': 0.0, 'breaking': 0.0, 'shape_and_size': 0.0, 'power': 0.0, 'vacation': 0.0, 'animal': 0.0, 'ugliness': 0.0, 'party': 0.0, 'terrorism': 0.0, 'smell': 0.0, 'blue_collar_job': 0.0, 'poor': 0.0, 'plant': 0.0, 'pain': 0.2, 'beauty': 0.0, 'timidity': 0.0, 'philosophy': 0.0, 'negotiate': 0.0, 'negative_emotion': 0.0, 'cleaning': 0.0, 'messaging': 0.0, 'competing': 0.0, 'law': 0.0, 'friends': 0.0, 'payment': 0.0, 'achievement': 0.0, 'alcohol': 0.0, 'disappointment': 0.0, 'liquid': 0.0, 'feminine': 0.0, 'weapon':

0.0, 'children': 0.0, 'ocean': 0.0, 'giving': 0.0, 'contentment': 0.0, 'writing': 0.0, 'rural': 0.0, 'positive_emotion': 0.0, 'musical': 0.0}

Apendix B

```
class BertClassifier(nn.Module):
    def __init__(self, dropout=0.1):
        super(BertClassifier, self).__init__()
        self.bert = BertModel.from_pretrained('bert-base-uncased')
        self.dropout = nn.Dropout(dropout)
        self.linear = nn.Linear(768, 1)
        self.sigmoid = nn.Sigmoid()
    def forward(self, tokens, masks=None):
        _, pooled_output = self.bert(tokens, attention_mask=masks, output_all_encoded_layers=False)
        dropout_output = self.dropout(pooled_output)
        linear_output = self.linear(dropout_output)
        proba = self.sigmoid(linear_output)
        return proba
```

Apendix C

```
class ROBERTA(torch.nn.Module, Model):
    def __init__(self, text, dropout_rate=0.4):
        super(ROBERTA, self).__init__()
        # Model.__init__(text)
        self.text = text
        self.tokenizer = RobertaTokenizer.from_pretrained("roberta-base")
        self.roberta = RobertaModel.from_pretrained('roberta-base',return_dict=False, num_labels = 4)
        self.d1 = torch.nn.Dropout(dropout_rate)
        self.l1 = torch.nn.Linear(768, 64)
        self.bn1 = torch.nn.LayerNorm(64)
        self.d2 = torch.nn.Dropout(dropout_rate)
        self.l2 = torch.nn.Linear(64, 4)
```