Antisocial Online Behavior Detection Using Deep Learning. **Online Appendix**

Elizaveta Zinovyeva*[1], Wolfgang Karl Härdle[1,2,3,4,5], and Stefan Lessmann[1]

[1]School of Business and Economics, Humboldt-Universität zu Berlin, Berlin, Germany
[2]Sim Kee Boon Institute for Financial Economics, Singapore Management University, Singapore, Singapore
[3]W.I.S.E. - Wang Yanan Institute for Studies in Economics, Xiamen University, Fujian, China
[4]Faculty of Mathematics and Physics, Charles University, Prague, Czech Republic
[5]Department of Information Management and Finance, National Chiao Tung University, Hsinchu, Taiwan

1 Appendix I. Deep Learning in AOB detection

The following section provides an overview of academic literature on the usage of deep learning (DL) in antisocial online behavior (AOB) detection. In the tables 1 and 2, we depict papers on DL in AOB detection. We identify some details, algorithms, type of tokenization for textual input, as well as whether the researchers train their models solely on textual features.

2 Deep Learning Architectures

The second section is dedicated to a more detailed description of DL architectures used in the research.

2.1 Bidirectionality

Regular RNNs have a causal structure, the state at time t is trained only on the past information [Goodfellow et al., 2016]. Some problems, though, require

*Corresponding author, elizaveta.zinovyeva@hu-berlin.de

Table 1: Deep learning in AOB detection - literature review

Study (in chronological order)	Details	RNN	LSTM/GRU	MLP/DNN	CNN	BLSTM/BGRU	ATTENTION	CNN/LSTM Hybrid	OTHER	Hierarchy Tokenization[1]	Text Features Solely
		Algorithm								Other Details	
Potha and Maragoudakis [2014]	Time-series modeling with Singular Value Decomposition and SVM. Comparison to MLP. Perverted Justice data			x						w	1
Mehdad and Tetreault [2016]	Comparison of features on character and word level, distributional representation of comments compared to Recurrent Neural Network Language Model and Naive Bayes SVM	x								c, w	1
Zhang et al. [2016]	Pronunciation-based CNN to deal with misspellings which do not affect words' pronunciation. Aim - practical, robust, universal method with high performance. Max Pooling				x					w	1
Badjatiya et al. [2017]	Compare DL with TML: RF, SVM, GBDT, usage of pre-trained embeddings		x		x					c[2], w	1
Gao and Huang [2017]	Incorporate contextual information, Fox News User Comments		x			x	x			c, w	0
Pavlopoulos et al. [2017]	Automatic comment moderation. Camparison of different GRU based attention models with CNNs, and detox - model based on MLP and LR		x	x	x		x			w	1
Ptaszynski et al. [2017]	Convolutional neural networks used on data in Japanese from unofficial school websites and fora. Part of speech POS, named entity recognition NER features				x					w	1
Vishwamitra et al. [2017]	Two-level cyberbullying detection for mobile devices - Android appplication. Pronunciation-based CNN				x					w	1
Agrawal and Awekar [2018]	Multiple social media platforms, transfer learning classification, usage of pre-trained embeddings		x		x	x	x			c, w	1
Al-Ajlan and Ykhlef [2018]	CNN use of pretrained embeddings, Glove, Twitter data, comparison with SVM, Max Pooling. Literature divided into detection types				x					w	1
Aroyehun and Gelbukh [2018]	Usage of data augmentation, pseudo-labelling techniques, pre-trained embeddings, Facebook data in English and Hindi, DL models vs NBSVM as a non-DL baselineLSTM,		x		x	x		x		c[3], w	1
Bu and Cho [2018]	Ensemble of character level CNN vs word-level LRCN, usage of word embeddings for LCRN, max pooling for CNN		x		x			x		c, w	1
Chen et al. [2018]	2D TF-IDF vs 1D TF-IDF and pre-trained embeddings, Twittert data, Max Pooling		x		x					w	1
Dadvar and Eckert [2018]	Reproducibility study, Wikipedia, Twitter, and Formspring, YouTube datasets. Random, GloVe and SSWE embeddings		x		x	x	x			w	1
Fortuna et al. [2018]	Italian language, Facebook and Twitter data			x						w	1
Founta et al. [2019]	RNN-based networks with attention, additional metadata features. Twitter		x							c, w	0
Georgakopoulos et al. [2018]	CNN vs. different TML models on Wikipedia talk pages data				x					w	1
Ibrahim et al. [2018]	Imbalanced data, data augmentation, Wikipedia data, Ensembles CNN				x	x				w	1
Khatua et al. [2018]	Exploration of sexual violence on Twitter, gender-based violence, #MeToo movement, types of assault classification		x	x	x	x				w	1
Pitsilis et al. [2018]	Use of user-behavioral characteristics through text, knowledge of previous user's behavior, pre-trained embeddings, ensembles of LSTMs, Twitter data		x							w	0

[1] w – word-level, c – character-level, [2],[3] character level is used only for traditional ML method

Table 2: Deep learning in AOB detection - literature review (cont.)

Study (in chronological order)	Details	Algorithm: RNN LSTM/GRU	MLP/DNN	CNN	BLSTM/BGRU	ATTENTION	CNN/LSTM Hybrid	OTHER	Other Details: Tokenization[4]	Hierarchy	Text Features Solely
Polignano and Basile [2018]	Twitter, Facebook with Italian data. MLP compared to SVM, KNN, RF and other models of TML, usage of pre-trained embeddings word2vec		x						w		1
Raisi and Huang [2018]	Weak-supervision weekly, ensemble of two co-trained learners, graph-node representation, pre-trained embeddings used	x							w, d		0
Risch and Krestel [2018]	Bidirectional GRU with average pooling, English and Hindi FB data set and English Twitter, data augmentation and usage of word embeddings, ensemble with gradient boosting trees				x				c[5], w		1
Rosa et al. [2018]	Differemt CNN-based models, usage of word2vec, DL vs DVM and LR, balanced and unbalanced data			x			x	x	w		1
Tommasel et al. [2018]	Pre-trained embeddings, sentiment features using SentiWordNet corpus, composed features TF-IDF. sentiment and punctuation related features, Gaussian noise after input layer	x							c, w		1
van Aken et al. [2018]	Error-Analysis, DL models compared with LR, pre-trained embeddings, common challenges, ensemble learning, multi-label classification, Twitter and Facebook data	x		x	x	x			c[6], w		1
Zhong et al. [2018]	Cyberaggression detection using convolutional networks, differentiation between cyberbullying and cyberaggression, session level bully incident, text and image features, Instagram data max pooling layer, comparison with LR, comment level, usage of word embeddings			x					w		0
Zimmerman et al. [2018]	Ensembles of convolutional NNs, Twitter data			x					w		1
Cheng et al. [2019]	HAN, attention on word and comment level, usage of word-embeddings. Compare to KNN, NB, LR, RF, XGBoost, Instagram data	x		x		x		x	c, w	1	1
Fagni et al. [2019]	Data from Facebook and Twitter in Italian, limit feature engineering phase, Dl models and ensemble. Compare with SVM	x		x			x		w		1
Pandey et al. [2018]	Sexual Assault intent detection, Twitter data, convolutional networks, usage of Part-of-Speech tags and pre-trained embeddings			x					w		1
Santosh and Aravind [2019]	Hierarchical attention model, only one attention layer, word and syllable encoder, compare to SVM, RF, Twitter data	x				x		x	w, s	1	1

[4] w – word-level, c – character-level, s – syllable level, [5],[6] character level is used only for traditional ML method

information from the future or the whole input sentence.
BRNN is a construct of two recurrent neural networks (RNNs) proposed by Schuster and Paliwal [1997] to incorporate future temporal dynamics. The idea is to combine a forward pass - one RNN going from the first to the last state with a backward pass, which goes vice versa. Outputs from forward states are not connected with those of the backward pass. This extension of RNN allows training the network in both time directions simultaneously [Schuster and Paliwal, 1997]. Bidirectional RNN (BRNN) is trained similarly to a regular RNN. Only if back-propagation through time is used, the forward and backward procedures are more complicated, since an update of a state and output cannot be done simultaneously, special actions are required at the beginning and the

end of training data. The forward state at $t = 1$ and $t = T$ are unknown, as well as local state derivatives, and set to 0.5 and 0 respectively [Schuster and Paliwal, 1997]. Figure 1 shows BRNN's general structure.

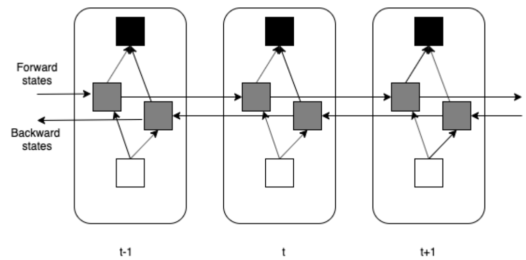

Figure 1: General structure of BRNN proposed by Schuster and Paliwal [1997]

2.2 Dimensionality reduction with attention and pooling

The original Attention Mechanism was developed as a memory extension for the encoder-decoder (E-D) architecture. Before the attention mechanism was invented, machine translation relied on encoding the whole input sequence into a one hidden state representation. Encoding data into only one hidden state representation might result in the loss of information. With an attention mechanism proposed by Bahdanau et al. [2014], the model should be able to cope with a limitation of the classical E-D model – difficulty with decoding of long sentences [Bahdanau et al., 2014]. Attentive model no longer encodes the full input sentence into a fixed-length vector – hidden state, yet, it enables the decoder to zoom or concentrate on different parts of the initial sentence while producing different elements of the output generation.

The attention mechanism used for text classification has a slightly different structure and acts as a reduction technique, such as average and max-pooling. By using the attention mechanism, we introduce an additional context vector that summarizes the input on the different time steps and is co-trained with the model. By using the scalar product of the hidden representation of the input at different time steps with the state of the context vector, we measure their similarity. This similarity score is used to weight the input [Yang et al., 2016]. As compared to pooling techniques, we introduce additional trainable parameters to the model and a "smart" way to get our model to concentrate only on the critical input.

Global Pooling Layers can be seen as a simpler alternative to the attention mechanism. Pooling layers are usually used in the context of CNNs and computer vision. Similarly to the attention layer, they also act as a reduction technique by taking the feature maps and transforming them into one vector [Lin et al., 2013]. Nonetheless, pooling layers can also be used in combination with recurrent neural networks, such as long short-term memory (LSTM) and gated recurrent units (GRU). They take the sentence matrix produced through embedding and encoding and reduces it to one vector. The most common approaches are averaging and taking the maximum along the embedding dimension, subsequently resulting in the names Global Average Pooling and Global Max Pooling.

2.3 HAN

Hierarchical attention network (HAN) was first proposed by Yang et al. [2016] for document classification task. The authors have tested it on six different datasets of different size, reflecting the hierarchical structure of the text shown a positive effect on the models' performances. This multi-leveled structure should enable to pay more or less attention to different parts of content when constructing a representation of the document. Different words are differently important by depicting the whole essence of one sentence. Moreover, sentences are differently important when we describe the meaning of the whole post or a document. Moreover, the same word can have different meanings, which depend on the context. The HAN is depicted in figure 2.

Imagine we have documents that we want to classify in some categories, e.g.,

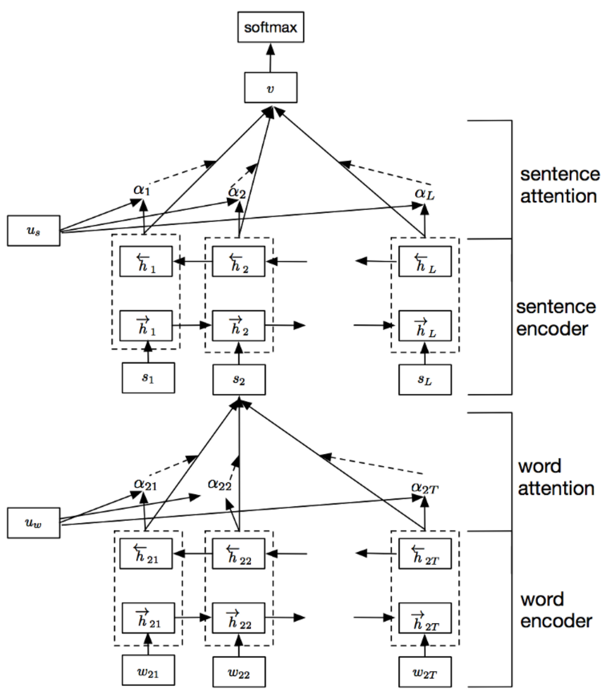

Figure 2: HAN Architecture. Proposed by Yang et al. [2016].

those about politics and those about something else. Each document has L

sentences s_i; each sentence has T_i words, whereas w_{it} are the words in the i-th sentence with $t \in [1, T]$.

Word Encoder The first part of the model is the word encoder, every document, in our case a post is broken down into sentences, where each of the sentences is encoded separately. Imagine we have a sentence s_i that has words $w_{it}, t \in [1, T]$, first each word will be embedded into an embedding matrix W_e, therefore, we obtain the x_{it} the word vector of the word t from $i - th$ sentence:

$$x_{it} = W_e w_{it}, \quad t \in [1, T]. \tag{1}$$

As described in the previous chapter, the original attention mechanism was used as an extension to the E-D architecture for solving sequence-to-sequence problems. Here, since we have a sequence-to-one problem, only the Encoder part is required. Similarly, as proposed by Bahdanau et al. [2014] an RNN network is used to encode the meaning of each word in the sentence, summarizing information of the word through the context it is used in. More precisely, the authors use a bidirectional GRU: a combination of two GRU passes networks, one GRU network makes a forward pass – reads the sentence s_i from the word w_{i1} to w_{iT}, and another GRU network – the backward pass, reads the sentence s_i from the last word back to the first one. In the end, the outputs from both passes are concatenated. GRUs are applied not to the words but to the word-vectors x_{it} resulted from embeddings, described in the previous step. The formal definition of the encoder used can be found in the equations (2), (3), (4)

$$\overrightarrow{h}_{it} = \overrightarrow{GRU}(x_{it}), \quad t \in [1, T], \tag{2}$$

$$\overleftarrow{h}_{it} = \overleftarrow{GRU}(x_{it}), \quad t \in [T, 1], \tag{3}$$

$$h_{it} = [\overrightarrow{h}_{it}, \overleftarrow{h}_{it}], \tag{4}$$

where h_{it} is the annotation for the word w_{it}: h_{it} represents the summary of the whole meaning of the sentence centered around the word w_{it}.

Word attention The next essential part if the first level of attention. As already as stated before, not every world is equally important in the understanding the essence of a sentence. Attention mechanism on a word level identifies the words that are especially important for the meaning of the sentence and enables the model to concentrate mostly on the important words. This is processed in the following manner: first, the word annotation h_{it}, obtained in the word encoder part, is fed into one-layer feed-forwarded neural network, in order to generate hidden-state representation u_{it}:

$$u_{it} = tanh(W_w h_{it} + b_w) \ , \quad t \in [1, T]. \tag{5}$$

This hidden-state u_{it} is then compared to the context vector u_w, which contains the information about the most relevant part of the input sentence at time step t (here, I refer to the time-steps as to the stepping from one to another element of the input sequence). A comparison is made in order to identify the similarity of the current hidden-state representation to the context vector at this time-step – summary of what is important in the current time-step. It is done by using the softmax function:

$$\alpha_{it} = \frac{exp(u_{it}^T u_w)}{\sum_t exp(u_{it}^T u_w)}, \quad t \in [1, T]. \tag{6}$$

Hence, it results in the α_{it} – the attention score, the normalized importance weight. In the last step, this attention score is multiplied by the initial word annotation, thus weighting the input of each word according to their importance. The sum of this weighted products is the sentence vector. The last calculation can be found in the equation (7). The authors used the notation s_i in this step again: s_i does not only represents the sentence i, but also the encoded version of the sentence i.

$$s_i = \sum_t \alpha_{it} h_{it}, \quad t \in [1, T] \tag{7}$$

The next two parts of the hierarchical attention network are very similar to the previous one. First, we start with the sentence encoder.

Sentence encoder quite similarly to the word encoder, uses bidirectional GRU in order to annotate each element of the sequence. Here, however, our elements are not the word, but already the sentences, which encoding we obtained in the two previous steps:

$$\overrightarrow{h}_i = \overrightarrow{GRU}(s_i), \quad i \in [1, L], \tag{8}$$

$$\overleftarrow{h}_i = \overleftarrow{GRU}(s_i), \quad i \in [L, 1], \tag{9}$$

$$h_i = [\overrightarrow{h}_i, \overleftarrow{h}_i]. \tag{10}$$

Note that even though the equations look very similar, we do not have the index t anymore and instead of the x_{it} we have the sentence vector s_i. The same as in the word encoder, in this step we obtain the annotation of a sentence through the context of this sentence – other sentences around the s_i

Sentence attention is used very similarly to the word attention: here we are trying to filter those sentences that mostly influence the meaning of the document at the time-step of reading. For this purpose, sentence annotation h_i is fed to another one-layered feed forward network and then its output u_i is compared to the sentence level context vector u_s using the softmax function. The resulting sentence attention scores are multiplied by the sentence annotation.

The resulting vector v represents the encoded version of the whole document. The formal steps can be found in the equations (11), (12), (13)

$$u_i = tanh(W_s h_i + b_s) \ , \quad i \in [1, L] \tag{11}$$

$$\alpha_i = \frac{exp(u_i^T u_s)}{\sum_t exp(u_i^T u_s)}, \quad i \in [1, L] \tag{12}$$

$$v = \sum_i \alpha_i h_i, \quad i \in [1, L] \tag{13}$$

Classification into categories is done in the very last step. The document vector v is fed into one-layer feed-forwarded network with a softmax layer (14). Negative log-likelihood is used as the measurement of the training loss, see (15).

$$p = softmax(W_c v + b_c) \tag{14}$$

$$L = -\sum_d log p_{dj} \tag{15}$$

2.4 Transformers

The last family of the DL models we want to concentrate on is the transformers. For the first time, they were proposed by Vaswani et al. [2017]. The models are based on the self-attention mechanism that removes the necessity of recurrence and convolution. Such structure allows higher parallelization and, therefore, the building of deeper structures that achieve state-of-the-art performance. The model, initially proposed for the sequence to sequence learning, consists of two parts: encoding and decoding component, which in their turn include six encoders and six decoders, respectively. The original architecture proposed by Vaswani et al. [2017] is depicted in figure 3.

All encoders are identical and have two sub-layers – multi-head self-attention and position-wise fully connected feed-forward network. Authors also use residual connection around every two sub-layers, which is followed by normalization layer, i.e., the input for the normalization layer is $x + Sublayer(x)$, where $Sublayer(x)$ delivers the self-attention or feed-forward network functionality. Each decoder consists of 3 sub-layers: masked multi-head attention, attention between encoder and decoder, and a feed-forward network. Similarly to encoders, each sub-layer of the decoder is surrounded with a residual connection. The self-attention mechanism is adjusted through masking to prevent learning from the subsequent input. We will not go much in detail of the self-attention mechanism. However, it is necessary to say that the self-attention mechanism allows one to attend each word with every other word in the sequence. Its multi-head nature expands the model's ability to concentrate its focus on inputs from

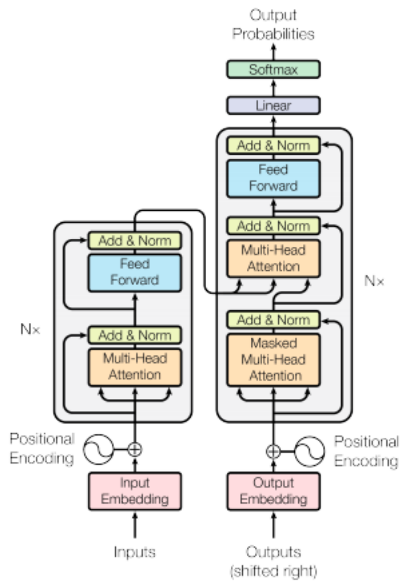

Figure 3: Transformer Architecture. Proposed by Vaswani et al. [2017]

different representations subspaces on different positions. Another vital feature of transformers is positional encoding. Since, in this family of architectures, the recurrence and convolution are no longer supported, the necessity to provide the information on the order of the input using alternative methodology arises. The transformers use sine and cosine functions to reflect the positional encoding of a token in a sequence through a sinusoid. Vectors of such encoding are added to the embeddings.

2.5 Pre-Trained Transformers

Due to high annotation costs, the majority of the datasets for text classification are rather small. Learning on small datasets is somewhat problematic for classical DL architectures. In computer vision, the standard of the recent years has been to pre-train a model on a huge dataset, so the models learn general features and then fine-tune it on a specific data task.
Until recently, the only way to use a similar approach in NLP was to pre-train the model on massive amounts on unlabelled data using word2vec [Mikolov et al., 2013], GloVe [Pennington et al., 2014] or other algorithms. The obtained vectors then initialize the first embedding layer of the DL model, which is trained on the specified task. However, recently it has become possible not only to initialize the first layer using the large corpora but also to pre-train the whole model on

unsupervised task of language modeling, which then can be fine-tuned on the labeled datasets for different NLP tasks.
One of such models is the BERT. BERT or Bidirectional Encoder Representations from Transformers is the architecture proposed by Devlin et al. [2018]. This architecture is based on transformers and is considered to be the state-of-the-art in many NLP tasks such as question answering, and language inference. BERT's architecture is based on a multi-layer bidirectional transformer encoder. BERT's training includes two tasks "mask language model" – randomly drawn tokens of the sentence are masked, then the model is trained to predict the missing word, "next sentence prediction" – a binary classification problem, where the model decides whether the sentence is the "next sentence" or not.
Another pre-trained language model we are interested in is the DistilBERT, a smaller general-purpose language representation model proposed by Sanh et al. [2019]. The authors use knowledge distillation concept to compress the BERT model while retaining the state-of-the-art performance.

3 Parameter Tuning

Meta-parameter tuning was performed. In the following table 3, the reader can find the parameter settings we tested on our models for the Wikipedia dataset. The parameters were chosen according to the previous modeling experience. For the TML models and the first experiment DL models – GRU and BGRU, the grid search procedure was performed. Due to limited computational resources, for more sophisticated models, such as attention and hierarchical models, a combination of grid and manual search was applied. After obtaining the parameters for the first dataset, we also decided to use them on other datasets to provide comparability between datasets. Only the token amount for the tokenizing procedure and the sentence lengths were corrected according to the number of tokens in the corpus and the length of the post, respectively. BERT and DistilBERT due to high computational requirement and long computational times had only 1 epoch, while other DL models were trained for 4 epochs.

Table 3: Meta-Parameters of the Models

Model	Parameters*
LR (Ridge)	C : 0.001, 0.01, 0.1, **1**, 10, 100, 1000 # Tokens (words) : 10000, 15000, 20000, 30000, **40000**, 50000
SVM	α: 0.000001, **0.00001**, 0.0001, 0.001, 0.01, 0.1 penalty: **'l1'**, 'l2' # iterations: **1000**, 10000, 15000 # Tokens (words) : 10000, 15000, 20000, 30000, **40000**, 50000
RF	# estimators : 200, 400, **600** max_depth : 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, **None** minimum_sample_split : **2**, 5, 10 max_features : **'auto'**, 'sqrt' # Tokens (words) : 10000, 15000, 20000, 30000, **40000**, 50000
GRU & BGRU	batch_size : 25, 128, **256**, 300, 500 maxlen : 100, 150, 200, **400** spatial_dropout on embeddings : **0**, 0.25, 0.5 embedding_dim : **50**, 100, 150
Continued on next page	

Model	Parameters
	rnn_dim : 50, 75, 100, **150** # Tokens (words) : 10000, 15000, 20000, 30000, **40000**, 50000
CNN	batch_size : **256** maxlen : 100, 150, 200, **400** spatial_dropout on embeddings : **0** embedding_dim : **50**, 100, 150 conv_filter : 50, 75, 100, 150, **300**, kernel_size : 3, 5, **7** padding : 'same', 'valid' # Tokens (words) : 20000, 30000, **40000**, 50000
BGRU + Att & BGRU + Avg & BGRU + Max	batch_size : 25, 128, **256**, 300, 500 maxlen : 200, **400** spatial_dropout on embeddings : 0, **0.25**, 0.5 embedding_dim : **50**, 100, 150 rnn_dim : 100, **150** # Tokens (words) : 10000, 15000, 20000, 30000, **40000**, 50000
HAN & psHAN	batch_size : 25, **256**, 500 spatial_dropout on embeddings : **0**, 0.25 # sentences: 2, **4**, 18, 25, 40 # word in sentence: 15, 25, 50, **100** # Tokens (words) : 20000, 30000, **40000**, 50000
BERT & DistilBERT	maxlen : **400** # Tokens (words) : set by the tokenizer batch_size : 32, **64**

*By using the # in the table we refer to the amount. The parameters in bold showed the best the best performance.

References

S. Agrawal and A. Awekar. Deep learning for detecting cyberbullying across multiple social media platforms. In *Advances in Information Retrieval*, pages 141–153, Cham, 2018. Springer International Publishing. ISBN 978-3-319-76941-7.

M. A. Al-Ajlan and M. Ykhlef. Deep learning algorithm for cyberbullying detection. *International Journal of Advanced Computer Science and Applications*, 9(9):199–205, 2018.

S. T. Aroyehun and A. Gelbukh. Aggression detection in social media: Using deep neural networks, data augmentation, and pseudo labeling. In *Proceedings of the First Workshop on Trolling, Aggression and Cyberbullying*, pages 90–97, 2018.

P. Badjatiya, S. Gupta, M. Gupta, and V. Varma. Deep learning for hate speech detection in tweets. In *Proceedings of the 26th International Conference on World Wide Web Companion*, pages 759–760, 2017.

D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

S.-J. Bu and S.-B. Cho. A hybrid deep learning system of CNN and LRCN to detect cyberbullying from SNS comments. pages 561–572. Springer, 2018.

J. Chen, S. Yan, and K.-C. Wong. Verbal aggression detection on twitter comments: Convolutional neural network for short-text sentiment analysis. *Neural Computing and Applications*, pages 1–10, 2018.

L. Cheng, R. Guo, Y. Silva, D. Hall, and H. Liu. Hierarchical attention networks for cyberbullying detection on the instagram social network. In *Proceedings of the 2019 SIAM International Conference on Data Mining*, pages 235–243, 2019.

M. Dadvar and K. Eckert. Cyberbullying Detection in Social Networks Using Deep Learning Based Models; A Reproducibility Study. *arXiv preprint arXiv:.08046*, 2018.

J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

T. Fagni, L. Nizzoli, M. Petrocchi, and M. Tesconi. Six things i hate about you (in italian) and six classification strategies to more and more effectively find them. In *Italian Conference on Cybersecurity*, 2019.

P. Fortuna, I. Bonavita, and S. Nunes. Merging datasets for hate speech classification in italian. In *EVALITA@ CLiC-it*, 2018.

A. M. Founta, D. Chatzakou, N. Kourtellis, J. Blackburn, A. Vakali, and I. Leontiadis. A unified deep learning architecture for abuse detection. In *Proceedings of the 10th ACM Conference on Web Science*, pages 105–114, 2019.

L. Gao and R. Huang. Detecting online hate speech using context aware models. *arXiv preprint arXiv:1710.07395*, 2017.

S. V. Georgakopoulos, S. K. Tasoulis, A. G. Vrahatis, and V. P. Plagianakos. Convolutional neural networks for toxic comment classification. In *Proceedings of the 10th Hellenic Conference on Artificial Intelligence*, page 35, 2018.

I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.

M. Ibrahim, M. Torki, and N. El-Makky. Imbalanced toxic comments classification using data augmentation and deep learning. In *2018 17th IEEE International Conference on Machine Learning and Applications*, pages 875–878, 2018.

A. Khatua, E. Cambria, and A. Khatua. Sounds of silence breakers: Exploring sexual violence on twitter. In *2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, pages 397–400, 2018.

M. Lin, Q. Chen, and S. Yan. Network in network. *arXiv preprint arXiv:1312.4400*, 2013.

Y. Mehdad and J. Tetreault. Do characters abuse more than words? In *Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 299–303, 2016.

T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

R. Pandey, H. Purohit, B. Stabile, and A. Grant. Distributional semantics approach to detect intent in twitter conversations on sexual assaults. In *2018 IEEE/WIC/ACM International Conference on Web Intelligence (WI)*, pages 270–277, 2018.

J. Pavlopoulos, P. Malakasiotis, and I. Androutsopoulos. Deep learning for user comment moderation. *arXiv preprint arXiv:1705.09993*, 2017.

J. Pennington, R. Socher, and C. 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.

G. K. Pitsilis, H. Ramampiaro, and H. Langseth. Effective hate-speech detection in Twitter data using recurrent neural networks. *Applied Intelligence*, 48(12): 4730–4742, 2018.

M. Polignano and P. Basile. Hansel: Italian hate speech detection through ensemble learning and deep neural networks. *EVALITA Evaluation of NLP and Speech Tools for Italian*, 12:224, 2018.

N. Potha and M. Maragoudakis. Cyberbullying detection using time series modeling. In *2014 IEEE International Conference on Data Mining Workshop*, pages 373–382, 2014.

M. Ptaszynski, J. K. K. Eronen, and F. Masui. Learning deep on cyberbullying is always better than brute force. In *IJCAI 2017 3rd Workshop on Linguistic and Cognitive Approaches to Dialogue Agents*, pages 3–10, 2017.

E. Raisi and B. Huang. Weakly supervised cyberbullying detection using co-trained ensembles of embedding models. In *2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, pages 479–486, 2018.

J. Risch and R. Krestel. Aggression identification using deep learning and data augmentation. In *Proceedings of the First Workshop on Trolling, Aggression and Cyberbullying*, pages 150–158, 2018.

H. Rosa, D. Matos, R. Ribeiro, L. Coheur, and J. P. Carvalho. A "deeper" look at detecting cyberbullying in social networks. In *2018 International Joint Conference on Neural Networks*, pages 1–8, 2018.

V. Sanh, L. Debut, J. Chaumond, and T. Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.

T. Y. S. S. Santosh and K. V. S. Aravind. Hate speech detection in hindi-english code-mixed social media text. In *Proceedings of the ACM India Joint International Conference on Data Science and Management of Data*, pages 310–313, 2019.

M. Schuster and K. K. Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997.

A. Tommasel, J. M. Rodriguez, and D. Godoy. Textual aggression detection through deep learning. In *Proceedings of the First Workshop on Trolling, Aggression and Cyberbullying*, pages 177–187, 2018.

B. van Aken, J. Risch, R. Krestel, and A. Löser. Challenges for toxic comment classification: An in-depth error analysis. In *Proceedings of the 2nd Workshop on Abusive Language Online*, pages 33–42, Brussels, Belgium, Oct. 2018. Association for Computational Linguistics.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

N. Vishwamitra, X. Zhang, J. Tong, H. Hu, F. Luo, R. Kowalski, and J. Mazer. Mcdefender: Toward effective cyberbullying defense in mobile online social networks. In *Proceedings of the 3rd ACM on International Workshop on Security And Privacy Analytics*, pages 37–42, 2017.

Z. Yang, D. Yang, C. Dyer, X. He, A. Smola, and E. Hovy. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489, 2016.

X. Zhang, J. Tong, N. Vishwamitra, E. Whittaker, J. P. Mazer, R. Kowalski, H. Hu, F. Luo, J. Macbeth, and E. Dillon. Cyberbullying detection with a pronunciation based convolutional neural network. In *2016 15th IEEE International Conference on Machine Learning and Applications*, pages 740–745, 2016.

H. Zhong, D. J. Miller, and A. Squicciarini. Flexible inference for cyberbully incident detection. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 356–371, 2018.

S. Zimmerman, U. Kruschwitz, and C. Fox. Improving hate speech detection with deep learning ensembles. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation*, 2018.