COURAGE at CheckThat! 2022: Harmful Tweet Detection using Graph Neural Networks and ELECTRA

Francesco Lomonaco[1], Gregor Donabauer[1,3] and Marco Siino[2]

[1]Università degli Studi di Milano Bicocca, Dipartimento di Informatica, Sistemistica e Comunicazione, Milano, 20126, Italy

[2]Università degli Studi di Palermo, Dipartimento di Ingegneria, Palermo, 90128, Italy

[3]University of Regensburg, Information Science, Regensburg, 93053, Germany

Abstract

In this paper we propose a deep learning model based on graph machine learning (i.e. Graph Attention Convolution) and a pretrained transformer language model (i.e. ELECTRA). Our model was developed to detect harmful tweets about COVID-19 and was used to tackle subtask 1C (harmful tweet detection) at the CheckThat!Lab shared task organized as part of CLEF 2022. In this binary classification task, our proposed model reaches a binary F1 score (positive class label, i.e. harmful tweet) of 0.28 on the test set. We demonstrate that our approach outperforms the official baseline by 8% and describe our model as well as the experimental setup and results in detail. We also refer to limitations of the approach and future research directions.

Keywords

text graph classification, harmful tweets, Twitter, ELECTRA, COVID-19

1. Introduction

Along the COVID-19 outbreak the spread of misleading information online related to news on the pandemic could be observed, for example on social media. The three tasks proposed for the CheckThat! Lab@CLEF2022 [1, 2] aim at addressing related issues, namely: (1) Identifying relevant claims in tweets, (2) Detecting previously fact checked claims and (3) Fake news detection. All tasks are framed as classification problems and build on collective effort to tackle the COVID-19 related infodemic.

The aim of this paper is to propose a model to address the first task [3]. Furthermore, this task includes four different subtasks. Subtask 1A is about determining check-worthiness of tweets (i.e. given a tweet, predict whether it is worth fact-checking). Subtask 1B is about verifiable factual claims detection: given a tweet, predict whether it contains a verifiable factual claim. In Subtask 1C the focus is on harmful tweet detection: given a tweet, predict whether it is harmful to the society and why. Finally, Subtask 1D is related to attention-worthy tweet detection: given

CLEF 2022: Conference and Labs of the Evaluation Forum, September 5–8, 2022, Bologna, Italy

f.lomonaco5@campus.unimib.it (F. Lomonaco); gregor.donabauer@ur.de (G. Donabauer); marco.siino@unipa.it (M. Siino)

0000-0002-4453-5352 (M. Siino)

a tweet, predict whether it should get the attention of policy makers and why. This task is defined with eight class labels. For the subtasks 1A and 1C the official evaluation metric is the binary F1 score with respect to the positive class, for subtask 1B the metric used is the accuracy and for subtask 1D the metric used is the weighted F1 score.

We present our model proposed for Subtask 1C - English language. The model takes advantage of an ELECTRA-based document embedding as well as a text graph that is processed using a Graph Convolutional Network (GCN). Our goal is to introduce a novel method that can handle different types of heterogeneous textual or social information. We show how a first version of such a model performs on the proposed task, leaving room for improvements on future research in the domain. To support reproducibility and future research directions we make our Code publicly available[1].

This paper is organized as follows: in Section 2 we present some related work about the usage of deep learning methods for similar text classification tasks. In Section 3 we describe our approach in detail, explaining our choices and the configuration of each layer of the model. In Section 5 we report the results we obtain on the official test set. In Section 6 we discuss some interesting future directions to investigate before closing our work with concluding remarks in Section 7.

2. Related work

Motivated by the notable performances reached in various text classification tasks, where deep AI models [4, 5] outperformed classic techniques used in natural language processing (e.g. Bayes, Decision Tree, K-Nearest Neighbour, Support Vector Machine) as also reported in [6, 7], we decided to use a deep learning-based approach for our submission.

Our proposed model is based on both, a transformer-based document embedding and a GCN (i.e. Graph Attention Convolution) applied to a text graph representing the structure of each tweet. The transformer-based embedding we used is ELECTRA a language model presented by [8]. ELECTRA does not mask the input as BERT but replaces some tokens with plausible alternatives sampled from a small generator network. Then, instead of training a model that predicts the original identities of the corrupted tokens, a discriminative model is trained to predict whether each token in the corrupted input is replaced by a generator sample or not.

GCNs take advantage of graph data structures that are made of vertices and edges (also called nodes and link). Nowadays, graphs are used in many real-world applications, including traffic prediction [9], computer vision [10], social networks [11, 12] and many more.

GCNs for text classification are discussed in [13]: the authors propose a novel graph neural network method and model a whole corpus as a heterogeneous graph to learn word and document embeddings with graph neural networks jointly. Results on several benchmark datasets demonstrate that the proposed method outperforms state-of-the-art text classification methods, without using pre-trained word embeddings or external knowledge. The model proposed also learns predictive word and document embeddings automatically.

In [14] a more sophisticated approach is proposed. In particular, the authors discussed a flexible Heterogeneous Information Network (HIN) for modeling short texts. The model can

[1] https://github.com/sagacemente/CLEF2022CheckThat.git

integrate any type of additional information as well as capturing their relations to address the semantic sparsity. Additionally, the authors propose heterogeneous graph attention networks to embed the HIN for short text classification based on a dual-level attention mechanism, including node-level and type-level attention. The attention mechanism can learn the importance of different neighboring nodes as well as the importance of different node (information) types to a current node.

A broader overview over a range of text classification applications using GCNs is given in [15].

3. Proposed model

The model architecture with input and output shapes of each layer is shown in Figure 1 along with parameter distributions of each layer. The proposed model is composed by two modules:

- Graph creation and embedding
- Pretrained document embedding

```
----------------------------------------------------------------
Model Parameters
----------------------------------------------------------------
     Layer.Parameter         Param Tensor Shape         Param #
----------------------------------------------------------------
           conv1.att               [1, 4, 450]            1800
          conv1.bias                    [1800]            1800
  conv1.lin_l.weight               [1800, 815]         1467000
    conv1.lin_l.bias                    [1800]            1800
  conv1.lin_r.weight               [1800, 815]         1467000
    conv1.lin_r.bias                    [1800]            1800
           conv2.att               [1, 1, 450]             450
          conv2.bias                     [450]             450
  conv2.lin_l.weight               [450, 1800]          810000
    conv2.lin_l.bias                     [450]             450
  conv2.lin_r.weight               [450, 1800]          810000
    conv2.lin_r.bias                     [450]             450
         lin1.weight               [609, 1218]          741762
           lin1.bias                     [609]             609
         lin3.weight                  [2, 609]            1218
           lin3.bias                        [2]               2
----------------------------------------------------------------
Total params: 5306591
Trainable params: 5306591
Non-trainable params: 0

----------------------------------------------------------------
Model Architechture
----------------------------------------------------------------
GCN(
  (conv1): GATv2Conv(815, 450, heads=4)
  (conv2): GATv2Conv(1800, 450, heads=1)
  (lin1): Linear(in_features=1218, out_features=609, bias=True)
  (lin3): Linear(in_features=609, out_features=2, bias=True)
)
```

Figure 1: Model parameters (Top) numbers in brackets indicate parameters' tensor dimensions; last column indicates the number of parameters in each layer. Model architecture (bottom) model input and output shapes in each layer (figure taken from our Google Colab notebook).

Geometric deep learning [16, 17] has led to a growing number of new architectures as well as novel applications, including text modelling [18]. The representation of each tweet into a

graph starts with text preprocessing and Part Of Speech (POS) tagging. After these steps each unique tagged word in the tweet corresponds to a node in the graph and the adjacency matrix is populated connecting each node with all words in a window equal to 3. Each node is annotated with various features discussed in 3.2. The proposed architecture is composed of two graph attention convolution (i.e. GATV2Conv) layers proposed by [19]; the node-wise representation outputted by the GATV2Conv layers is passed to a max pooling operator and a dropout layer. The output is then concatenated with the document embedding generated using ELECTRA [8]. Finally, two dense layers and a rectified linear unit (ReLu) activation function between them output the predictions of the model for each class.

Before discussing the network architecture as well as our hyperparameter settings, we want to mention that each split of the dataset (training and test per language) consists of individual tweets and their corresponding labels.

3.1. Graph creation

The graph module takes as input a raw sample (tweet) and outputs the tweet represented as an undirected, attributed graph. In Figure 2 each step of the preprocessing pipeline is depicted. Our custom preprocessing function uses the python NLTK package [20]. Below we list all the preprocessing steps involved:

- *Lowercasing*. This step is used to get the same embedding e.g. for the words **Hello** and **hello**
- *Removing Stopwords*. Stopwords are generally speaking used with high frequency but they are in many cases not really informative, e.g. preposition and articles belong to this category.
- *POS Tagging*. In this step each word in the tweet is classified into its parts of speech class and labelled accordingly using a one-hot encoding. These vectors correspond to the respective POS tag out of all 43 POS classes in the NLTK package.
- *URL removing*. All URLs in each tweet have been removed.
- *Hashtag symbol and tagged accounts*. All hashtag symbols have been removed along with tagged users.

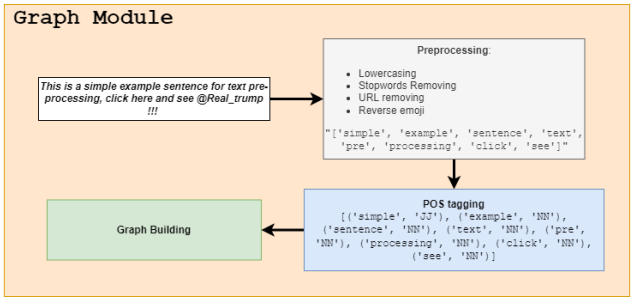

Figure 2: Graph representation: each tweet is represented as a graph after pre-processing and POS tagging

Starting from the output of POS-tagging we adopt a strategy that associates an edge to each word with all words in a window equal to 3. If a word is repeated more then once, only the first occurrence is considered as node while edges are updated accordingly. Edges are unweighted.

3.2. Node characterisation

As mentioned above in Figure 1 each node is characterized as a 815-dimensional vector. The first 768 features correspond to the pretrained ELECTRA document embedding, obtained using FLAIR, applying the introduced preprocessing steps [2][21]. To select the best embeddings we evaluated different transformer-based models using the tweet text data as input. The official evaluation metric was used during this experiment and it turned out that ELECTRA outperformed other pretrained language models. Using ELECTRA we collected both word embedding for each node in the graph as well as document embedding for the whole tweet. Each node was also annotated with the corresponding one-hot-encoded vector of its POS tag (45 features). Given the fact that graph networks are order invariant w.r.t the nodes processed during message passing the order of words in the original tweet is lost. To maintain this information we characterized each node with a feature vector of two dimensions that encodes the distance from the origin of each node (word) in the graph using sine and cosine positional encoding of a transformer model [22]. This vector is concatenated to the other node features.

3.3. Graph attention convolution and max pooling layer

In our model we use two GATV2Conv [19] layers. This layer is characterized by the computation of dynamic attention scores. Moreover we adopt multiple heads in the first layer where the number of heads is set to four, because (as demonstrated previously by [23]) the learning process can benefit from employing multi-head attention and concatenating their outputs. As highlighted in Figure 1 the number of features used to represent each node is halved between the 2 layers. The output of the graph attention layer is a 2D matrix with shape: Number of nodes (d) * Number of features (N). The maximum value is calculated along the dimension of size N, in fact reducing the dimension of the input tensor by one. As an example consider the following matrix X.

$$X = \begin{bmatrix} x_{11} & x_{12} & x_{13} & \dots & x_{1n} \\ x_{21} & x_{22} & x_{23} & \dots & x_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{d1} & x_{d2} & x_{d3} & \dots & x_{dn} \end{bmatrix}$$

Providing X as input to a 1D-max pooling layer returns the following array Y as output, where each y_i is computed taking the maximum of all the values along the i-th column of the matrix X.

$$Y = \begin{bmatrix} y_1 & y_2 & y_3 & \dots & y_n \end{bmatrix}$$

[2]Documentation available here: https://github.com/flairNLP/flair

3.4. Dense

The max pooling layer's output is concatenated with the tweet embedding obtained from ELECTRA. This vector is fully connected to a dense layer which is followed by a rectified linear unit function element-wise (e.g., $Relu(x) = max(0, x)$) and finally a dense layer with two units as output. This float values correspond to the softmax logits, a vector of raw (non-normalized) predictions.

4. Experimental setup

We participated in *Subtask C (harmful tweet detection)* of CheckThat!'s Task 1 for the English language. Before addressing experimental setup and model training we briefly describe the provided dataset.

4.1. Dataset

The corpus includes a list of tweets that are either labeled as harmful (1) or not (0). In addition the ID of the tweets and its urls are available. All samples are related to COVID-19. In general, a train, development and dev-test set are provided as well as an official test set that was used for evaluation of the submissions. While for the first three dataset splits the gold labels were available, those of the official test set were held out till the end of the evaluation phase. In general, the number of samples for all parts of the dataset are distributed as shown in table 1. The data were released as multiple tab-separated files (one per split)

Dataset	**Number of Samples**	Label 0	Label 1
Train	3323	3031	292
Development	307	276	31
Dev-Test	910	828	82
Test	251	211	40

Table 1
Dataset statistics of all provided splits for English.

An exploratory analysis shows the dataset imbalance with respect to the class labels. For the training set the positive class samples correspond to only 8% of the total entries. Using the Tweet IDs provided we crawled twitter data via the official Twitter API [3]. However, only a small subset of the samples (w.r.t. the training set only 20% of the original tweets) was still available as the rest of this information was already deleted by Twitter. Given this observation, we choose to discard including social context information such as the number of tweet interaction (favourites, shares), author features (follower following relationships as well as user timeline tweets). Besides using the graph based approach introduced in Section 1 we performed further experiments on the dataset featuring transformer based methods as well as alternative graph construction techniques. We present details on the results of these experiments in Section 5.

[3] API Documentation available here: https://developer.twitter.com/en/docs/twitter-api

4.2. Model training

The hyperparameter settings are in line with many of the decisions made in a study conducted in [24] and used to subsequently fine-tune our proposed model. To initialize the weights of the model we used a Glorot uniform initializer [25]. The model was compiled using binary cross entropy loss; this function calculates the loss with respect to two classes (i.e., 0 and 1) as defined in 1.

$$Loss_{BCE} = -\frac{1}{N}\sum_{n=1}^{N}[y_n \times \log\left(h_\theta(x_n)\right) + (1 - y_n) \times \log\left(1 - h_\theta(x_n)\right)] \tag{1}$$

where:

- N is the number of training examples;
- y_n is the target label for the training sample n;
- x_n is the input sample n;
- h_θ is the neural network model with weights θ.

To improve the model performance and counteract class imbalance we set up class weights that correspond to a manual rescaling weight assigned to each class. Optimization is performed using the Adam optimizer [26]. To reduce overfitting we take advantage of a learning rate scheduler reducing the learning rate by a factor of 0.9 with a step size of 25 epochs. The model architecture is depicted in figure 1, where the number of the various network hyperparameters are provided.

5. Results

5.1. Baseline

The organizers provide official baseline results based on random predictions. The metric to evaluate the task is the binary F1 score of the positive class label (harmful tweet). The official baseline result amounts to 0.200 binary F1 on the evaluation test set. We compare the baseline results to the values our approaches did achieve in Table 2.

We established an additional, strong baseline based on a fine-tuned transformer model. We evaluated different transformer architectures including BERT, RoBERTa and ELECTRA regarding the classification task. It turned out that ELECTRA achieves the best results among these models (using 3 epochs for fine-tuning as recommended by [27]). The performance of this approach amounts to 0.250 binary F1 score.

5.2. Our approach

Our main experiments are based on the model proposed in Section 3. We will report results on the test set used for evaluation using the official binary F1 metric, as well as binary precision and binary recall of the positive class label.

As presented in Table 2 our submitted approach (*GCN+ELECTRA*) outperforms the official baseline by 8%. The official baseline approach generates class labels in random order.

Approach	Binary Precision	Binary Recall	Binary F1
Baseline	0.200	0.200	0.200
GCN+3-gram-ELECTRA	0.138	0.625	0.226
ELECTRA (3 epochs)	0.263	0.250	0.256
GCN+POS w/o word embeddings	0.166	0.650	0.264
ELECTRA (50 epochs)	0.275	0.275	0.275
GCN+ELECTRA	0.166	0.875	0.280

Table 2
Results (binary Precision, Recall and F1 of the positive class label) on the official test set for English with respect to different approaches.

Compared to the performance of our own baseline using ELECTRA, the *GCN+ELECTRA* outperforms this approach by 3%. We also evaluated a ELECTRA fine-tuning setup using 50 epochs resulting in a performance almost as good as the finally submitted approach. However, the high number of epochs lead to strong overfitting on the training data.

In addition we report results obtained by experimental setups that we evaluated as part of the development process of the submitted approach. In Table 2 *GCN+POS w/o word embeddings* refers to a setting where we omit word embeddings and represent graph nodes by only considering one-hot encoded POS-tag vectors. In the *GCN+3-gram-ELECTRA* model we characterize graph nodes by mean-pooled word embeddings of 3 subsequent words at each position. Thus we also wanted to take into account word orders that can be lost during graph convolution.

6. Discussion and future work

All other approaches used during our experiments result in a lower performance compared to the actually submitted model. This observation strengthens our decision of choosing to submit predictions generated with the model proposed in this paper as it outperforms a range of other setups we used during our experiments. Obviously, we can report a high recall score compared to low precision using our proposed model as it tends to frequently predict the positive class label and only a few times the negative one.

As already mentioned above, the inclusion of social and user information in the model (as in [28]) can improve the classification accuracy. Deleted tweets that can not be crawled anymore limit the usage of those information in the actual training set.

In future work, it would be interesting to analyze the contribution of such social context features. In addition, different types of embeddings (stacked or pooled) could be compared regarding the characterization of each node.

7. Conclusion

In this paper, we describe our approach to Subtask- 1C at CheckThat!Lab@CLEF2022 based on graph data structures and transformer language models. The proposed hybrid solution of a text graph and pretrained document emebddings should be studied in more detail as it leads to improvements over fine-tuning transformer-based models as demonstrated on the given

dataset. In addition there is room left for including additional types of information (e.g. social media context) in these data structures which could be helpful in solving other tasks. Overall, as reported by the organizers, our approach — achieving an F1-score of the positive class of 0.280 — ranked eight in the CheckThat!Lab@CLEF2022 - Subtask- 1C.

Acknowledgments

We would like to thanks reviewers for precious insight to increase readability of the paper and remove typos. This work has been partially founded by COURAGE - A social media companion safeguarding and educating students (no. 95567), funded by the Volkswagen Foundation in the topic Artificial Intelligence and the Society of the Future.

References

[1] P. Nakov, A. Barrón-Cedeño, G. Da San Martino, F. Alam, J. M. Struß, T. Mandl, R. Míguez, T. Caselli, M. Kutlu, W. Zaghouani, C. Li, S. Shaar, G. K. Shahi, H. Mubarak, A. Nikolov, N. Babulkov, Y. S. Kartal, J. Beltrán, M. Wiegand, M. Siegel, J. Köhler, Overview of the CLEF-2022 CheckThat! lab on fighting the COVID-19 infodemic and fake news detection, in: Proceedings of the 13th International Conference of the CLEF Association: Information Access Evaluation meets Multilinguality, Multimodality, and Visualization, CLEF '2022, Bologna, Italy, 2022.

[2] P. Nakov, A. Barrón-Cedeño, G. Da San Martino, F. Alam, J. M. Struß, T. Mandl, R. Míguez, T. Caselli, M. Kutlu, W. Zaghouani, C. Li, S. Shaar, G. K. Shahi, H. Mubarak, A. Nikolov, N. Babulkov, Y. S. Kartal, J. Beltrán, The clef-2022 checkthat! lab on fighting the covid-19 infodemic and fake news detection, in: M. Hagen, S. Verberne, C. Macdonald, C. Seifert, K. Balog, K. Nørvåg, V. Setty (Eds.), Advances in Information Retrieval, Springer International Publishing, Cham, 2022, pp. 416–428.

[3] P. Nakov, A. Barrón-Cedeño, G. Da San Martino, F. Alam, R. Míguez, T. Caselli, M. Kutlu, W. Zaghouani, C. Li, S. Shaar, H. Mubarak, A. Nikolov, Y. S. Kartal, J. Beltrán, Overview of the CLEF-2022 CheckThat! lab task 1 on identifying relevant claims in tweets, in: Working Notes of CLEF 2022—Conference and Labs of the Evaluation Forum, CLEF '2022, Bologna, Italy, 2022.

[4] M. Siino, E. Di Nuovo, T. Ilenia, M. La Cascia, Detection of hate speech spreaders using convolutional neural networks, in: PAN 2021 Profiling Hate Speech Spreaders on Twitter@ CLEF, volume 2936, CEUR, 2021, pp. 2126–2136.

[5] M. Siino, M. La Cascia, I. Tinnirello, McRock at SemEval-2022 Task 4: Patronizing and Condescending Language Detection using Multi-Channel CNN and DistilBERT, in: Proceedings of the 16th International Workshop on Semantic Evaluation (SemEval-2022), Association for Computational Linguistics, 2022.

[6] H. Wu, Y. Liu, J. Wang, Review of text classification methods on deep learning, CMC-Computers, Materials & Continua 63 (2020) 1309–1321.

[7] S. Hashida, K. Tamura, T. Sakai, Classifying tweets using convolutional neural networks

with multi-channel distributed representation, IAENG International Journal of Computer Science 46 (2019) 68–75.
[8] K. Clark, M.-T. Luong, Q. V. Le, C. D. Manning, Electra: Pre-training text encoders as discriminators rather than generators, arXiv preprint arXiv:2003.10555 (2020).
[9] Y. Li, R. Yu, C. Shahabi, Y. Liu, Diffusion convolutional recurrent neural network: Data-driven traffic forecasting, arXiv preprint arXiv:1707.01926 (2017).
[10] P. Pradhyumna, G. Shreya, et al., Graph neural network (gnn) in image and video understanding using deep learning for computer vision applications, in: 2021 Second International Conference on Electronics and Sustainable Communication Systems (ICESC), IEEE, 2021, pp. 1183–1189.
[11] L. Backstrom, J. Leskovec, Supervised random walks: predicting and recommending links in social networks, in: Proceedings of the fourth ACM international conference on Web search and data mining, 2011, pp. 635–644.
[12] M. Siino, M. La Cascia, I. Tinnirello, Whosnext: Recommending twitter users to follow using a spreading activation network based approach, in: 2020 International Conference on Data Mining Workshops (ICDMW), IEEE, 2020, pp. 62–70.
[13] L. Yao, C. Mao, Y. Luo, Graph convolutional networks for text classification, in: Proceedings of the AAAI conference on artificial intelligence, volume 33, 2019, pp. 7370–7377.
[14] H. Linmei, T. Yang, C. Shi, H. Ji, X. Li, Heterogeneous graph attention networks for semi-supervised short text classification, in: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), 2019, pp. 4821–4830.
[15] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, M. Sun, Graph neural networks: A review of methods and applications, AI Open 1 (2020) 57–81.
[16] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, P. Vandergheynst, Geometric deep learning: going beyond euclidean data, IEEE Signal Processing Magazine 34 (2017) 18–42.
[17] T. N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, arXiv preprint arXiv:1609.02907 (2016).
[18] D. Embarcadero-Ruiz, H. Gómez-Adorno, A. Embarcadero-Ruiz, G. Sierra, Graph-based siamese network for authorship verification, Mathematics 10 (2022) 277.
[19] S. Brody, U. Alon, E. Yahav, How attentive are graph attention networks?, arXiv preprint arXiv:2105.14491 (2021).
[20] S. Bird, E. Loper, Nltk: the natural language toolkit, Association for Computational Linguistics, 2004.
[21] A. Akbik, T. Bergmann, D. Blythe, K. Rasul, S. Schweter, R. Vollgraf, Flair: An easy-to-use framework for state-of-the-art nlp, in: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations), 2019, pp. 54–59.
[22] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, Advances in neural information processing systems 30 (2017).
[23] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, Y. Bengio, Graph attention networks, arXiv preprint arXiv:1710.10903 (2017).
[24] R. Wilkens, D. Ognibene, Mb-courage@ exist: Gcn classification for sexism identification

in social networks, IberLEF@ EXIST (2021).
[25] Keras, Layer weight initializers, https://keras.io/api/layers/initializers/, 2021.
[26] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, 2017. arXiv:1412.6980.
[27] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, Bert: Pre-training of deep bidirectional transformers for language understanding, 2018. URL: https://arxiv.org/abs/1810.04805. doi:10.48550/ARXIV.1810.04805.
[28] V. Balakrishnan, S. Khan, H. R. Arabnia, Improving cyberbullying detection using twitter users' psychological features and machine learning, Computers & Security 90 (2020) 101710.