CEUR-WS.org/Vol-3496/autextification-paper8.pdf

Detecting Generated Text and Attributing Language Model Source with Fine-tuned Models and Semantic Understanding

Margherita Gambini[1,2,*], Marco Avvenuti[2], Fabrizio Falchi[3], Maurizio Tesconi[1] and Tiziano Fagni[1]

[1] *CNR - Institute of Informatics and Telematics, via G. Moruzzi 1, 56124, Pisa, Italy*

[2] *University of Pisa - School of Engineering, L.go Lucio Lazzarino 1, 56122 Pisa, Italy*

[3] *CNR - Institute of Information Science and Technologies, via G. Moruzzi 1, 56124, Pisa, Italy*

Abstract

The improvements in natural language generation have led to the development of sophisticated language models capable of generating long and short texts that are incredibly difficult to distinguish from human-written ones. This remarkable generative capability has spread concerns about the potential misuse of such language models, such as the spread of misinformation, plagiarism, and causing disruption in the education system. Therefore, it is important to have automatic systems to distinguish generated texts from human-authored ones (deepfake text detection), as well as recognise the language model which produced a certain text for legal and security issues (generative language model attribution). The aim of the AuTexTification challenge was to address those two tasks on texts generated by state-of-the-art language models like text-davinci-003, being one of the first versions of the powerful ChatGPT. We proposed two detection models for both tasks: fine-tuned BERTweet and TriFuseNet, a three-branched network working on stylistic and contextual features. We achieved an F1 score of 0.616 (0.565) with fine-tuned BERTweet and 0.715 (0.499) with TriFuseNet on the deepfake text detection (generative language model attribution) task. Our results emphasize the significance of leveraging style, semantics, and context to distinguish machine-generated from human-written texts and identify the generative language model source.

Keywords

deepfake text detection, NLG, machine-generated, generative source, attribution, language models

1. Introduction

Advancements in Natural Language Generation have greatly enhanced the quality, diversity, and control of texts produced by ever more powerful language models, ranging from GPT [1], GPT-2 [2], GPT-3 [3], PaLM [4], BLOOM [5] to ChatGPT [6] and BARD [7] to name a few. Despite being valuable instruments to enhance education [8] and business sectors [9], their remarkable

IberLEF 2023, September 2023, Jaén, Spain

*Corresponding author.

†These authors contributed equally.

m.gambini@iit.cnr.it (M. Gambini); marco.avvenuti@unipi.it (M. Avvenuti); f.falchi@isti.cnr.it (F. Falchi); m.tesconi@iit.cnr.it (M. Tesconi); t.fagni@iit.cnr.it (T. Fagni)

0000-0003-0640-2724 (M. Gambini); 0000-0002-8547-0348 (M. Avvenuti); 0000-0001-6258-5313 (F. Falchi); 0000-0001-8228-7807 (M. Tesconi); 0000-0003-1921-7456 (T. Fagni)

© 2023 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

CEUR Workshop Proceedings (CEUR-WS.org)

and efficient capacity to generate human-like texts serves as an additional means to manipulate public opinion by propagating false information, deceptive reviews, or untruthful opinions on the internet [10, 11], increasing phishing activities [12, 13], and undermine academic honesty [14]. Therefore, it is crucial to advance technology to automatically detect generated text and thus prevent unauthorized use. Moreover, in legal and security domains, it is insufficient to merely identify a text as machine-generated. The game-changer lies in Authorship Attribution, which involves determining the specific generative language model (among many) responsible for producing a given text [15].

Many research works addressed the detection of generated text and the attribution of the generative language model over the first released Transformer-based language models, such as GPT-2 [2], GROVER [16], XLM [17] and BART [18], usually obtaining good detection and transfer-learning performances both on long texts (such as news articles) [19, 20, 16, 21] and social media posts (e.g., tweets, reviews) [22, 23, 24, 25, 15]. However, the new generation of large language models (e.g. Google FLAN, Facebook OPT, BLOOM, ChatGPT, LLaMA) is less explored due to the recent releases [26, 27, 28]. The aim of the AuTexTification challenge [29, 30] was to enhance the development of automated detection systems to recognize text automatically generated by cutting-edge large language models, such as the Big Science's BLOOM [5] and the OpenAI Babbage, Curie and DaVinci [31]. For both English and Spanish language, the organizers provided human-written samples and texts generated on five different writing domains. Two subtasks were provided for each language: a) (subtask_1) distinguishing machine-generated texts from human-written ones, focusing on the transfer-learning capabilities of the detector, and b) (subtask_2) determining which language model generated a text. This paper describes the participation of the blade-runner team in this challenge, focusing on the English subtasks. We proposed two detectors used for both tasks: i) the jointly fine-tuned BERTweet [32] with a classification layer on top of it, and ii) TriFuseNet, a neural network consisting of three branches: one based on stylistic and textual features, another based on a language model's embeddings, and a third based on char-CNN. Our contribution was to leverage the fine-tuning effectiveness of pre-trained language models [28], as well as leveraging semantic information and contextual understanding [26]. Fine-tuned BERTweet achieved an F1 score of 0.62 (0.57), while TriFuseNet reached an F1 score of 0.72 (0.5) on subtask_1 (subtask_2). Our results encourage leveraging stylistic, semantic and contextual information to recognize machine-generated texts from human-written ones while pushing on the usage of (fine-tuned) pre-trained language models to detect the generative LM of a generated text.

The rest of the paper is organized as follows. Section 2 reviews previous works for both subtasks, that is deepfake text detection and generative language model attribution. Section 3 describes the two subtasks and the provided dataset. In Section 4, the architectures for both developed detectors are presented. Finally, Section 5 illustrates and discuss the results.

2. Related Work

During the last few years, especially with the development of the GPT language model series, the deepfake text detection research field has surged [33, 34], providing techniques to both recognize texts generated by Transformer-based language models (LM) from those written by humans and to attribute a generated text to a specific generative LM.

Deepfake Text Detection The detection of machine-generated texts has been addressed encompassing different feature engineering and classification methods. As most generative LM exhibit statistical differences in word choice with respect to humans [35, 36], several works leveraged features like perplexity [27], per-token model probability [37], and TF-IDF over n-grams [19]. Moreover, generated texts tend to have different linguistic patterns with respect to human-written ones [38], such as a lack of syntactic and lexical diversity [37, 16, 39], repetitiveness [35, 36], lack of coherence and purpose [36, 3]; Frohling et al. (2021) [21] exploited this gap and successfully trained a simple machine learning classifier using crafted linguistic features. Additionally, Zhong et al. (2020)[20] held on the generative LM's flaw of producing nonsensical or inconsistent text to train a graph-based model that leverages the factual structure of a document, emphasizing the importance of fact-verification as a critical component in deepfake text detection.

Typically, deepfake text detection is approached as a binary classification problem: standard machine learning algorithms (such as linear regression, SVM, Decision Trees and Naive Bayes) are widely used together with crafted statistical and stylistic features [19, 21], as well as with semantic textual features [40] or embeddings [22] extracted from pre-trained LMs. The machine learning algorithms can also be substituted with (deep) neural networks such as CNN, LSTM or RNN [22, 41, 42]. Furthermore, techniques that utilize LMs as a foundation are also being investigated: the concept involves jointly fine-tuning a LM with a neural network classifier on a combination of human-written and LM-generated texts, allowing to capture the implicit textual nuances [16, 19, 35, 22, 23, 26, 28].

The aim AuTextification challenge was to promote the development of models able to learn features that generalize to unseen writing styles. Closely related to this goal, Solaiman et al. (2019) [19] probed the OpenAI RoBERTa detector on texts generated by different versions of GPT-2. They discovered that the RoBERTa detector trained on top-p sampling performs well across various sampling methods, demonstrating transferability. Also, fine-tuning RoBERTa achieves higher accuracy compared to GPT-2, potentially due to superior bidirectional representations. This is in contrast with Zellers et al. (2019) [16]'s work, which suggested using the generative LM itself for detection. Furthermore, Stiff et al. (2021) [43] found that GROVER [16] and OpenAI RoBERTa detectors struggle to generalize effectively, especially for social media posts. Additionally, Crothers et al. (2022) observed that smaller LMs can effectively detect text generated by larger models, while Gambini et al. (2022) [23] proved that fine-tuning XLNET on tweets written by humans, GPT-2 and older generative techniques (Markov Chain, RNN) can effectively recognize GPT-3 tweets as machine-generated with an accuracy of 82.1%. Last but not least, Li et al. (2023) [28]'s work is close to the AuTexTification challenge since they tested the detection of deepfake texts across several writing styles (including opinion statements, scientific writing and stories) experimenting with out-domain and out-model settings; they found out that jointly fine-tuning a pre-trained language model (e.g., Longformer [44]) with classification layer obtains the best performance (with an average recall over 90%) across all testbeds. Considering these findings and aiming to develop a generalizable model, we pursued two approaches: firstly, we fine-tuned a language model and assessed its generalization capabilities over similar writing domains (e.g., tweets and reviews); secondly, we developed a neural network with three branches based on linguistic features, BERTweet embeddings, and char-CNN respectively. These branches can capture statistical, stylistic, semantic, and structural differences between human-written and generated text, providing robust and comprehensive detection capabilities [40, 26].

Generative Language Model Attribution Recognizing the LM which generated a text is a less explored research field, but not less important. Tay et al. (2020) [41] investigated the extent to which generated texts can be assigned to the correct generative LM's configuration (decoding method, model size and prompt length). Their findings reveal that even a simple classifier like Bag-of-words followed by a Multilayer Perceptron Network can predict the modeling configuration with accuracy (55.2%) higher than chance. The fact that a bag-of-words detector performs similarly to a more complex encoder-based detector (e.g., based on the transformer [45]) means that word order is less important. They also found that detecting the generative LM's configuration is harder than discriminating between human-written and machine-generated texts. These findings support Uchendu et al.'s (2020) [15] study, which shows that basic models (linguistic and psychological features inputted into traditional machine learning models or simple neural networks like CNN and RNN) perform effectively in three contexts: i) deciding whether two articles are written by the same generative LM (F1 score of 0.98), ii) distinguishing human-written from machine-generated articles (F1 score of 0.92), and iii) identifying the generative LM used to generate a given article (F1 score of 0.90). Furthermore, Munir et al. (2021) [46] explored generative LM attribution across four tasks: attributing pre-trained LMs, attributing fine-tuned LMs to parent pre-trained LMs, attributing pre-trained or fine-tuned LMs with different sampling parameters, attributing fine-tuned variants of a pre-trained LM. The probed detectors made use of stylometric features (Writeprint) as well as static (leveraging Glove) and dynamic embeddings (leveraging the last hidden state of XLNET and/or GPT-2). They showed that the attributor based on fine-tuned XLNet embeddings outperformed the other approaches with an accuracy higher than 90% over all tasks.

These previous works were conducted among the first Transformer-based language models available from 2019, that is GPT [1], GPT-2 [2], CTRL [47], GROVER [16], XLM [17], XLNet [48], BART [18], PPLM [49] and FAIR [50]. On the other hand, the second task of the AuTextification challenge targeted more recent language models like ChatGPT (a previous version was based on *text-davinci-003*). We contributed with the same approaches used for the deepfake text detection subtask but trained on a multi-class classification task: fine-tuning a language model could achieve good results, as Munir et al. (2021) [46] showed in training the XLNet embeddings followed by a softmax layer. Moreover, with the three-branched network on stylistic features, language model embeddings and a character-based CNN we combined Uchendu et al.'s (2020) [15] and Munir et al. (2021) [46] findings.

3. Tasks and Dataset

The AuTextification challenge involved five different domains to cover a wide variety of writing styles, from more structured and formal to less structured and informal: legal documents (legal), how-to articles (wiki), news articles (news), and social media (tweets, reviews). We focused solely on the English subtasks:

- *subtask_1* - Determine whether the text is automatically generated or not. To promote generalization across new writing styles, the train set involves three domains (wiki, legal, tweets), while the test set comprises two different domains (reviews, news).
- *subtask_2* - Identify the specific text generation model among six options (*bloom-1b7, bloom-3b, bloom-7b1, babbage, curie, text-davinci-003*) [5, 31] that produced a given generated text. Each class corresponds to a text generation model, with the models varying in the number of neural parameters from 2B to 175B. All five writing styles are involved in both train and test sets.

Table 1 shows the dataset's details for each subtask.

Table 1
Dataset details for both (English) subtasks. The *H* and *G* labels for subtask_1 stand for *Human* and *Generated*. The *A*, *B*, *C*, *D*, *E*, *F* labels for subtask_2 represent the *bloom-1b7, bloom-3b, bloom-7b1, babbage, curie, text-davinci-003* generation models respectively.

Subtask	Writing	Train			Test		
	Style	H	G	total	H	G	total
subtask_1	wiki	5,918	5,862	11,780	-	-	-
	tweets	5,884	5,813	11,697	-	-	-
	legal	5,244	5,124	10,368	-	-	-
	reviews	-	-	-	5,178	5,726	10,904
	news	-	-	-	5,464	5,464	10,928
	total	17,046	16,799	33,845	10,642	11,190	21,832

		Train							Test						
		A	B	C	D	E	F	total	A	B	C	D	E	F	total
subtask_2	wiki	775	781	766	798	791	749	4,660	187	195	216	195	202	214	1,209
	tweets	782	769	799	747	780	768	4,645	205	199	181	204	183	201	1,173
	legal	650	626	667	744	689	761	4,137	159	153	165	146	198	166	987
	reviews	750	791	736	806	768	777	4,628	194	155	203	171	206	195	1,124
	news	605	681	719	775	794	772	4,346	142	173	187	208	190	212	1,112
	total	3,562	3,648	3,687	3,870	3,822	3,827	22,416	887	875	952	924	979	988	5,605

Every methodology decision was based on the exploration of both train and test sets, as detailed in the Methods section. The complete train and test sets include info about the writing domain, the generative LM and the used prompt to generate texts for all provided texts.

4. Methods

Two different architectures were explored to tackle both tasks: fine-tuning BERTweet and training a neural network incorporating stylistics, sentence embeddings, and character features.

A language model pre-trained or fine-tuned on tweets was employed for both approaches. This decision was based on a random examination of the train and test sets, revealing that the maximum character length is 669 and the presence of language patterns specific to social media posts, such as abbreviations, mentions, hashtags, and informal language. Consequently, these distinctive features led us to select a language model suitable for analyzing social media texts. To the best of our knowledge, the most effective advanced language models for handling social media posts are those tailored for tweets.

To optimize the hyper-parameters, we used a 5-fold cross-validation approach on the training set. With the selected hyper-parameters we then trained the model under analysis on the complete training set. Python (3.9.15) and the NVIDIA Tesla V100 (16GB) GPU were used for all experiments.

4.1. Fine-tuning BERTweet

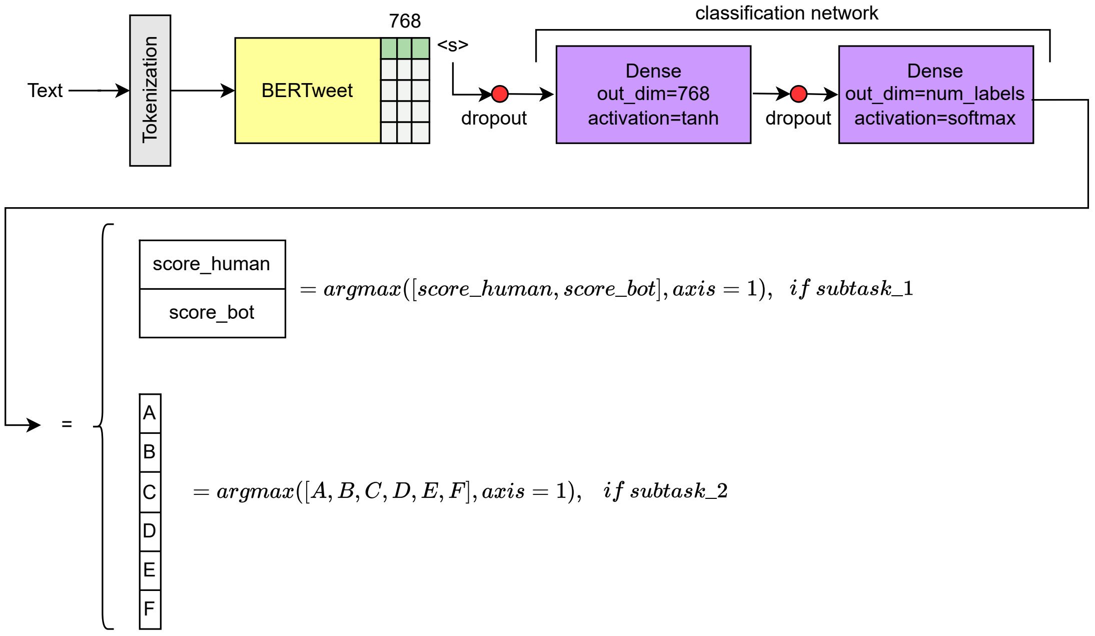
Jointly fine-tuning a Transformer-based language model followed by a fully connected neural network for classification purposes is one of the most affirmed techniques for deepfake text detection [28], especially on social media texts [22, 24, 23]. As its high performance for the detection of deepfake social media texts [24, 23] shows, BERTweet [32] (pre-trained on English tweets using the RoBERTa [51] pre-training procedure) is the optimal language model candidate. Figure 1 shows the designed architecture for both tasks. We used the same architecture for subtask_2 as well to understand its potential in attributing the correct author (language model) to the generated texts.

Firstly, a text is tokenized through BERTweet's tokenizer inserting the special classification token <*s*> at the beginning and pad tokens to reach the maximum sequence length allowed (128 tokens). Then, the tokens are fed to BERTweet, generating a 768-dimensional vector for each of the 128 tokens. The classification network takes the vector associated with the <s> token as input. It consists of a fully connected dense layer with 768 units and a tanh activation function, directly connected to a linear layer with a softmax activation. This generates probability scores for each class, and the class with the highest score is selected as the output. Dropout is applied after BERTweet and the fully connected dense layer to mitigate overfitting.

We leveraged the Python SimpleTransformers (0.63.9) and Wandb library (0.14.2) [52] to fine-tune and optimize the hyper-parameters of the described architecture. The used hyper-parameters are listed in Table 2. We tuned the highlighted hyper-parameters using the Bayesian Optimization [53] and maximizing the F1-macro score. The number of training epochs is fixed to reduce the time of the optimization phase; it was set to five as it was the best value for similar experiments with deepfake tweets [23].

4.2. TriFuseNet: Stylistic Features, Embeddings and Chars

Both determining whether a text has been automatically generated or not and detecting the generative source are authorship attribution tasks. Methods leveraging combinations of stylistic features, contextualized word embeddings from Transformer-based language models, and char-base deep neural networks are widely used for these kinds of challenges [54, 55]. Stylistic features capture the unique writing style of an author, such as sentence structure, punctuation



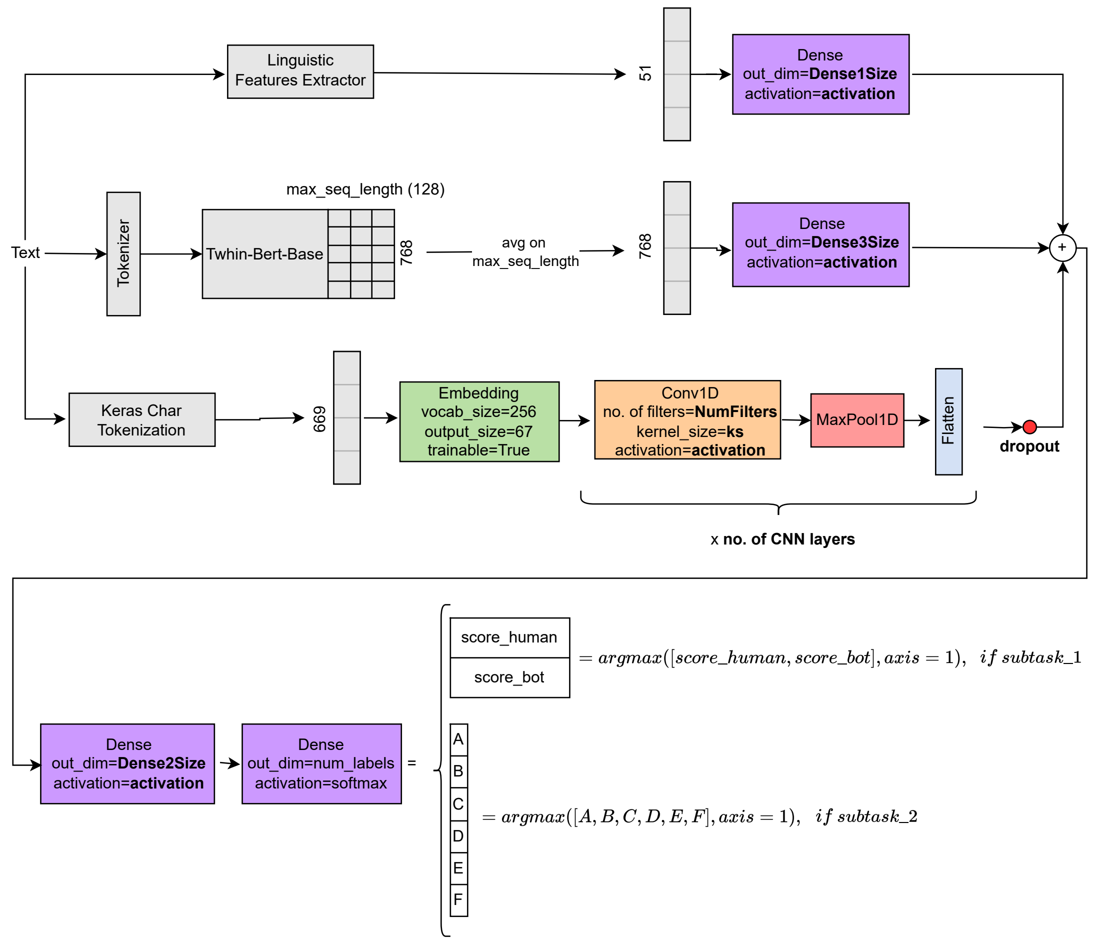
Text
Tokenization
BERTweet
768
<s>
classification network
dropout
Dense out_dim=768 activation=tanh
dropout
Dense out_dim=num_labels activation=softmax
=
score_human
score_bot
$= argmax([score\_human, score\_bot], axis = 1), \quad if\ subtask\_1$
A
B
C
D
E
F
$= argmax([A, B, C, D, E, F], axis = 1), \quad if\ subtask\_2$

Figure 1: BERTweet followed by a fully connected neural network for classification purposes. **<s>** is the classification token, which is a 768-dimensional vector representing the meaning of the entire sentence.

Table 2
Setting of the hyper-parameters used to fine-tune BERTweet. The tuned parameters are highlighted in bold. Not mentioned hyper-parameter are the default ones from HuggingFace.

Parameter	Subtask_1 Value	Subtask_2 Value	Tuned Range/ Set
Max Sequence Length	128	128	-
Batch Size	4	4	-
No. of Training Epochs	5	5	-
num_labels	2	6 -	
Optimizer	AdamW	AdamW	-
Scheduler	linear with warmup	linear with warmup	-
Dropout	0.1	0.1	-
Learning Rate	0.000031651428939296644	0.00004138122463373879	[0.0 - 1.1e-4]
Weight Decay	0.0881751204354182	0.0893495773266548	[0.0 - 0.1]
Warmup Ratio	0.0266509952435599	0.0400680159083322	[0.0 - 0.1]

usage, and vocabulary choice. Contextualized word/sentence embeddings from Transformer-based models enhance the network's understanding of the semantics and syntax of the text,



Text
Linguistic Features Extractor
51
Dense out_dim=Dense1Size activation=activation
Tokenizer
max_seq_length (128)
Twhin-Bert-Base
768
avg on max_seq_length
768
Dense out_dim=Dense3Size activation=activation
+
Keras Char Tokenization
669
Embedding vocab_size=256 output_size=67 trainable=True
Conv1D no. of filters=NumFilters kernel_size=ks activation=activation
MaxPool1D
Flatten
dropout
x no. of CNN layers
Dense out_dim=Dense2Size activation=activation
Dense out_dim=num_labels activation=softmax
=
score_human
score_bot
= argmax([score_human, score_bot], axis = 1), if subtask_1
A
B
C
D
E
F
= argmax([A, B, C, D, E, F], axis = 1), if subtask_2

Figure 2: The *TriFuseNet*'s architecture. The hyper-parameters to tune are highlighted in bold.

allowing it to capture fine-grained linguistic nuances. Additionally, a char-CNN, which operates on character-level representations, can capture information about spelling patterns, typos, and other text-level characteristics that are independent of word choice or sentence structure. By combining these various features, a neural network can leverage high-level and low-level cues to distinguish genuine text from deepfake text. Figure 2 illustrates *TriFuseNet*, the implemented neural network architecture featuring a *Stylistic-based*, a *Contextualized Sentence Embedding-based*, and a *Char-CNN* branch.

Firstly, inputs for each of the three branch are extracted from the input text:

- *Stylistic Features* - Table 3 shows the 51 extracted features.
- *Contextualized Sentence Embedding* - As a sentence embedding generator we adopted TwHIN-BERT-base [56], a multilingual language model for tweet representations that could have been used for the Spanish subtasks as well. The TwHIN-BERT-base tokenizer prepares the input for the TwHIN-BERT-base model by tokenizing the text and padding

the tokens to match the maximum sequence length of 128 tokens. Then, the TwHIN-BERT-base's last hidden state is averaged on the sequence length, obtaining a 768-dimensional features vector for the input text.

- *Tokenized Chars* - We tokenized the input text at the character level using the Keras Tokenizer and padding to the maximum number of characters (669) among the train and test sets.

Table 3
The 51 stylistic features are divided into five groups. The Python *spacy* and *textstat* packages have been used.

Type	Features Name	Normalized By
Lexical	avg_word_length num_unique_words, num_stop_words, num_upper_case_words, num_lower_case_words, num_title_case_words, num_proper_nouns, num_nouns, num_verbs, num_adjectives, num_adverbs, num_pronouns, num_named_entities, num_noun_chunks	- num_words
Syntactical	avg_num_words_per_sentence num_noun_phrases, num_verb_phrases, num_adj_phrases, num_adv_phrases, num_prep_phrases, num_coord_conj, num_subord_conj, num_coord_clauses, num_subord_clauses	- num_sentences
Semantic	num_personal_pronouns, num_possessive_pronouns, num_reflexive_pronouns, num_reciprocal_pronouns, num_quantifiers, num_determiners, num_prepositions, num_aux_verbs, num_modal_verbs, num_negations	num_words
Structural	avg_sentence_length, avg_noun_phrases_per_sentence, avg_verbs_per_sentence, proper_noun_ratio	-
Subject Specific (Readability Scores)	text_length, flesch_reading_ease, smog_index, flesch_kincaid_grade, coleman_liau_index, automated_readability_index, dale_chall_readability_score, difficult_words, linsear_write_formula, gunning_fog	-

To reduce the number of features, the Stylistic and Contextualized Sentence Embedding branches are followed by a dense layer. Instead, the Char-CNN branch consists of an embedding layer followed by a simple CNN involving convolutional and max pooling layers; the outputs of the CNN layers are concatenated in one vector. The latter is then concatenated with the outputs of the Stylistic and Contextualized Sentence Embedding branches. This intermediate vector is given in input to a classification network consisting of two dense layers with the final one having softmax activation. The latter generates probability scores for each class, and the class with the highest score is selected as the output. Dropout is applied after the CNN layers to mitigate overfitting.

We leveraged Tensorflow (2.10.1), Keras (2.10.0) and Hyperopt (0.2.7) Python packages to optimize the hyper-parameters of the described architecture and train the final model. The used hyper-parameters are listed in Table 4. We tuned the highlighted hyper-parameters using the Bayesian Optimization [53] and maximizing the F1-macro score. We also employed the Early Stopping strategy to stop training when the loss does not improve by 0.01 during 6 consecutive epochs. Therefore, the number of training epochs for optimizing the hyper-parameters is decided by the Early Stopping technique. To train the final model on the whole training set we chose the number of training epochs resulting in the best F1_macro score among the 5-folds of the optimal setting.

Table 4
Setting of the hyper-parameters used to train TriFuseNet. The tuned parameters are highlighted in bold. We used the optimizers with their default values. *FilterSizes* controls two aspects: the number of layers in the char-CNN and the kernel size *ks* for each of those layers.

Parameter	Subtask_1 Value	Subtask_2 Value	Tuned Range/ Set
Max Sequence Length (TwHIN-BERT-base)	128	128	-
Batch Size	4	4	-
No. of Training Epochs	13	17	-
num_labels	2	6	-
Dense1Size	16	16	[8,16,32]
Dense2Size	64	16	[8,15,32,64]
Dense3Size	51	8	[8,16,32,51,64][a]
NumFilters	8	8	[8,16,32,64]
FilterSizes	(3, 4, 5)	(4,)	[(3,), (4,), (5,), (6,), (2,3), (2,3,4), (3,4), (3,4,5), (2,4), (3,5)]
Dropout	0.6865683153299412	0.632083776587503	[0.1-0.7]
Activation	relu	tanh	[selu, relu, tanh, elu]
Optimizer	adam	rmsprop	[adadelta, adam, rmsprop, sgd]

[a]We erroneously fixed Dense3Size to 51. After the publication of the challenge's results we optimized TriFuseNet again using these values.

5. Results and Discussion

The AuTexTification organizers provided the results of five baselines: Logistic Regression with bag of n−grams at word and character levels, fine-tuned DeBERTa V3 [57] with default hyper-parameters, Symanto Brain [58] (Few and Zero-Shot), and a Random baseline. Table 5 shows the results for both subtasks. The F1_macro metric was adopted as the main evaluation metric for the AuTexTification challenge.

Table 5
Macro-averaged performance scores for English subtask_1 and subtask_2. The averaged 5-cross validation scores are the ones from the optimal setting of the model. The 95% confidence intervals are shown in brackets.

subtask	model	Averaged 5-cross validation			Test set		
		F1	P	R	F1	P	R
subtask_1	fine-tuned BERTweet	0.922	0.928	0.923	0.616 (0.610, 0.623)	0.793 (0.788, 0.797)	0.656 (0.651, 0.660)
	TriFuseNet	0.872	0.874	0.873	0.715 (0.710, 0.722)	0.743 (0.738, 0.749)	0.719 (0.715, 0.725)
	Logistic Regression[a]	-	-	-	0.658	-	-
	Symanto Brain (Few-Shot)[a]	-	-	-	0.594	-	-
	DeBERTa V3[a]	-	-	-	0.571	-	-
	Random[a]	-	-	-	0.500	-	-
	Symanto Brain (Zero-Shot)[a]	-	-	-	0.435	-	-
subtask_2	fine-tuned BERTweet	0.566	0.582	0.567	0.565 (0.554, 0.576)	0.580 (0.567, 0.593)	0.565 (0.554, 0.575)
	TriFuseNet	0.489	0.496	0.493	0.499 (0.486, 0.510)	0.502 (0.488, 0.517)	0.505 (0.495, 0.516)
	Logistic Regression[a]	-	-	-	0.400	-	-
	Symanto Brain (Few-Shot)[a]	-	-	-	0.290	-	-
	DeBERTa V3[a]	-	-	-	0.604	-	-
	Random[a]	-	-	-	0.167	-	-
	Symanto Brain (Zero-Shot)[a]	-	-	-	0.157	-	-

[a]baseline provided by AuTexTification organizers

Subtask_1 - Deepfake Text Detection The significant difference in results between validation and test is primarily due to the different nature of the data present in the training and test sets and the ability of the models to effectively perform transfer learning. While the TriFuseNet solution maintains an acceptable level of accuracy even in the test set thanks to the usage of general high-level and low-level linguistic cues, the fine-tuned Bertweet solution shows a drastic decrease in accuracy, which significantly affects the quality of the proposed model. Regarding writing domain detection, both models better recognize the reviews than the news (Figure 3); this is likely due to the training on tweets, which is a similar writing style domain to the reviews. Further validating the generalization capabilities of TriFuseNet, it achieves a significant F1_macro score of 0.7 in detecting generated news, surpassing the performance of fine-tuned BERTweet, which only achieves an F1_macro score of 0.497. Overall, TriFuseNet outperforms all AuTexTification's baselines, while fine-tuned BERTweet is outperformed by the Logistic Regression only. This proves that the usage of linguistic features, character-based neural networks, bag of n-grams at the word and character levels, as well as the embeddings from a LM capturing hidden linguistic cues is a good strategy for deepfake text detection across unknown writing styles.

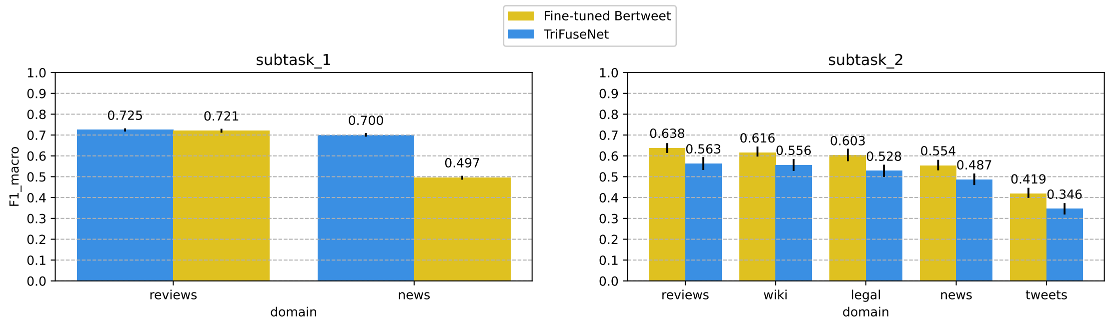


Figure 3: F1_macro score across the writing domains in the test set. F1 scores for subtask_1 are related to the binary classification of 'human' and 'generated' texts, while the scores for subtask_2 are related to the multi-class classification task attributing a generated text to its generative model among six.

Subtask_2 - Generative LM Attribution Surprisingly, fine-tuning BERTweet is the best strategy to recognize the size of the text generation model used. This suggests that the hidden characteristics of texts generated by Transformer-based language models can only be revealed by the models themselves, as the DeBERTa V3 baseline supports. This is apparently true even if the fine-tuned language model does not belong to the set of generation models used. Furthermore, the relatively lower performance of TriFuseNet compared to fine-tuning BERTweet indicates that the employed generation models may not possess markedly distinct stylistic writing patterns. However, Figure 4 reveals that our detection models have higher success in identifying texts generated by the largest model (*text-davinci-003*), suggesting distinct hidden and plain linguistic characteristics compared to the other models. Notably, the second model accurately detected is the smallest one (*bloom-1b7*), hinting at a potential similarity in writing style between *text-davinci-003* and *bloom-1b7*. As for the detection of writing domains, both our models excel in identifying the generative source LM of review texts (Figure 3). This can be attributed to the use of BERTweet and TwHIN-BERT in our models, which are trained or fine-tuned on tweets. These models demonstrate a better understanding of the writing style exhibited in reviews, which bears a resemblance to tweets rather than sentences from sources like Wikipedia, legal documents, or news articles. Despite being trained or fine-tuned on tweet data, their performance on tweets is notably lower compared to reviews, as there still exists a difficulty in recognizing the unique characteristics of short generated texts [35, 22, 23].

6. Conclusions

In this paper, we presented our participation in the AuTextification 2023 challenge, focusing on deepfake text detection and generative language model attribution for English texts. Regarding deepfake text detection, the TriFuseNet model demonstrated superior performance compared

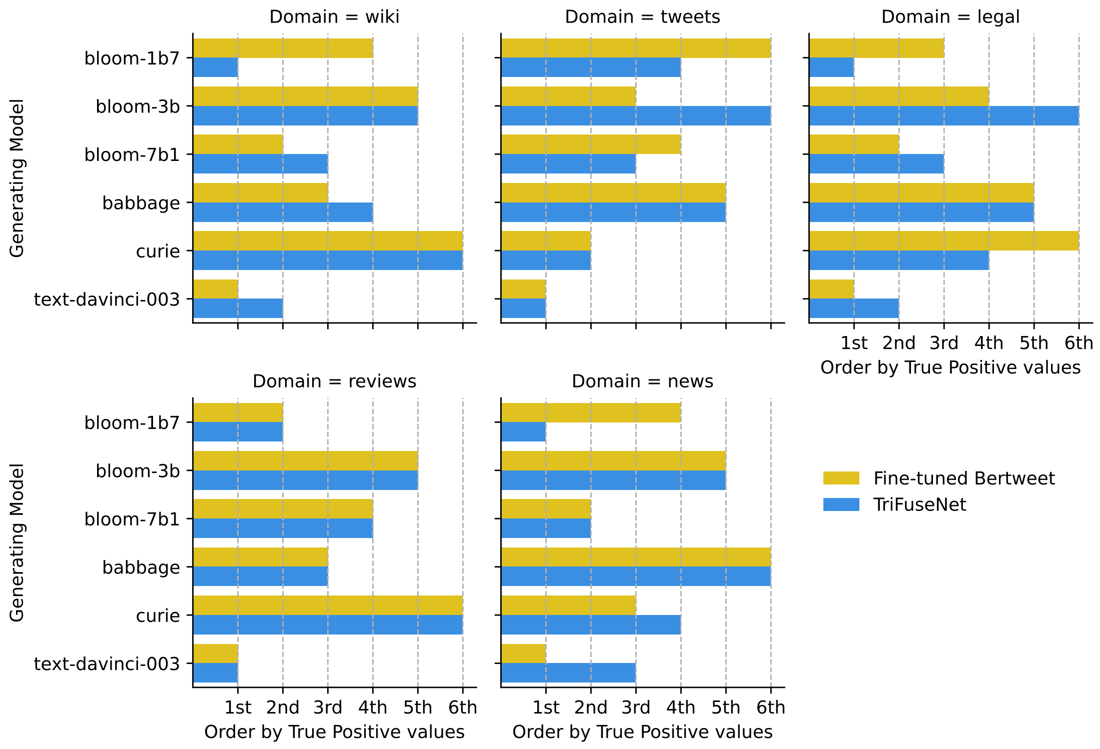


Figure 4: Recognition order of the generation models by true positive values. True positive values were first normalized by the number of examples for that specific generative model.

to baseline models, effectively detecting deepfake texts across various writing styles with an F1_macro score of 0.72. This indicates the importance of incorporating such linguistic features, character-based neural networks, and language model embeddings to capture hidden linguistic cues in deepfake text detection. Concerning generative language model attribution, fine-tuning BERTweet proves to be a better strategy than TriFuseNet for recognizing the size of the text generation model used with an F1_macro score of 0.57. This suggests that Transformer-based language models reveal the hidden characteristics of their generated texts, as the AuTexTification's DeBERTa V3 baseline supports. However, TriFuseNet successfully identifies distinct hidden and plain linguistic characteristics in texts generated by the largest model, although with slightly lower performances. The main limitation of our detection models is the usage of language models pre-trained or fine-tuned on tweets, a particular category of writing style. By exploiting more general language models, such as Longform, the results may be improved for both models and subtasks.

Overall, our findings contribute to the understanding of deepfake text detection and generative language model attribution. They emphasize the significance of linguistic features and character-based approaches in deepfake text detection and highlight the role of fine-tuning a pre-trained language model for the accurate attribution of generative language models. These insights pave the way for further advancements in fighting deepfake texts and enhancing our understanding of generative language models.

References

[1] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever, Improving language understanding by generative pre-training (2018). URL: https://s3-us-west-2.amazonaws.com/openai-assets/research-covers/language-unsupervised/language_understanding_paper.pdf.

[2] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, Language models are unsupervised multitask learners, In: OpenAI Blog [Internet], 2019. URL: https://openai.com/blog/better-language-models/.

[3] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, D. Amodei, Language models are few-shot learners, in: H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, H. Lin (Eds.), Advances in Neural Information Processing Systems, volume 33, Curran Associates, Inc., 2020, pp. 1877–1901. URL: https://proceedings.neurips.cc/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

[4] A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H. W. Chung, C. Sutton, S. Gehrmann, et al., Palm: Scaling language modeling with pathways, arXiv preprint arXiv:2204.02311 (2022).

[5] T. L. Scao, A. Fan, C. Akiki, E. Pavlick, S. Ilić, D. Hesslow, R. Castagné, A. S. Luccioni, F. Yvon, M. Gallé, et al., Bloom: A 176b-parameter open-access multilingual language model, arXiv preprint arXiv:2211.05100 (2022).

[6] Y. Liu, T. Han, S. Ma, J. Zhang, Y. Yang, J. Tian, H. He, A. Li, M. He, Z. Liu, et al., Summary of chatgpt/gpt-4 research and perspective towards the future of large language models, arXiv preprint arXiv:2304.01852 (2023).

[7] S. Pichai, An important next step on our ai journey, 2023. URL: https://blog.google/technology/ai/bard-google-ai-search-updates/.

[8] J. Qadir, Engineering education in the era of chatgpt: Promise and pitfalls of generative ai for education, in: 2023 IEEE Global Engineering Education Conference (EDUCON), IEEE, 2023, pp. 1–9.

[9] A. S. George, A. H. George, A review of chatgpt ai's impact on several business sectors, Partners Universal International Innovation Journal 1 (2023) 9–23.

[10] G. Jawahar, M. Abdul-Mageed, L. V. Lakshmanan, Automatic detection of machine generated text: A critical survey, arXiv preprint arXiv:2011.01314 (2020).

[11] L. Floridi, M. Chiriatti, Gpt-3: Its nature, scope, limits, and consequences, Minds and Machines 30 (2020) 681–694.

[12] S. S. Roy, K. V. Naragam, S. Nilizadeh, Generating phishing attacks using chatgpt, arXiv preprint arXiv:2305.05133 (2023).

[13] K. T. Gradon, Electric sheep on the pastures of disinformation and targeted phishing campaigns: The security implications of chatgpt, IEEE Security and Privacy 21 (2023) 58–61. doi:10.1109/MSEC.2023.3255039.

[14] M. Elsen-Rooney, Nyc education department blocks chatgpt on school devices, networks, 2023. URL: https://ny.chalkbeat.org/2023/1/3/23537987/nyc-schools-ban-chatgpt-writing-artificial-intelligence.

[15] A. Uchendu, T. Le, K. Shu, D. Lee, Authorship attribution for neural text generation, in: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), Association for Computational Linguistics, Online, 2020, pp. 8384–8395. doi:10.18653/v1/2020.emnlp-main.673.

[16] R. Zellers, A. Holtzman, H. Rashkin, Y. Bisk, A. Farhadi, F. Roesner, Y. Choi, Defending against neural fake news, in: Proceedings of the 33rd International Conference on Neural Information Processing Systems, Curran Associates Inc., Red Hook, NY, USA, 2019.

[17] G. Lample, A. Conneau, Cross-lingual language model pretraining, arXiv preprint arXiv:1901.07291 (2019). URL: https://arxiv.org/abs/1901.07291.

[18] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, L. Zettlemoyer, BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension, in: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, Association for Computational Linguistics, Online, 2020, pp. 7871–7880. URL: https://aclanthology.org/2020.acl-main.703. doi:10.18653/v1/2020.acl-main.703.

[19] I. Solaiman, M. Brundage, J. Clark, A. Askell, A. Herbert-Voss, J. Wu, A. Radford, J. Wang, Release strategies and the social impacts of language models, arXiv:1908.09203 [Preprint], 2019.

[20] W. Zhong, D. Tang, Z. Xu, R. Wang, N. Duan, M. Zhou, J. Wang, J. Yin, Neural deepfake detection with factual structure of text, in: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), Association for Computational Linguistics, Online, 2020, pp. 2461–2470. URL: https://aclanthology.org/2020.emnlp-main.193. doi:10.18653/v1/2020.emnlp-main.193.

[21] L. Fröhling, A. Zubiaga, Feature-based detection of automated language models: tackling gpt-2, gpt-3 and grover, PeerJ Computer Science 7 (2021) e443.

[22] T. Fagni, F. Falchi, M. Gambini, A. Martella, M. Tesconi, Tweepfake: About detecting deepfake tweets, Plos one 16 (2021) e0251415.

[23] M. Gambini, T. Fagni, F. Falchi, M. Tesconi, On pushing deepfake tweet detection capabilities to the limits, in: Proceedings of the 14th ACM Web Science Conference 2022, WebSci '22, Association for Computing Machinery, New York, NY, USA, 2022, p. 154–163. URL: https://doi.org/10.1145/3501247.3531560. doi:10.1145/3501247.3531560.

[24] S. M. Saravani, I. Ray, I. Ray, Automated identification of social media bots using deepfake text detection, in: Information Systems Security: 17th International Conference, ICISS 2021, Patna, India, December 16–20, 2021, Proceedings, Springer-Verlag, Berlin, Heidelberg, 2021, p. 111–123. URL: https://doi.org/10.1007/978-3-030-92571-0_7. doi:10.1007/978-3-030-92571-0_7.

[25] P. Kowalczyk, M. Röder, A. Dürr, F. Thiesse, Detecting and understanding textual deepfakes in online reviews (2022).

[26] J. Pu, Z. Sarwar, S. M. Abdullah, A. Rehman, Y. Kim, P. Bhattacharya, M. Javed, B. Viswanath, Deepfake text detection: Limitations and opportunities, in: 2023 2023 IEEE Symposium on Security and Privacy (SP) (SP), IEEE Computer Society, Los Alamitos, CA, USA, 2023, pp. 1613–1630. URL: https://doi.ieeecomputersociety.org/10.1109/SP46215.2023.00002. doi:10.1109/SP46215.2023.00002.

[27] E. Mitchell, Y. Lee, A. Khazatsky, C. D. Manning, C. Finn, Detectgpt: Zero-shot machine-

generated text detection using probability curvature, arXiv preprint arXiv:2301.11305 (2023).

[28] Y. Li, Q. Li, L. Cui, W. Bi, L. Wang, L. Yang, S. Shi, Y. Zhang, Deepfake text detection in the wild, arXiv preprint arXiv:2305.13242 (2023).

[29] A. M. Sarvazyan, J. Á. González, M. Franco Salvador, F. Rangel, B. Chulvi, P. Rosso, Overview of autextification at iberlef 2023: Detection and attribution of machine-generated text in multiple domains, in: Procesamiento del Lenguaje Natural, Jaén, Spain, 2023.

[30] S. M. Jiménez-Zafra, F. Rangel, M. Montes-y Gómez, Overview of IberLEF 2023: Natural Language Processing Challenges for Spanish and other Iberian Languages, Procesamiento del Lenguaje Natural 71 (2023).

[31] O. AI, Open ai models, 2023. URL: https://platform.openai.com/docs/models.

[32] D. Q. Nguyen, T. Vu, A. T. Nguyen, BERTweet: A pre-trained language model for English Tweets, in: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, 2020, pp. 9–14.

[33] E. Crothers, N. Japkowicz, H. Viktor, Machine generated text: A comprehensive survey of threat models and detection methods, arXiv preprint arXiv:2210.07321 (2022).

[34] R. Tang, Y.-N. Chuang, X. Hu, The science of detecting llm-generated texts, arXiv preprint arXiv:2303.07205 (2023).

[35] D. Ippolito, D. Duckworth, C. Callison-Burch, D. Eck, Automatic detection of generated text is easiest when humans are fooled, arXiv preprint arXiv:1911.00650 (2019).

[36] A. Holtzman, J. Buys, L. Du, M. Forbes, Y. Choi, The curious case of neural text degeneration, arXiv preprint arXiv:1904.09751 (2019). URL: https://arxiv.org/abs/1904.09751, read "TEXT GENERATION DECODING STRATEGIES" section.

[37] S. Gehrmann, H. Strobelt, A. Rush, GLTR: Statistical detection and visualization of generated text, in: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations, Association for Computational Linguistics, Florence, Italy, 2019, pp. 111–116. URL: https://aclanthology.org/P19-3019. doi:10.18653/v1/P19-3019.

[38] B. Guo, X. Zhang, Z. Wang, M. Jiang, J. Nie, Y. Ding, J. Yue, Y. Wu, How close is chatgpt to human experts? comparison corpus, evaluation, and detection, arXiv preprint arXiv:2301.07597 (2023).

[39] A. See, A. Pappu, R. Saxena, A. Yerukola, C. D. Manning, Do massively pretrained language models make better storytellers?, arXiv preprint arXiv:1909.10705 (2019).

[40] E. Crothers, N. Japkowicz, H. Viktor, P. Branco, Adversarial robustness of neural-statistical features in detection of generative transformers, in: 2022 International Joint Conference on Neural Networks (IJCNN), 2022, pp. 1–8. doi:10.1109/IJCNN55064.2022.9892269.

[41] Y. Tay, D. Bahri, C. Zheng, C. Brunk, D. Metzler, A. Tomkins, Reverse engineering configurations of neural text generation models, in: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, Association for Computational Linguistics, Online, 2020, pp. 275–279. doi:10.18653/v1/2020.acl-main.25.

[42] S. G. Tesfagergish, R. Damaševičius, J. Kapočiūtė-Dzikienė, Deep fake recognition in tweets using text augmentation, word embeddings and deep learning, in: O. Gervasi, B. Murgante, S. Misra, C. Garau, I. Blečić, D. Taniar, B. O. Apduhan, A. M. A. C. Rocha, E. Tarantino, C. M. Torre (Eds.), Computational Science and Its Applications – ICCSA 2021,

Springer International Publishing, Cham, 2021, pp. 523–538.

[43] H. Stiff, F. Johansson, Detecting computer-generated disinformation, International Journal of Data Science and Analytics (2021) 1–21.

[44] I. Beltagy, M. E. Peters, A. Cohan, Longformer: The long-document transformer, arXiv preprint arXiv:2004.05150 (2020).

[45] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, I. Polosukhin, Attention is all you need, in: Proceedings of the 31st International Conference on Neural Information Processing Systems, NIPS'17, Curran Associates Inc., Red Hook, NY, USA, 2017, p. 6000–6010.

[46] S. Munir, B. Batool, Z. Shafiq, P. Srinivasan, F. Zaffar, Through the looking glass: Learning to attribute synthetic text generated by language models, in: Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume, Association for Computational Linguistics, Online, 2021, pp. 1811–1822. URL: https://aclanthology.org/2021.eacl-main.155. doi:10.18653/v1/2021.eacl-main.155.

[47] N. S. Keskar, B. McCann, L. R. Varshney, C. Xiong, R. Socher, Ctrl: A conditional transformer language model for controllable generation, arXiv:1909.05858 [Preprint], 2019.

[48] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. Salakhutdinov, Q. V. Le, Xlnet: Generalized autoregressive pretraining for language understanding, in: Proceedings of the 33rd International Conference on Neural Information Processing Systems, Curran Associates Inc., Red Hook, NY, USA, 2019.

[49] S. Dathathri, A. Madotto, J. Lan, J. Hung, E. Frank, P. Molino, J. Yosinski, R. Liu, Plug and play language models: A simple approach to controlled text generation, arXiv preprint arXiv:1912.02164 (2019).

[50] N. Ng, K. Yee, A. Baevski, M. Ott, M. Auli, S. Edunov, Facebook fair's wmt19 news translation task submission, arXiv preprint arXiv:1907.06616 (2019).

[51] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, V. Stoyanov, Roberta: A robustly optimized bert pretraining approach, arXiv preprint arXiv:1907.11692 (2019).

[52] T. Rajapakse, Simple transformers, 2021. URL: https://simpletransformers.ai/.

[53] S. Paul, Bayesian hyperparameter optimization - a primer, 2020. URL: https://wandb.ai/site/articles/bayesian-hyperparameter-optimization-a-primer.

[54] J. Tyo, B. Dhingra, Z. C. Lipton, On the state of the art in authorship attribution and authorship verification, arXiv preprint arXiv:2209.06869 (2022).

[55] E. Lukin, J. C. Roberts, D. Berdik, E. Mugar, P. Juola, Adjectives and adverbs as stylometric analysis parameters, International Journal of Digital Humanities (2023) 1–13.

[56] X. Zhang, Y. Malkov, O. Florez, S. Park, B. McWilliams, J. Han, A. El-Kishky, Twhin-bert: A socially-enriched pre-trained language model for multilingual tweet representations, arXiv preprint arXiv:2209.07562 (2022).

[57] P. He, J. Gao, W. Chen, Debertav3: Improving deberta using electra-style pre-training with gradient-disentangled embedding sharing, 2021. arXiv:2111.09543.

[58] Symanto, Symanto brain, 2023. URL: https://www.symanto.com/nlp-tools/symanto-brain/.