A Comparative Study of ML Approaches for Detecting AI-Generated Essays

Mihai Nechita and Madalina Raschip[a]
"Alexandru Ioan Cuza" University of Iasi, Romania

Keywords: AI-Generated Text Detection, LLM, Transformers, DeBERTa, Zero-Shot Learning, NLP, Explainability.

Abstract: Recent advancements in generative AI introduced a significant challenge to academic credibility and integrity. The current paper presents a comprehensive study of traditional machine learning methods and complex neural network models such as recurrent neural networks and Transformer-based models to detect AI-generated essays. A two-step training of the Transformer-based model was proposed. The aim of the pretraining step is to move the general language model closer to our problem. The models used obtain a good AUC score for classification, outperforming the SOTA zero-shot detection approaches. The results show that Transformer architectures not only outperform other methods on the validation datasets but also exhibit increased robustness across different sampling parameters. The generalization to new datasets as well as the performance of the models at a small level of FPR was evaluated. In order to enhance transparency, the explainability of the proposed models through the LIME and SHAP approaches was explored.

1 INTRODUCTION

The rapid development of AI systems in recent years produced models capable of generating content almost indistinguishable from human created ones. While such instruments are key to increasing productivity by automating some of the repetitive tasks, excessive usage can undermine the proper development of human creativity and critical thinking, essential aspects in education. In the academic setting, the increasing prevalence of students using artificial intelligence for their home assignments raises concerns about the authenticity and integrity of their work, which may not be entirely original, although appearing as such.

Although GPT-3 (Brown et al., 2020) was made publicly available in 2021, only after releasing its chatbot SaaS variant a year later, we saw a surge in popularity, indicating that specialized knowledge was a limiting factor for broad adoption. Open models such as LLama (Touvron et al., 2023), Mistral (Jiang et al., 2023), Phi (Gunasekar et al., 2023) offer variants that can run in inference mode on most edge devices. The development of quantization methods (Dettmers et al., 2024) further reduced hardware requirements, making it easier to deploy personal AI assistants. It is now easier than ever to interact with and get a response from a Large Language Model (LLM), with some users preferring it over Google.

In recent years, large language models have taken over the Natural Language Processing (NLP) space, replacing previous recurrent neural network architectures and encoder-only Transformers like BERT (Bidirectional Encoder Representation from Transformers) (Kenton, 2019) on a wide range of applications. One such task where LLMs particularly excel is Question Answering and Text generation, due to the increased number of parameters and larger context window. To put things in perspective, while the original BERT had a window of 512 tokens, with later developments bringing that number to 2048 tokens, LLMs such as GPT-4 now reach 32768 token windows. The increased context window allows them to fully process longer documents and generate long sequences with a lower risk of hallucination.

As such, it is essential to develop reliable systems capable of identifying and signaling misuse with great accuracy, in order to protect the credibility and integrity of academic institutions.

In this work, we are comparing multiple approaches for detecting generated content, from classical approaches based on statistical models that can run solely on the CPU to more complex architectures such as recurrent neural networks and Transformers

[a] https://orcid.org/0000-0003-0020-636X

Nechita, M., Raschip and M.
A Comparative Study of ML Approaches for Detecting AI-Generated Essays.
DOI: 10.5220/0013570200003967
In *Proceedings of the 14th International Conference on Data Science, Technology and Applications (DATA 2025)*, pages 144-155
ISBN: 978-989-758-758-0; ISSN: 2184-285X

(Vaswani et al., 2017) that require dedicated hardware and are considerably more expensive to run. By comparing these models, we aim to identify the most efficient model that could be applied on a larger scale. Another element of novelty brought by the current work is the two-stage training process used for DeBERTa, which is introduced to obtain a more effective classification. The model is not trained directly on the essays dataset but first on a large corpus of human-written and AI-generated texts, for a small number of iterations. Moreover, the comparison of the classical approaches against zero-shot learning approaches yielded some interesting observations. To shed some light inside black-box models like artificial neural networks, we explored explainability approaches such as LIME (Local Interpretable Model-Agnostic Explanations) (Ribeiro et al., 2016) and SHAP (SHapley Additive exPlanations) (Lundberg and Lee, 2017).

The remaining paper is organized as follows: after discussing some of the related work (Section 2), we present some theoretical aspects of the techniques utilized in this article (Section 3). The following sections focus on describing our methodology in approaching the problem (Section 4) and the experimental setup with the obtained results (Section 5). We explore the interpretability in Section 6. Section 7 concludes the paper with a summary, information about future work, and an acknowledgment of our limitations.

2 RELATED WORK

In the last few years, after the release of powerful LLMs, an increase in the detection of AI-written content has been noticeable. Multiple approaches, starting from statistical methods to actual complex neural networks, were proposed to detect AI generated texts. A survey on the detection of LLMs-generated content is given in (Yang et al., 2023).

DetectGPT (Mitchell et al., 2023) analyzes the likelihood landscape of text within the same language model that may have generated it. This detection method uses differences in log-probabilities after applying various perturbations, such as rewriting some phrases using a Transformer-based model. Another similar approach is GPTZero (Tian and Cui, 2023), which uses the sequence perplexity to distinguish AI content from human-written text under the assumption that human writing exhibits greater randomness. AI-generated content typically has lower perplexity. Another scoring metric used is burstiness, which assesses whether repeated tokens are likely to appear in proximity to each other. A classifier-based approach that uses a suite of NLP features, n-gram features, topic modeling features, and various readability scores is described in (Nguyen et al., 2023). The performance of different classes of detectors, including watermarking-based (Kirchenbauer et al., 2023), neural network based, zero-shot based, and retrieval-based detectors, is analyzed in (Sadasivan et al., 2023). The study raised concerns about the reliability of these detection methods.

Detecting AI-generated essays is more challenging than plagiarism detection since it is impossible to identify the "original" content. The widespread availability of SaaS platforms such as ChatGPT [1], combined with their writing capabilities, often surpassing those of an average student, has significantly increased the difficulty of this problem. Two zero-shot methods, Ghostbuster (Verma et al., 2024) and Binoculars (Hans et al., 2024), are among the current SOTA approaches for detecting AI-generated content such as essays. Ghostbuster uses language model probabilities to distinguish between human-written and AI-generated content. It identifies statistical inconsistencies that emerge in AI-generated text. Binoculars, on the other hand, employs a contrastive approach by comparing the target text against human-written and AI-generated text distributions. It examines features such as perplexity and cross-perplexity to determine whether the text is AI-generated.

3 THEORETICAL ASPECTS

3.1 Word Embeddings

To enable a model to process textual information, it must first be transformed into numerical vectors.

TF-IDF (Term Frequency-Inverse Document Frequency) (Joachims, 1997) measures the importance of a word t in a document d from a collection of documents D, adjusted by how frequent the term is in general. It is defined by a two-part formula. The first part, term frequency, measures the relative frequency of term t in document d. The second part, inverse document frequency, measures how common or rare the term is across the document corpus D. A high TF-IDF value indicates that the term appears frequently in the given document while being relatively infrequent in the rest of the corpus.

Word2Vec (Mikolov et al., 2013) is a self-supervised, shallow neural network designed to capture the similarities and semantic analogies between

[1] Introducing ChatGPT, https://openai.com/index/chatgpt/

different words. The training process learns conditional probabilities of a target word given its context.

While the previously described approaches use word-level tokenization, Byte Pair Encodings (BPE) (Sennrich et al., 2015) operates at the character level, recursively merging the most frequent pairs of consecutive tokens and replacing them in the vocabulary. This process is repeated until the desired vocabulary size is achieved. WordPiece (Kenton, 2019) is similar to BPE, with the primary difference being in the merging algorithm. Instead of merging the most frequent token pairs, it computes a score by also taking into account the individual frequency of the two components. This way the approach prevents merging individual tokens that appear very often throughout the corpus.

3.2 Recurrent Neural Networks

Multi-layer perceptrons struggle to handle temporal dependencies in sequential data because they consider data points as independent. They cannot process sequences of varying length and are limited in capturing long-term dependencies.

Recurrent Neural Networks (RNNs) come as a solution to these issues: instead of modeling the probability of token x_t at time step t as $P(x_t|x_{t-1},...,x_1)$, they use a latent hidden space h_{t-1} that compresses the information such that

$$P(x_t|x_{t-1},...,x_1) \approx P(x_t|h_{t-1})$$

.

Long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997) networks are a type of recurrent neural networks in which the recurrent node is replaced by a memory cell containing an internal state. LSTM introduce the concept of gated hidden states to mitigate the vanishing gradient problem. These gates act as mechanisms that dictate to what degree the hidden states should be updated with new information and the amount of old information that should be discarded. LSTMs use three distinct types of gates, namely the *input* gate, the *forget* gate, and the *output* gate. The amount of information that is taken into account is governed by the input gate, while the amount that is forgotten is controlled by the forget gate.

Gated Recurrent Units (GRU) (Dey and Salem, 2017) are easier to compute and often achieve comparable results with LSTMs. The difference lies in the type of gates used, namely GRU uses only two types of gates, the *reset* and *update* gates. The reset gate determines how much of the past information to forget, while the update gate controls how much of the previous memory to keep. Reset gates allow GRU to capture short-term dependencies, while update gates help in detecting long-term dependencies in sequences.

3.3 Transformers

Since their introduction in the famous *Attention Is All You Need* paper (Vaswani et al., 2017), Transformers have revolutionized the natural language processing field by expanding on the attention mechanism (Bahdanau et al., 2016). They follow an encoder-decoder architecture and offer better parallelism via attention, compared to traditional RNNs which process sequences in order. The key components of a Transformer are given below.

Self-Attention

The idea behind attention is that token embeddings should vary depending on the context. To achieve this, three matrices Q, K, and V which represent different projections of the same input, are used. The query represents what the model is looking for in the sentence, and the key-value pair acts like a lookup table. The attention mechanism that Transformers use is the *scaled dot-product* attention, a variant of self-attention, i.e. the keys and values are the same. The formula used is given below:

$$Attention(Q,K,V) = softmax(\frac{QK^T}{\sqrt{d_k}})V$$

where d_k is the length of keys and queries.

Multi-Head Attention

Rather than performing a single attention on query and key-value pairs, linear projections are used to reduce the dimensionality to smaller values and do multiple attention passes at the same time, increasing parallelism. The outputs of the smaller attention blocks are concatenated, and a weighted sum produces the final output. In terms of performance, it is similar to running the attention on the entire input, but with a much lower computational cost.

Positional Encoding

In order to preserve information about the order of the tokens, the inputs pass through a positional encoding layer which adds positional embeddings described by sine and cosine functions.

The Transformer-based model used in the current paper is DeBERTaV3, a BERT-like model. BERT

(Kenton, 2019) uses only the encoder part from Transformer which is trained in a self supervised fashion with two strategies.

1. Masked Language Modelling: For this objective, around 15% of words in each sequence are replaced by a *[MASK]* token. The model has to predict the hidden token using the other tokens in the sequence.
2. Next Sentence Prediction: In the training process, the model receives pairs of sentences, and it has to predict whether the second sentence follows the former in the original document.

The output of the *[CLS]* token is used for a classification task.

DeBERTa (He et al., 2021b) comes with two major improvements: a new attention mechanism called disentangled attention, and it incorporates absolute positioning in the decoder layer. Disentangled attention aims to separate the content embedding information from the positional embedding, each word now carrying two separate vectors. The attention weights are calculated using disentangled matrices and are equal to the sum of content-to-content, content-to-position, position-to-content, and position-to-position attention scores.

In DeBERTaV3 (He et al., 2021a), the masked language modeling task is replaced with a more efficient one, token detection.

4 METHODOLOGY

In this section, we describe the model architectures used for detection.

4.1 CPU Models

Model choices were limited to XGBoost, Logistic Regression, and SGDClassifier due to memory constraints. The embeddings used were TF-IDF and BPE tokenizers like WordPiece.

4.2 RNN

In our experiments, we used two RNN architectures based on LSTM and GRU. We chose a bidirectional architecture with various numbers of stacked layers and embedding sizes for the RNN based models. The long context of 1024 tokens is approaching the limit of LSTM layers due to using *sigmoid* and *tanh* activation functions. When training from scratch, the gradient sometimes vanishes, making optimization challenging.

Figure 1 showcases the architecture of a BiLSTM model which uses word2vec as text embedding with the number of hidden units of 256.

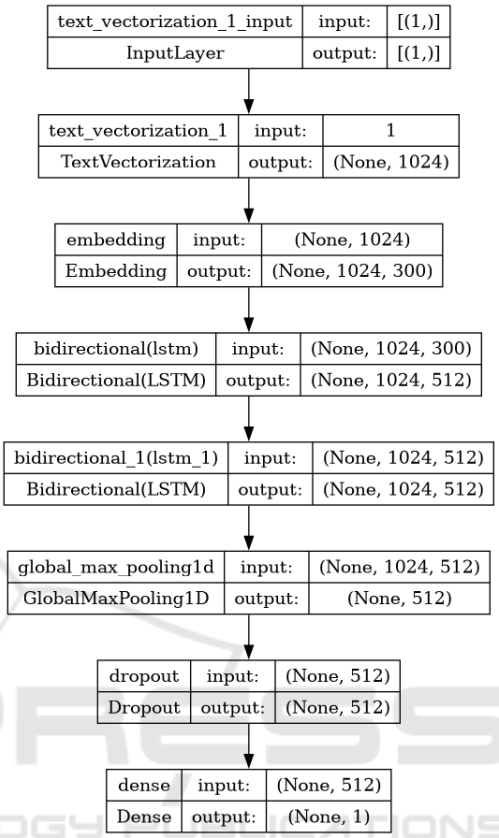

Figure 1: A BiLSTM architecture with 2 LSTM heads.

We experimented with different architectures and embeddings. We tested with a 32,000 vocabulary sized BPE tokenizer but also pretrained Word2Vec, GloVe, and FastText embeddings which ended up performing better than training our own embedding layer.

4.3 DeBERTaV3

For the Transformer-based models, we resorted to DeBERTaV3 as it's battle proven in the realm of sequence classification. The architecture used for the classification task consists of a pretrained DeBERTa model with a trainable classifier added to the final layer.

While not very helpful for smaller models, big pretrained models such as DeBERTa can show a significant performance boost from adding a pretraining step. In addition, it also helps large pretrained models get closer to the domain problem without overfitting on the smaller fine-tuning dataset. The two-stage training process used is described below.

1. First we bring the general model closer to our domain problem by training for one epoch on the "pretrain" corpus.
2. Finally, we fine-tune the model using our essay corpus.

Figure 2 shows explicitly the AUC scores depending on the training strategy. Experiments showed better results when performing a two-stage training process.

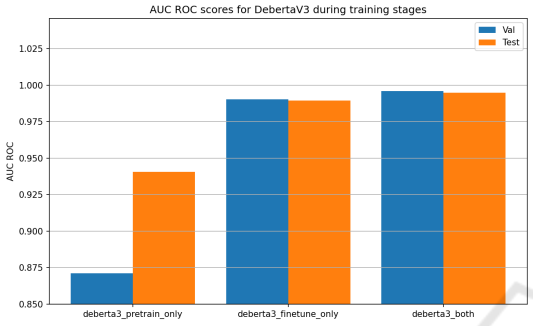

Figure 2: DeBERTa AUC scores depending on the training strategy.

We tried both, fully fine-tuning the pretrained model and using a LoRA (Hu et al., 2021) adapter approach. However, while training steps were faster with LoRA, convergence was slower, requiring multiple epochs to achieve comparable results. The results are presented in the experimental results section only for the classical fine-tuning approach.

5 EXPERIMENTAL RESULTS

This section presents the datasets used, experimental settings and results.

5.1 Datasets

Two major datasets were used, each with a specific purpose, inspired by the recent *LLM - Detect AI Generated Text* Kaggle Competition (King et al., 2023).

5.1.1 Pretraining Corpus

The "pretraining" corpus [2] was generated by the second-place winner of the competition by sampling texts from the SlimPajama (Soboleva et al., 2023) dataset. This was done by selecting random sequences of 1024 tokens and prompting LLMs to generate the next 1024 tokens (Figure 3). The real continuation ends up in the human-written pile and the LLM counterpart in the AI-generated one. This method allows us to generate an unlimited amount of AI-generated content, while also having a human-written example within the same context.

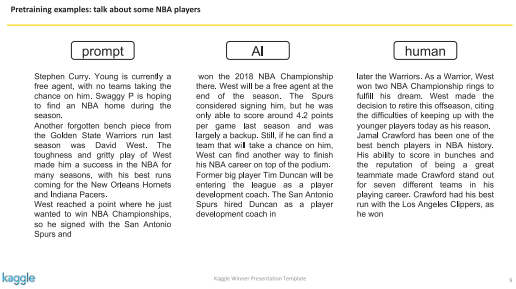

Figure 3: A random SlimPajama text sequence alongside a LLM-generated continuation and its human counterpart.

The corpus is evenly distributed with around 500k examples for each class. The LLMs used for generation were LLama2 7B, Open Llama 3B, 7B and 13B, Mistral v1 7B, Phi-2 7B, Yi-6b and Falcon 7B.

This pretraining corpus was chosen because it already aggregates publicly shared datasets from the competition and contains the least amount of AI-generated gibberish. Moreover, the methodology can be applied to any human-written dataset to obtain AI-generated counterparts.

The "pretraining" corpus was used only by the DeBERTa model, as the two-step approach had a negative effect on less complex architectures.

5.1.2 Fine-Tuning Corpus

The fine-tuning corpus (Biswas et al., 2024) is based on Persuade 2.0 (Crossley et al., 2023) which is a 25,000 argumentative essays produced by 6th-12th grade students on 15 assignments. The assignment prompts were used to generate diverse responses by prompting various LLMs (Llama, Falcon, Mistral, Mixtral, GPT-3.5, GPT-4, Claude, Cohere, Gemini, PaLM) with varying sampling parameters, such as top_k, top_p, and $temperature$.

Additionally, data from Ghostbuster (Verma et al., 2024) was incorporated. The resulting dataset consists of 167,446 curated examples, with a generated-to-human ratio of approximately 3:1 (see Table 1).

We split the combined dataset with a 90:10 ratio to create our **training** and **validation** sets. The validation dataset was used for early-stopping and threshold setting. In order to assess the ability to generalize outside of the training environment, we randomly selected an entire assignment (**Car-free cities**) with 14,000 entries and kept it as a **holdout test** set.

[2] daigtdataandcode, https://www.kaggle.com/datasets/wowfattie/daigtpretraindata/versions/5

Table 1: Fine-tuning corpus assignment distribution.

Assignment	*Human*	*Generated*	*Total*
Car-free cities	3349	10636	13985
A Cowboy Who Rode the Waves	1991	9937	11928
Cell phones at school	1959	5770	7729
Community service	1818	5505	7323
Distance learning	2558	5918	8476
Does the electoral college work?	3261	10732	13993
Driverless cars	2360	11276	13636
Exploring Venus	2342	11638	13980
Facial action coding system	2683	11307	13990
Grades for extracurricular activities	1993	5232	7225
Mandatory extracurricular activities	1994	6546	8540
Phones and driving	1375	4535	5910
Seeking multiple opinions	1881	6233	8114
Summer projects	1921	5719	7640
The Face on Mars	2147	10654	12801
ghostbuster_essays	977	3122	4099
ghostbuster_news	981	2999	3980
ghostbuster_writing prompts	1022	3075	4097
	36612	**130834**	**167446**

5.2 Linguistic Features

We analyzed the linguistic characteristics that distinguish AI-generated texts from human-written texts. Specifically, we examined vocabulary features such as average word length, vocabulary size, and word density, as well as sentiment analysis features and stylistic features, including repetitiveness and readability. The features computed for the Car-free cities assignment are shown in Table 2.

In agreement with previous studies (Muñoz-Ortiz et al., 2024), human-written texts exhibit a more diverse vocabulary but tend to be shorter in length. Unlike humans, LLMs tend to be more neutral than humans and lack emotional expression. Lexical diversity scores are lower in human-written texts. Additionally, readability scores are higher for human-written texts, suggesting the use of simpler sentence structures.

5.3 Experimental Settings

For the classical approaches, we mainly experimented with different vocabulary sizes and the N-gram range.

The RNN models were trained using *Adam* for 20 epochs. We employed *EarlyStopping* on the *validation loss* with a patience of 3 to prevent overfitting. Experiments showed that replacing *Cross Entropy Loss* with *Focal Loss* improved the model's ability to generalize on the holdout set.

The Transformer models use the official pretrained weights as base, and the randomly initialized classifier head was trained for one epoch per dataset in the two-stage training approach.

5.4 Baseline

In order to have a more accurate assessment, we compared our results with existing zero-shot methods used to solve the classification problem. *Binoculars* (Hans et al., 2024) was used as our baseline, as it claims superior performance compared to other popular methods such as Ghostbuster (Verma et al., 2024), DetectGPT (Mitchell et al., 2023), and GPTZero (Tian and Cui, 2023). Their approach builds on the idea behind Ghostbuster of using other LLM's log-probabilites as features. While Ghostbuster constructs features through arithmetic operations on token probabilities, Binoculars follows a simpler approach. It uses the ratio between the model's perplexity and another metric called *cross-perplexity*.

The authors of Binoculars used the same dataset as Ghostbuster, which is part of our corpus. Results were generated using their original choice of LLMs, *Falcon-7B* and *Falcon-7B-Instruct*.

5.5 Validation and Test Results

Table 3 presents the Area under the ROC Curve (AUC) scores for the top-performing models on each category, evaluated on the validation and holdout test (represented by **Car-free cities**) sets.

Naming convention for the RNNs follows the nomenclature:

`{model}_{embed}_{#stacked}_{hid_size}.`

We used a similar convention for the classical ML models, except for the last two values, which represent the minimum and maximum n-grams.

The logistic regression and the BiGRU model achieve good results on the validation dataset, but perform slightly worse on the holdout data. The best performing results on both datasets are obtained by the DeBERTa model with two-step training. One of the reasons the DeBERTa model outperforms the others is its training approach. In the first phase, we "specialize" the model for identifying AI-generated content. Even though we use a small number of iterations for the first phase, this ensures that the vector space is more effective for representing the data.

Table 2: Linguistic features for Car-free cities assignment.

	avg_word_length	vocab_size	word_density	empathy	lexical_div	readab
human	4.532	170.153	21.483	0.00119	0.431	56.613
generated	4.621	167.831	21.778	0.00112	0.454	52.087

Table 3: AUC scores on validation and holdout test set broken down by assignment for top performing classifiers in each category.

Assignment	*Binoculars*	*LogReg-BPE-(1,3)gram*	*BiGRU-Word2Vec-1-256*	*DebertaV3-both*
Holdout test (Car-free cities)	0.7724	0.8952	0.8716	**0.9948**
All validation	0.6754	0.9576	0.9714	**0.9957**
A Cowboy Who Rode the Waves	0.6725	0.9400	0.9598	**0.9905**
Cell phones at school	0.5271	0.9481	0.9652	**0.9994**
Community service	0.6010	0.9577	0.9702	**0.9968**
Distance learning	0.7144	0.9626	0.9748	**0.9988**
Does the electoral college work?	0.7363	0.9552	0.9683	**0.9941**
Driverless cars	0.6438	0.9504	0.9727	**0.9916**
Exploring Venus	0.6865	0.9594	0.9695	**0.9930**
Facial action coding system	0.7103	0.9448	0.9717	**0.9962**
Grades for extracurricular activities	0.5375	0.9504	0.9695	**0.9998**
Mandatory extracurricular activities	0.6195	0.9675	0.9804	**0.9960**
Phones and driving	0.5873	0.9695	0.9764	**0.9968**
Seeking multiple opinions	0.5926	0.9621	0.9672	**0.9956**
Summer projects	0.6267	0.9682	0.9827	**0.9977**
The Face on Mars	0.6753	0.9511	0.9594	**0.9936**
ghostbuster_essays	0.9979	0.9944	0.9825	**1**
ghostbuster_news	0.9863	0.9979	0.9963	**0.9996**
ghostbuster_writingprompts	0.9935	0.9933	0.9823	**1**

The performance of *Binoculars* varies by a significant degree depending on the text source, indicating that Zero-shot methods might not maintain their expected performance across different datasets. While a slight drop in holdout AUC is observed for our models, it is less pronounced.

TPR @ FPR

While relevant, AUC-ROC provides only a high-level overview of the model's performance. In our case, false positives have serious consequences, so adjusting the threshold to ensure a low FPR is essential. Figure 4 illustrates the TPR achieved at different FPR levels on a logarithmic scale on the validation and holdout test sets. Binoculars enforces an FPR of 10^{-4}; however, such a low rate turns out to be unfeasible for our corpus. Using their low FPR threshold, the method achieved a value of 5×10^{-3} on our data, while the TPR dropped to 0.35, far from their reported 0.99. This might indicate that zero-shot methods are susceptible to corpus specific patterns. For methods that use other LLMs logprobs as a proxy, the similarity between the chosen LLM and the one that generated the sequence could be important.

To compare performance at the same level of FPR, we set our models' thresholds using the validation dataset at an FPR of 5×10^{-3}, which was achieved by *Binoculars*. Table 4 summarizes the performance in terms of AUC, TPR, FPR, and F1 score for the top five models per architecture type on both validation and holdout test sets (the full table is provided in Appendix 7.2). The results are presented in sorted order of their F1 score on the validation dataset. We observe that the DeBERTa variations maintain their position as the top performers across all metrics, even on the holdout set. Other models, while outperforming the baseline Binoculars, show a significant increase in the FPR on the test data. Table 4 indicates that better performance is obtained when using a larger hidden size for the RNN models, as all of the top five variants use a hidden size of 256. In terms of the choice of RNN unit, results appear to slightly favor the simpler *GRU*. Using a deeper RNN architecture has diminishing returns, if any, 4 out of the 5 top performing RNN architectures only use a single bidirectional layer. The choice is inconclusive in terms of pretrained embedding types, both *word2vec* and *fasttext* are close in terms of validation AUC and the differences on the test set may be attributed to other choices in the model's architecture.

Effects of Nucleus Sampling

While varying nucleus sampling parameters (Holtzman et al., 2020) promotes diversity in generated content, it does not appear to affect DeBERTa's discriminative power, as shown in Figure 5. However, this

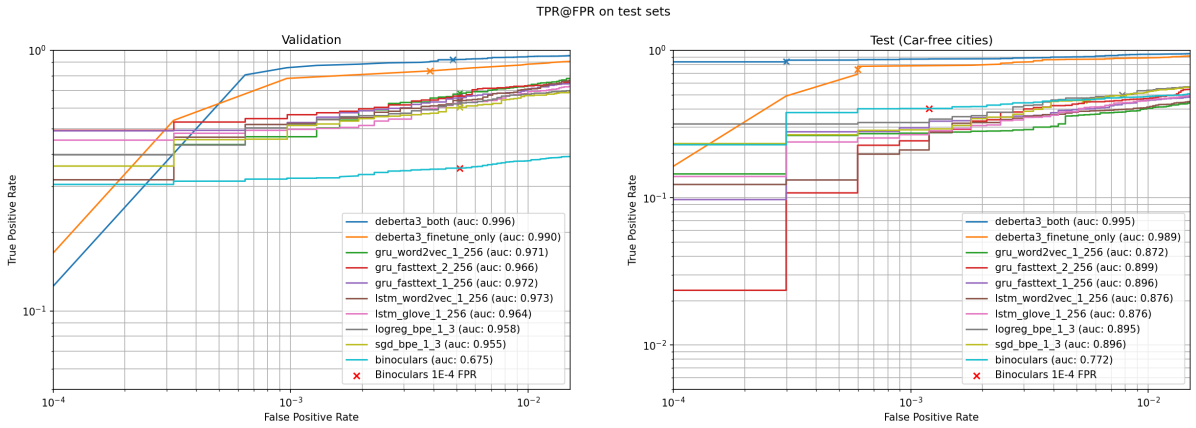

Figure 4: TPR at different levels of FPR on a logarithmic scale. X marks the threshold set at 5×10^{-3} validation FPR.

Table 4: Metrics on the validation and holdout test sets using thresholds computed at 5×10^{-3} validation FPR sorted by Validation F1.

	classifier	val_roc	val_tpr	val_f1	test_roc	test_tpr	test_fpr	test_f1	test_f1_rank
transformer	deberta3_both	**0.9959**	**0.9203**	**0.9578**	**0.9947**	**0.8367**	**0.0003**	**0.9110**	**1**
transformer	deberta3_finetune_only	0.9902	0.8336	0.9087	0.9895	0.7406	0.0006	0.8509	2
rnn	gru_word2vec_1_256	0.9714	0.6868	0.8136	0.8716	0.6949	0.1197	0.8022	3
rnn	gru_fasttext_2_256	0.9662	0.6688	0.8009	0.8992	0.5896	0.0260	0.7380	6
rnn	gru_fasttext_1_256	0.9723	0.6610	0.7952	0.8957	0.6101	0.0427	0.7516	5
rnn	lstm_word2vec_1_256	0.9731	0.6425	0.7817	0.8760	0.5773	0.0421	0.7259	7
rnn	lstm_glove_1_256	0.9637	0.6299	0.7723	0.8761	0.6587	0.0806	0.7823	4
cpu	logreg_bpe_1_3	0.9576	0.6285	0.7712	0.8952	0.4982	0.0078	0.6640	8
cpu	sgd_bpe_1_3	0.9548	0.6050	0.7532	0.8960	0.4582	0.0054	0.6277	9
cpu	xgb_bpe_1_1	0.9344	0.5309	0.6929	0.8200	0.3574	0.0272	0.5233	12
cpu	sgd_bpe_1_1	0.9396	0.5160	0.6801	0.8570	0.4496	0.0161	0.6182	10
cpu	sgd_bpe_3_5	0.9319	0.4683	0.6372	0.8629	0.3410	0.0026	0.5082	13
baseline	binoculars	0.6754	0.3533	0.5215	0.7724	0.4031	0.0012	0.5744	11
transformer	deberta3_pretrain_only	0.8711	0.0000	0.0000	0.9408	0.0000	0.0000	0.0000	14

is not the case for *Binoculars*, a pattern emerges at higher values of these parameters, completely evading detection in some instances. This highlights the inherent limitations of zero-shot methods with regard to style biases. In the presence of higher top-p and temperature values, which increase the generative model's creativity, Binoculars incorrectly flagged generated text as human written.

6 EXPLAINABILITY

Since deep models often behave like black-boxes, we explored whether known explainability techniques can shed some light into why an observation might be positively classified or not. In addition, the explanations of the predictions can help justify decisions and assist teachers in providing personalized feedback, for example.

To this end, LIME and SHAP, two widely used model-agnostic approaches were applied. LIME (Local Interpretable Model-agnostic Explanations) (Ribeiro et al., 2016) is designed to explain individual predictions of a black-box model. It first generates perturbed samples and computes their predictions using the model and then a simple interpretable model is trained on the newly generated dataset. Hence, LIME provides local interpretability. The simple model's weights represent feature importance scores.

SHAP (Shapley Additive exPlanations (Lundberg and Lee, 2017)) uses a concept borrowed from cooperative game theory, i.e. the Shapley values to quantify the contribution of the features to a prediction. The technique computes the Shapley values by considering different combinations of input features and then it assesses their impact on the model's output.

While analyzing the corpus of false positives, we identified a noisy observation consisting of a repeated sentence, which we used for our explainability experiments.

Figure 6 presents SHAP values for the DeBERTaV3 classifier on the noisy observation before and after cleaning. Since SHAP is not supported for

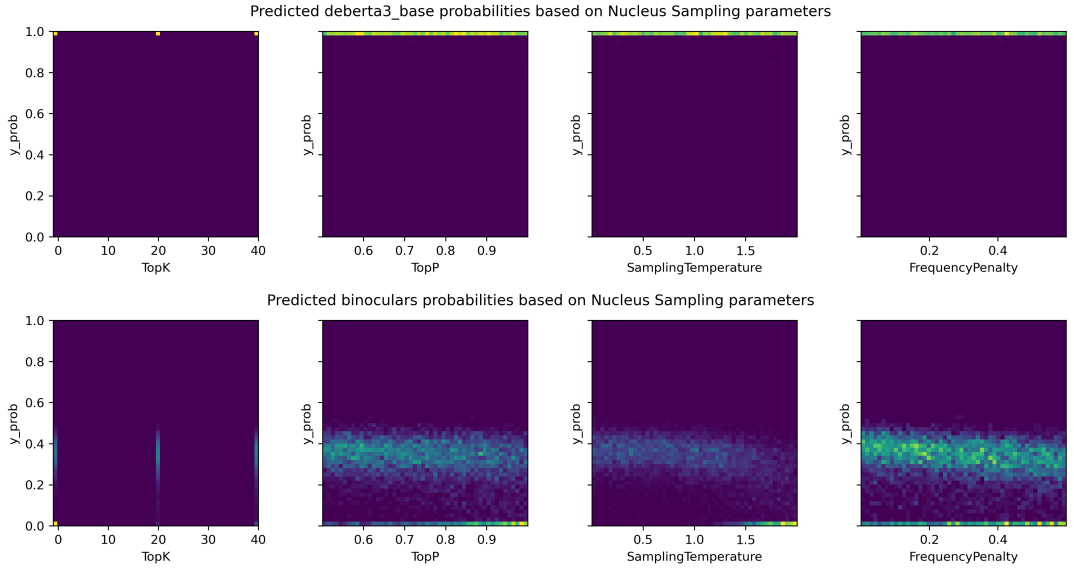

Figure 5: Effect on DeBERTaV3 and Binoculars predicted probabilities by varying nucleus sampling parameters on generated observations.

RNNs, we instead generated LIME explanations on the same texts. They are shown in Figure 7.

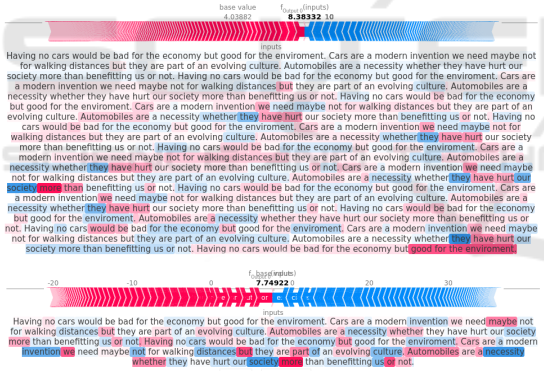

Figure 6: SHAP values for DeBERTaV3 on a noisy observation before and after correction.

In both cases, cleaning the noisy observation led to a decrease in the predicted probability, dropping below the threshold for the RNN model. Regarding the contribution of specific words, the two techniques show different behaviors, except for *"Automobiles"* which is hinting towards the AI generated class in both representations.

7 CONCLUSIONS

The problem of detection of AI-generated text is analyzed in this paper. Various machine learning approaches have been proposed to distinguish between human-written and AI-generated text. SOTA approaches were used for comparison. While traditional ML methods struggle to capture the subtle differences between human and AI generated text, medium-sized language models maintain their expected performance even on unseen data. Zero-shot approaches, such as Ghostbuster and Binoculars, perform well on their original data, but fail to maintain their TPR on new data. However, their FPR remained relatively small and consistent throughout the datasets. We also analyzed the impact of varying nucleus sampling parameters. The predicted confidence of Binoculars appears to be impacted by these parameters, which was not the case for the DeBERTa model. With enough data, Transformer models seem to become invariant to such perturbations in generated text.

Future work will focus on maintaining a strict FPR on unseen data, while at the same time achieving a reasonable TPR. For a detection system to be deployed in a production scenario, its performance needs to be closely monitored to ensure trustworthy predictions. While in the case of plagiarism, one could validate suspicions by finding the original work that was copied, generated content is impossible to confirm without external mechanisms such as watermarking (Kirchenbauer et al., 2023). The current study focused only on evaluating different approaches on shorter text sequences. However, the final solution should scale to longer documents, either by ensambling predictions on a sliding context window or using model architectures that inherently scale to longer sequences.

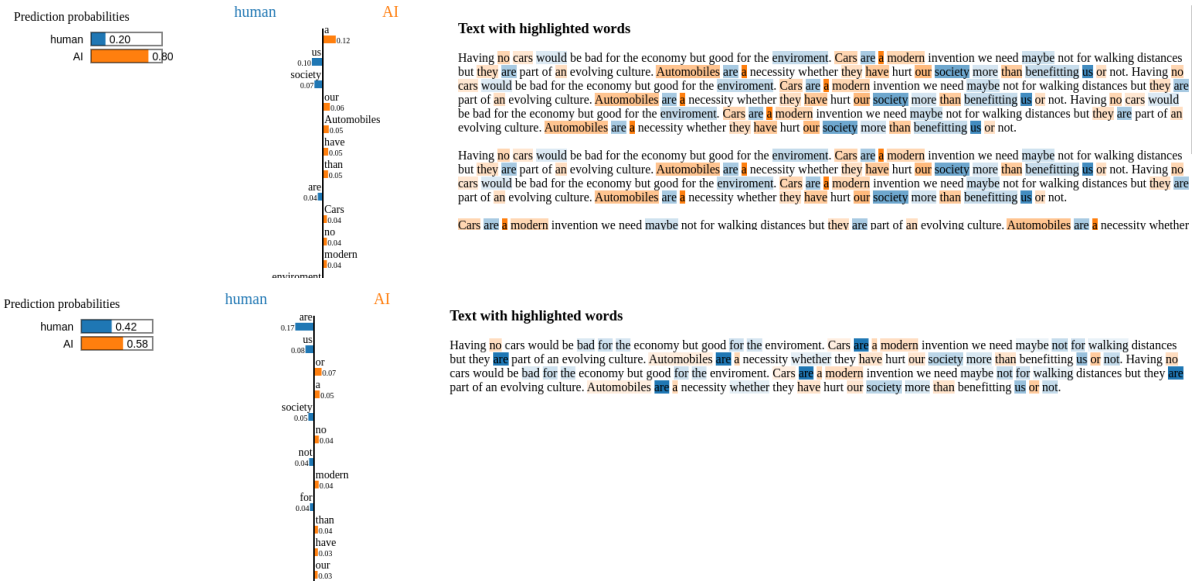

Figure 7: LIME of the top performing RNN model on a noisy observation before and after correction.

7.1 Limitations

We conclude with some limitations of our study. Firstly, the evaluated models have a maximum context window of 1024 tokens which limits the real world applicability of this research. Moreover, our dataset consists of sequences ranging from 500 to 2000 tokens. Combined with potential biases in the generation process, this could lead to a significant data distribution shift when applied to real-world examples.

Secondly, the large number of architecture variants examined in this paper made it difficult to perform comprehensive hyperparameter tuning. This issue is alleviated to some degree by the fact that our best performing models, mainly the Transformer-based approaches, have a reduced sensibility to hyperparameter choice.

Lastly, in the interest of reproducibility, all the experiments were run in an environment with 32GB RAM and 16GB GPU VRAM, matching the specifications of free hardware offered by major cloud providers. However, without hardware constraints, we could train larger models and improve our dataset with content generated by more powerful LLMs.

7.2 Ethical Considerations

The dataset is based on the public Persuade 2.0 corpus which is anonymized and doesn't include any gender or racial information, nor is to our knowledge biased against any minority. Any third party dataset that was used is cited in the references.

REFERENCES

Bahdanau, D., Cho, K., and Bengio, Y. (2016). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Biswas, R., Bamba, U., and Broad, N. (2024). Llm - detect ai datamix (version 1).

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., and Askell, A. (2020). Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Crossley, S., Baffour, P., Tian, Y., Franklin, A., Benner, M., and Boser, U. (2023). A large-scale corpus for assessing written argumentation: Persuade 2.0. *Available at SSRN 4795747*.

Dettmers, T., Pagnoni, A., and Holtzman, A. & Zettlemoyer, L. (2024). Qlora: Efficient finetuning of quantized llms. *Advances in Neural Information Processing Systems*, 36.

Dey, R. and Salem, F. (2017). Gate-variants of gated recurrent unit (gru) neural networks. In *In 2017 IEEE 60th international midwest symposium on circuits and systems (MWSCAS), IEEE*, pages 1597–1600.

Gunasekar, S., Zhang, Y., Aneja, J., Mendes, C. C. T., Del Giorno, A., Gopi, S., and Li, Y. (2023). Textbooks are all you need. *arXiv preprint arXiv:2306.11644*.

Hans, A., Schwarzschild, A., Cherepanova, V., Kazemi, H., Saha, A., and Goldblum, M. & Goldstein, T. (2024). Spotting llms with binoculars: Zero-shot detection of machine-generated text. *arXiv preprint arXiv:2401.12070*.

He, P., Gao, J., and Chen, W. (2021a). Debertav3: Improving deberta using electra-style pre-training with gradient-disentangled embedding sharing. *arXiv preprint arXiv:2111.09543*.

He, P., Liu, X., Gao, J., and Chen, W. (2021b). Deberta: Decoding-enhanced bert with disentangled attention. *arXiv preprint arXiv:2006.03654*.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Holtzman, A., Buys, J., Du, L., Forbes, M., and Choi, Y. (2020). The curious case of neural text degeneration. *arXiv preprint arXiv:1904.09751*.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., and Chen, W. (2021). Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Jiang, A. Q., Sablayrolles, A., Mensch, A., Bamford, C., Chaplot, D. S., Casas, D. D. L., and Sayed, W. E. (2023). Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Joachims, T. (1997). A probabilistic analysis of the rocchio algorithm with tfidf for text categorization. *In ICML*, 97:143–151.

Kenton, J. D. M. W. C. & Toutanova, L. K. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

King, J., Baffour, P., Crossley, S., Holbrook, R., and Demkin, M. (2023). Llm - detect ai generated text. *Kaggle*.

Kirchenbauer, J., Geiping, J., Wen, Y., Katz, J., Miers, I., and Goldstein, T. (2023). A watermark for large language models. In *In International Conference on Machine Learning PMLR*, pages 17061–17084.

Lundberg, S. M. and Lee, S. I. (2017). A unified approach to interpreting model predictions. In *NIPS'17: Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 4768–4777.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Mitchell, E., Lee, Y., Khazatsky, A., and Manning, C. D. & Finn, C. (2023). Detectgpt: Zero-shot machine-generated text detection using probability curvature. In *In International Conference on Machine Learning PMLR*, pages 24950–24962.

Muñoz-Ortiz, A., Gómez-Rodríguez, C., and Vilares, D. (2024). Contrasting linguistic patterns in human and llm-generated news text. *Artificial Intelligence Review*, 57(10):265.

Nguyen, T., Hatua, A., and Sung, A. (2023). How to detect ai-generated texts? In *In IEEE 14th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON), IEEE*.

Ribeiro, M. T., Singh, S., and Guestrin., C. (2016). " why should i trust you?" explaining the predictions of any classifier. In *In Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144.

Sadasivan, V. S., Kumar, A., Balasubramanian, S., and Wang, W. & Feizi, S. (2023). Can ai-generated text be reliably detected? *arXiv preprint arXiv:2303.11156*.

Sennrich, R., Haddow, B., and Birch, A. (2015). Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*.

Soboleva, D., Al-Khateeb, F., Myers, R., Steeves, J., Hestness, J., and Dey, N. (2023). Slimpajama: A 627b token cleaned and deduplicated version of redpajama.

Tian, E. and Cui, A. (2023). Gptzero: Towards detection of ai-generated text using zero-shot and supervised methods.

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M. A., Lacroix, T., and Lample, G. (2023). Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A., and Kaiser, L. & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

Verma, V., Fleisig, E., and Tomlin, N. & Klein, D. (2024). Ghostbuster: Detecting text ghostwritten by large language models. *arXiv preprint arXiv:2305.15047*.

Yang, X., Pan, L., Zhao, X., Chen, H., Petzold, L., and Wang, W. Y. & Cheng, W. (2023). A survey on detection of llms-generated content. *arXiv preprint arXiv:2310.15654*.

APPENDIX

Table 5: Metrics for all models on validation and holdout test set using thresholds computed at 5×10^{-3} validation FPR.

	classifier	val_roc	val_tpr	val_f1	test_roc	test_tpr	test_fpr	test_f1	val_f1_rank	test_f1_rank
transformer	deberta3_both	0.99586	0.92029	0.95781	0.99473	0.83669	0.00030	0.91104	1	1
transformer	deberta3_finetune_only	0.99024	0.83364	0.90873	0.98946	0.74060	0.00060	0.85088	2	2
rnn	gru_word2vec_1_256	0.97135	0.68682	0.81363	0.87155	0.69490	0.11974	0.80215	3	3
rnn	gru_fasttext_2_256	0.96621	0.66883	0.80085	0.89919	0.58960	0.02598	0.73803	4	9
rnn	gru_fasttext_1_256	0.97231	0.66098	0.79519	0.89566	0.61010	0.04270	0.75156	5	8
rnn	lstm_word2vec_1_256	0.97306	0.64253	0.78167	0.87598	0.57729	0.04210	0.72590	6	10
rnn	lstm_glove_1_256	0.96370	0.62993	0.77226	0.87609	0.65871	0.08062	0.78227	7	5
cpu	logreg_bpe_1_3	0.95762	0.62847	0.77116	0.89521	0.49821	0.00776	0.66399	8	15
rnn	gru_word2vec_1_128	0.96241	0.62692	0.76999	0.84911	0.63736	0.11227	0.76207	9	7
rnn	gru_glove_2_256	0.95125	0.61760	0.76292	0.87197	0.56450	0.04598	0.71502	10	11
rnn	lstm_fasttext_2_256	0.95351	0.60747	0.75512	0.87224	0.48562	0.02001	0.65100	11	19
cpu	sgd_bpe_1_3	0.95478	0.60500	0.75321	0.89599	0.45816	0.00538	0.62768	12	22
rnn	gru_glove_1_256	0.95938	0.59989	0.74923	0.85557	0.64517	0.09555	0.77023	13	6
rnn	lstm_glove_2_256	0.95974	0.59560	0.74587	0.85514	0.55030	0.05882	0.70155	14	13
cpu	xgb_bpe_1_1_nlp	0.94313	0.59551	0.74580	0.84806	0.40767	0.00866	0.57810	15	24
rnn	lstm_glove_1_128	0.95072	0.57469	0.72923	0.83838	0.56628	0.06360	0.71396	16	12
rnn	lstm_fasttext_1_128	0.94275	0.57442	0.72901	0.90279	0.49276	0.00478	0.65954	17	17
rnn	gru_fasttext_1_128	0.95356	0.57423	0.72886	0.89374	0.48185	0.01254	0.64861	18	20
rnn	gru_glove_1_128	0.95360	0.56364	0.72026	0.85779	0.67102	0.12601	0.78450	19	4
rnn	lstm_fasttext_1_256	0.95511	0.55771	0.71539	0.88964	0.49539	0.01344	0.66069	20	16
rnn	lstm_word2vec_1_128	0.95616	0.54931	0.70843	0.83992	0.53902	0.06718	0.69097	21	14
cpu	xgb_bpe_1_1	0.93442	0.53086	0.69289	0.81997	0.35737	0.02717	0.52327	22	26
rnn	gru_word2vec_2_256	0.95256	0.52995	0.69211	0.86797	0.48928	0.02150	0.65410	23	18
cpu	sgd_bpe_1_1	0.93962	0.51598	0.68007	0.85698	0.44961	0.01612	0.61815	24	23
rnn	lstm_word2vec_2_256	0.95201	0.51342	0.67784	0.83767	0.46644	0.02508	0.63274	25	21
cpu	lightgbm_bpe_1_1_nlp	0.91537	0.49160	0.65851	0.82959	0.31478	0.00119	0.47870	26	29
cpu	sgd_bpe_3_5	0.93193	0.46832	0.63726	0.86291	0.34101	0.00269	0.50827	27	28
cpu	lightgbm_bpe_1_1	0.90742	0.45773	0.62737	0.81559	0.26034	0.00299	0.41282	28	32
cpu	logreg_bpe_1_1	0.92862	0.42348	0.59439	0.83127	0.35182	0.00896	0.51943	29	27
baseline	binoculars	0.67542	0.35327	0.52153	0.77242	0.40307	0.00119	0.57440	30	25
cpu	multinb_bpe_3_5	0.90596	0.33747	0.50409	0.75496	0.27304	0.00000	0.42895	31	31
cpu	multinb_bpe_1_1	0.85244	0.25265	0.40291	0.75866	0.29504	0.00418	0.45518	32	30
cpu	logreg_bpe_1_1_nlp	0.66155	0.05807	0.10962	0.69065	0.04128	0.00090	0.07926	33	33
cpu	sgd_bpe_1_1_nlp	0.49959	0.00000	0.00000	0.50051	0.00000	0.00000	0.00000	34	34
transformer	deberta3_pretrain_only	0.87106	0.00000	0.00000	0.94077	0.00000	0.00000	0.00000	34	34