DUTh at SemEval 2024 Task 8: Comparing classic Machine Learning Algorithms and LLM based methods for Multigenerator, Multidomain and Multilingual Machine-Generated Text Detection

Theodora Kyriakou Ioannis Maslaris Avi Arampatzis
Database & Information Retrieval research unit,
Department of Electrical & Computer Engineering,
Democritus University of Thrace, Greece
{theokyri6,imaslari,avi}@ee.duth.gr

Abstract

Text-generative models evolve rapidly nowadays. Although, they are very useful tools for a lot of people, they have also raised concerns for different reasons. This paper presents our work for SemEval2024 Task-8 on 2 out of the 3 subtasks. This shared task aims at finding automatic models for making AI vs. human written text classification easier. Our team, after trying different preprocessing, several Machine Learning algorithms, and some LLMs, ended up with mBERT, XLM-RoBERTa, and BERT for the tasks we submitted. We present both positive and negative methods, so that future researchers are informed about what works and what doesn't.

1 Introduction

LLMs are becoming more and more part of our everyday lives due to their easy accessibility and their remarkably fluent responses in different fields like news, healthcare and education. This extensive usage can lead to unintended consequences. Specifically, LLMs could replace humans, provide sometimes false, incomplete or even misleading information, risk the critical thinking of students and progressively of the whole society. So, it is of high importance to find a way to identify if a text was written by a human or by a machine. Since all these complex models are trained on large datasets and have achieved generating texts that are so human like, it is difficult for a person to identify who generated a text. Here comes the importance of the automatic models, capable of differentiating between human written texts and machine generated texts, by exploiting patterns invisible to a human.

In this paper we describe the DUTh participation in *SemEval 2024 Task 8: Multigenerator, Multidomain, and Multilingual Black-Box Machine-Generated Text Detection*. The task features three directions: *Binary Human-Written vs. Machine-Generated Text Classification, Multi-Way Machine-Generated Text Classification and Human-Machine Mixed Text Detection*. The first two refer to the scenario where a system must classify input texts which are fully written by a human or a machine. One detail regarding the second scenario is that we are also provided with the specific language model which generated the input text. In the third scenario we are presented with a text which is half human and half machine written and we have to determine the boundary, where the change occurs. In all subtasks the text data are coming from different sources and different generators, and we are not allowed to use any external data except for the ones given from the organizers.

The sub-tasks can be briefly described as follows: SubtaskA monolingual: given only English texts, we need to determine whether a text is human-written or machine-generated.

SubtaskA multilingual: given texts from 8 different languages (English, Arabic, Chinese, Indonesian, Urdu, German, Bulgarian, Russian), we need to determine whether a text is human-written or machine-generated.

SubtaskB: given only English texts, we need to determine whether a text is human-written or machine-generated and which is the specific generator.

SubtaskC: given only English mixed texts, where the first part is human-written and the second part is machine-generated, we need to determine the boundary.

Our team participates by submitting on SubtaskA (both monolingual and multilingual) and SubtaskB. During the competition we examine several methods, especially on SubtaskA monolingual, like different preprocessing techniques on the text data, several Machine Learning Algorithms, some ensembling methods and LLMs. We ended up submitting LLMs to all subtasks.

The models we choose are mBERT for subtaskA monolingual, XLM-RoBERTa for subtaskA multilingual and BERT for subtaskB, as they achieve better performance on average. All these pre-trained models have been proven to be powerful for different NLP tasks.

Our proposed system for every subtask is a classifier based on a fine-tuned Large Language Model. During the training process, our model is provided with a text as input and a label regarding whether this input is human or machine generated. Additionally, this paper provides a comparative study of different LLMs fine tuned in this task. In this context, we also provide results regarding the use of classic machine learning algorithms trained to tackle this task.

2 Background

2.1 Dataset

All datasets given from organizers are on jsonl format. For both subtaskA and subtaskB the English human-written texts are coming from the following five sources, “wikihow“, “wikipedia“, “peerread“, “reddit“ and “arxiv“. The generators for machine-generated texts are “chatGPT“, “cohere“, “davinci“, “bloomz“ and “dolly“.

For multilingual data, the sources and generators are the same. The languages it consists of are English, Arabic, Bulgarian, Chinese, Indonesian, Urdu, German and Russian.

More information about the datasets and tasks can be found from the organizers.(Wang et al., 2024a) (Wang et al., 2024b) (Wang et al., 2024c)

2.2 Evaluation Metrics

The evaluation metrics for this task are accuracy, micro-f1 and macro-f1. Though, the organizers ranked both on validation and test set the participants basically based on the accuracy scores.

3 System Overview

Transformers have achieved state-of-the-art(Wolf et al., 2020) performances on several natural language processing tasks such as text classification. This is why all final models submitted are LLMs. Here we present the submitted model on each subtask.

We have all the hyperparameters for tuning the models submitted in the appendix section 6.

3.1 Tokenization applied

In all three models we use the tokenizer they already have. We define a max length of 512 tokens, which means that each encoder will take the first 512 tokens of the text as an input. We use truncation and padding, so that if a text has more than 512 tokens it gets cut off on the 512th token and if it has less than 512 tokens it gets padded until it reaches 512. We want all texts to have the same length. All the encoders of the transformers can give a representation of their input tokens, in a high dimensional space (512D here), based on the meaning of each token. For example, the same word can have different representation if its meaning changes. There is no other preprocessing made on the texts except for the tokenization applied by each model.

3.2 SubtaskA models

For the monolingual part of this subtask, we select multilingualBERT (bert-base-multilingual-cased)(Devlin et al., 2018). After comparing lots of classifiers, the two most performing are BERT and multilingualBERT. Previous research finds that there is no apparent benefit in training dedicated monolingual models for single language tasks, and actually by using a multilingual model instead may yield slightly improved performance de Vargas Feijó and Moreira (2007). Our case is no different. We can see that multilingualBERT slightly outperforms BERT on Table 8. MultilingualBERT is a pretrained model on 104 languages and has 179M parameters. For the multilingual part, we select XLM-RoBERTa (xlm-roberta-base)(Conneau et al., 2019) as it demonstrates the best performance between all models we examine. We do not apply any preprocessing on the input text, so XLM-R gets used as a cased model. XLM-R is pre-trained on 2.5TB of filtered CommonCrawl data containing 100 languages and has 279M parameters.

3.3 SubtaskB model

Between ML algorithms and LLMs we select BERT (bert-base-cased) for this task as we have seen that transformers, most of the times, have better results. It is pre-trained on a large corpus of English data and it has 109M parameters.

4 Experimental Setup

4.1 Preprocessing

We apply some preprocessing only on English data, when we use Machine Learning classifiers. There is no preprocessing on English data when we use

LLMs (either for embeddings or classification) or in the multilingual subtask.

The preprocessing is done with the following order. At first, we lowercase the text data and then we can either replace unicodes or remove them, but since we see that the latter has better results, we remove them. After that, we replace the emails with the word <email> and the URLs with the word <url>. We, also, remove the digits and all punctuations. Considering both tokenization and lemmatization, we see that tokenization performs better. Moreover, we achieve better results by the removal of stop words. Finally, we create some new features from the text data. These features are the number of times some phrases, words or combinations of punctuations (“cannot“, “do not“, “.,“ and more) appear in each text, the number of characters on the original texts and the number of words after tokenization.

4.2 Embeddings

For SubtaskA monolingual, we try to get the embeddings using Word2Vec(Mikolov et al., 2013), Tf-Idf(Ramos et al., 2003) and BERT encoder. On Word2Vec we try the following vector sizes 20, 40, 60, 80, 100, 150, 200 and 300. On Tf-Idf we try the following X more frequent words 500 and 1000 with Ngrams of (3,5). Finally, on BERT encoder we get embeddings from the last of the 12 layers.

After comparing all the above, we see that the best results are coming from Word2Vec with vector size 20.

On SubtaskA multilingual, we get embeddings with the multilingualBERT encoder from its last layer.

We use Word2Vec with vector size 20 as on SubtaskA monolingual, to get embeddings on SubtaskB text data too, in order to see the performance of some ML algorithms on this task.

4.3 Machine Learning Algorithms VS LLMs

The metric we use is accuracy. We have seen that micro-f1 and macro-f1 values fluctuate according to the accuracy value. We, also, standardize the embeddings before we feed them into the ML algorithms. The accuracy values are calculated based on the preprocessing mentioned above except for the part of stop words. When stop words are removed it is specified on the table. Also, when we do not standardize the input data we mention it on the table.

In this section, all models are trained and evaluated using the training set and validation set the organizers give us. All values are calculated on the validation set. ML algorithms without * or additional information presented on the tables, have the default parameters of scikit-learn and XGBClassifier libraries (versions 1.3.0 and 1.7.3 respectively).

4.3.1 SubtaskA monolingual

We start to get embeddings using Word2Vec different vector sizes and compare them based on Logistic Regression. The results are on the Table 1.

We can see that Word2Vec embeddings with vector size 20 is the best based on LR. Now, we take the best embeddings, with vector size 20, and try different ML algorithms to see what results we can take on the validation set. The results are on the Table 2.

We can see that the best result here is default AdaBoost (Freund and Schapire, 1997) with optimized RandomForest (Breiman, 2001) and removed stop words. We have also tried optimizations to other algorithms and some voting ensembling methods with some of the best results from above, but everything was worse than the best one.

Now, we try different methods on getting embeddings to see if anything can beat Word2Vec with vector size 20 based on Logistic Regression and Random Forest. The results are on Table 3.

Now, we compare the best result from the ML algorithms with miniLM and BERT. Both miniLM and BERT are trained for 5 epochs on the training set and evaluated on the validation set. We evaluate them on every 100 batches, and we take the mean of all evaluations. The results are on the Table 4.

4.3.2 SubtaskA multilingual

We take embeddings using the last layer of multilingualBERT encoder and try some ML algorithms. The results are on the Table 5.

We can see from the algorithms compared that the best here is XG Boost(Chen and Guestrin, 2016) with no standardization applied.

Now, we compare the best ML algorithm with multilingualBERT and XLMRoBERTa as classifiers. The results are on Table 6 and again for the LLMs’ values, because we evaluate them on every epoch from the 5, we take the mean of them.

We can see here that XLM-RoBERTa is slightly better from default XgBoost. So, this is the best model for this task.

	vector size						
Algorithm	**20**	**40**	**60**	**80**	**150**	**200**	**300**
Logistic Regression	**0.559**	0.553	0.5164	0.5046	0.4848	0.4854	0.4902

Table 1: Logistic Regression accuracy per vector size (Word2Vec).

Model	**Acccuracy**
Logistic Regression	0.559
XG Boost	0.7362
Decision Tree (Breiman, 2017)	0.6946
SGDC *	0.5532
Random Forest	0.7538
Random Forest optimized	0.7552
AdaBoost optimized	0.756
AdaBoost **	**0.7584**
Bagging optimized (Breiman, 1996)	0.7538

Table 2: Machine Learning algorithms accuracy on monolingual validation set. *modified huber loss. ** Adaboost with optimized random Forest with removed stop words.

4.3.3 SubtaskB

On this subtask we take embeddings using the last layer of BERT and try some ML algorithms. The results of how ML algorithms perform on this multiclass task are presented on Table 7.

5 Modification on datasets

5.1 Rationale

We make some comparisons between datasets, and we decide to create new training and validation sets for subtaskA and subtaskB. We notice that subtaskA multilingual training set contains all the English data of the rest datasets and some extra, which means that it has the most English data. So, we decide to create a new monolingual dataset with all English data. We, also, notice that in the multilingual training set there are only the English, Chinese, Indonesian, Urdu and Bulgarian data and on multilingual validation set there are the three other languages. Thus, we create a new dataset with all multilingual data containing all languages. Now, from these two new datasets we create the new training and validation sets of subtaskA and subtaskB.

5.1.1 SubtaskA monolingual

Using the dataset with all the English data, we keep 131589 for training and 5000 for validation, where the 2500 texts are human-written, and the 2500 texts are machine-generated. With these new datasets we train and evaluate BERT. Because we want to train multilingualBERT, also, to see its performance on English data, we take the multilingual training set given from organizers and exclude the same as before 5000 English data for validation. By this way, this training set has the same remaining 131589 English data for training and the same 5000 English data for evaluation, with the difference now that this training set has 4 more languages (Chinese, Indonesian, Urdu, Bulgarian) and not only English data.

Both models are trained for 5 epochs and they are evaluated on all 5 epochs. The results are on Table 8 and both values are the mean of all their 5 evaluations.

5.1.2 SubtaskA multilingual

Using the dataset with all multilingual data, we create a new training set and a new validation set that contain texts from all languages. Specifically, the training set contains 133589 English, 10000 Chinese, 5000 Indonesian, 5000 Urdu, 10000 Bulgarian, 900 Arabic, 1800 Russian and 900 German data. The validation set contains 3000 English, 1934 Chinese, 995 Indonesian, 899 Urdu, 2000 Bulgarian, 100 Arabic, 200 Russian and 100 German with 50 percent human-written texts and 50 percent machine-generated texts. There is no specific technique behind the chosen percentages of each language.

We train the XLM-RoBERTa on this new training set for 5 epochs and make evaluations on each one of the 5 epochs. The result is 95.5 percent and it is the mean of all 5 evaluations.

5.1.3 SubtaskB

Using the dataset with all English data, the training and validation set for SubtaskB given from organizers, we concatenate the training and validation sets to compare them with the dataset of all English data. We can see that there are 62562 English data that are not used on this subtask. In these 62562 data there are different percentages of each class from the 6 (human, chatGPT, cohere, davinci, bloomz and dolly). We keep the same sample of

Algorithm	**W2V with 20 VS**	Tf-Idf 500 *	Tf-Idf 1000 *	Bert last layer
Logistic Regression	0.559	0.6288	0.6348	0.6618
Random Forest	**0.7552**	0.6226	0.7132	0.654

Table 3: Logistic Regression and Random Forest accuracy per different embedding methods. *With Ngram

Model	**Accuracy**
AdaBoost *	0.7584
MiniLM	0.7695
BERT	**0.783**

Table 4: Comparison of best machine learning algorithm with evaluated LLMs on monolingual task. *Adaboost with optimized Random Forest with removed stop words.

1844 texts from each class. The value is 1844, because in the 62562 texts, one of the classes has only this amount.

So, we create a new training set where we just add on the training set given from organizers these 1844 samples from each class. The validation set is the same. On this new training set we train BERT for 5 epochs, as we have seen on the other tasks that LLMs more often than not beat Machine Learning Algorithms. We evaluate BERT on every 10000 batches using the validation set organizers give us. The result is 96.81 percent and it is the mean of all evaluations.

5.2 Final Models

Finally, we combine on every subtask the new training set and validation set we created. We train the best models for 5 epochs on these datasets.

We can say that LLMs, most of the times, perform better on these tasks than Machine Learning algorithms. Nevertheless, there are some ML algorithms, combined with the right preprocessing and embeddings' method, that can give good results close to those LLMs give. As we can see, default AdaBoost with optimized Random Forest and with removed stop words achieves a quite good performance on subtaskA monolingual. The preprocessing made and the features we created on subtaskA monolingual seem to improve the performance of algorithms. Also, Default XgBoost with no standardization achieves a close enough to the best model performance on subtaskA multilingual. We believe that LLMs perform better because they are pre-trained models on a large corpus of English or multilingual data. Thus, they can better understand the meaning of a word and a whole text, and maybe this makes it easier for them to differentiate human-written from machine-generated texts.

Finally, the models submitted on the competition scored 73.243 on subtaskA monolingual, 76.45 on subtaskA multilingual and 56.683 on subtaskB. This means about 20 percent below for subtaskA and 30 percent below for subtaskB from the scores we had on the new validation sets we created from the dataset organizers give us. We think that this drop is due to the fact that the texts on the test sets are coming from different domain, generator and language. Basically, on subtaskA monolingual all texts are coming from a new domain “outfox“ and there is also a new generator “gpt-4“. On subtaskA multilingual, again the texts are coming from the same new domain and the languages it consists of are German, Arabic and Italian. The two first were also on the training but in a small amount and the 3rd one was not in the training set. On subtaskB, the only difference is the domain, which is the same as on every subtask, “outfox“.

6 Conclusion

Based on the results we have on our new validation set and the results on the test set, we assume that this drop occurs since our model cannot generalize well. We believe that if the test sets had texts coming from the same domains and generators as the training texts, and had the same languages, then our models would have achieved better results.

Future work could focus on either training larger language models or trying to improve generalization of ours, possibly with some preprocessing like data augmentation. We look forward to further research on these tasks, hoping for better results.

References

Leo Breiman. 1996. Bagging predictors. *Machine learning*, 24:123–140.

Leo Breiman. 2001. Random forests. *Machine learning*, 45:5–32.

Leo Breiman. 2017. *Classification and regression trees*. Routledge.

Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Unsupervised cross-lingual representation learning at scale. *arXiv preprint arXiv:1911.02116*.

Diego de Vargas Feijó and Viviane Pereira Moreira. 2007. Mono vs multilingual transformer-based models: a comparison across several language tasks. *CoRR, abs*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Yoav Freund and Robert E Schapire. 1997. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139.

Jerome H Friedman. 2001. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 26.

Juan Ramos et al. 2003. Using tf-idf to determine word relevance in document queries. In *Proceedings of the first instructional conference on machine learning*, volume 242, pages 29–48. Citeseer.

Yuxia Wang, Jonibek Mansurov, Petar Ivanov, Jinyan Su, Artem Shelmanov, Akim Tsvigun, Chenxi Whitehouse, Osama Mohammed Afzal, Tarek Mahmoud, Giovanni Puccetti, Thomas Arnold, Alham Fikri Aji, Nizar Habash, Iryna Gurevych, and Preslav Nakov. 2024a. Semeval-2024 task 8: Multigenerator, multidomain, and multilingual black-box machine-generated text detection. In *Proceedings of the 18th International Workshop on Semantic Evaluation*, SemEval 2024, Mexico, Mexico.

Yuxia Wang, Jonibek Mansurov, Petar Ivanov, Jinyan Su, Artem Shelmanov, Akim Tsvigun, Chenxi Whitehouse, Osama Mohammed Afzal, Tarek Mahmoud, Toru Sasaki, Thomas Arnold, Alham Fikri Aji, Nizar Habash, Iryna Gurevych, and Preslav Nakov. 2024b. M4: Multi-generator, multi-domain, and multi-lingual black-box machine-generated text detection. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics*, Malta.

Yuxia Wang, Jonibek Mansurov, Petar Ivanov, Akim Tsvigun, Jinyan Su, Artem Shelmanov, Osama Mohammed Afzal, Tarek Mahmoud, Giovanni Puccetti, Thomas Arnold, Alham Fikri Aji, Nizar Habash, Iryna Gurevych, and Preslav Nakov. 2024c. MG-Bench: Evaluation benchmark for black-box machine-generated text detection.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*, pages 38–45.

A Appendix

Model	Accuracy
Logistic Regression*	0.606
XGBoost*	**0.663**
Random Forest*	0.598
AdaBoost**	0.500
XGBoost	0.654

Table 5: Comparison of Machine Learning algorithms accuracy on multilingual task. * Without standardization. ** AdaBoost with XGBoost and without standardization.

Model	Accuracy
XGBoost*	0.663
Multilingual BERT	0.577
XLM-RoBERTa	**0.668**

Table 6: Comparison of best machine learning with evaluated LLMs on the multilingual. * Without standardization

Model	Accuracy
Logistic Regression	0.4636
XGBoost	**0.4720**
Gradient Boosting	0.4523

Table 7: Comparison of Logistic Regression, XGBoost and Gradient Boosting (Friedman, 2001) on sub-task B.

Model	Accuracy
BERT	0.9430
MultilingualBERT	**0.9540**

Table 8: Comparison of BERT and MultilingualBERT on the monolingual new validation set

Hyperparameter	**Range/Value**
Epochs	5
Batch size	16
Weight decay	0.02
Learning rate	2e-5

Table 9: Hyperparameter values for the multilingual BERT and XLM-RoBERTa.

Hyperparameter	**Range/Value**
Epochs	5
Batch size	16
Weight decay	0.03
Learning rate	2e-5

Table 10: Hyperparameter values for BERT.