BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY

Excel@FIT 2025
http://excel.fit.vutbr.cz

Detecting AI-generated Text

Bc. Matej Koreň*

Abstract

This project addresses the detection of texts generated by artificial intelligence (AI) with the use of various machine learning models and large language models (LLM). It explores text analysis techniques, machine learning methods, and modern transformer-based models, such as BERT or GPT and their ability to generate text. The main goal is to create a reliable model/s for text classification, which are then used within a web application and integrated into the test module of the company Lakmoos AI. The developed solution is useful in plagiarism detection or text ownership attribution and helps the company to create language models that produce answers as most humanly as possible

*xkoren10@vut.cz, *Faculty of Information Technology, Brno University of Technology*

1. Introduction

The rise of AI-generated text presents new challenges in ensuring the authenticity of online content. Machine learning offers a powerful tool for detecting AI-generated text by analyzing linguistic patterns and inconsistencies that distinguish it from human writing. By identifying and flagging AI-generated content, we can maintain trust in digital communication and prevent the spread of misinformation. Using machine learning for this purpose helps protect the integrity of information and ensures a more transparent and reliable online space. Since the creation of the generative models, much research has been conducted in order to provide a robust and objective tool for differentiating between human-written and AI-generated texts. These tools are relatively good but are often monetized and do not provide enough explanation of their classification decisions. As we will mention, there are more than a dozen online AI detectors, of which some are performing better than others. However, quantity does not often mean quality – these tools are usually designed to perform the detection on a specific type of text (such as short news articles and social media statuses or, by contrast, on large documents and online papers), mainly depending on the size and quality of the training data.

2. Proposed solution

The general idea of creating a binary classifier is very simple - the first step is to decide whether to use a machine learning algorithm with a statistical approach or do it the modern way - use LLMs to detect LLMs. Either way, the steps needed to produce such tools are similar to any classification task using supervised learning – dataset selection and/or creation of a new dataset, model selection by experiments and more thorough testing of selected models.

To visualise all the steps mentioned above, we can create a pipeline, which shows the difference between feature-based and feature-less approaches:

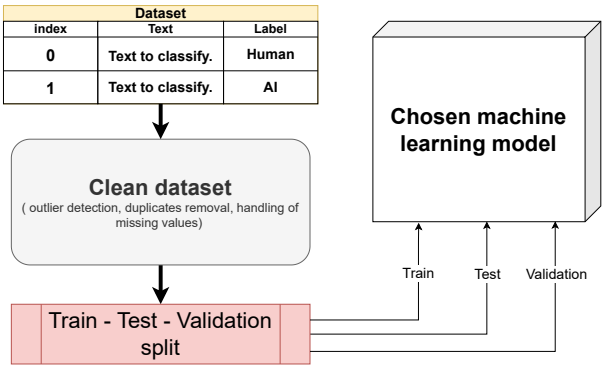

Figure 1. Visualization of the pipeline to create an AI-generated text detector using machine learning. The *train* and *valid* subsets are used during the cross-validation training period, and later, the *test* subset is used to rank the model's ability to classify unseen data. To find the best model, the F1 score and the accuracy metric can be utilised to determine the model's suitability for future application.

The final product of this thesis should be an easy tool for detecting the authorship of a given text. Since this work is being done in collaboration with Lakmoos AI, it also needs to be included in their codebase as well. Therefore, the final product will consist of a standalone application that can be deployed online and a module supporting the tests and the main functionality of the company tool.

3. Experiments

One of the first step done before experimenting with machine learning models was extracting more features, which will be used by the feature-based models we will pick. The selected numerical and literal attributes were:

1. Embeddings – words as vectors,
2. n-grams – digrams and trigrams,
3. Perplexity [1, 2] – uncertainty of a language model when predicting the next word.
4. Burstiness [2] – word appearance,
5. Entropy – randomness or unpredictability in a text,
6. Stop words count – frequently used words that usually carry little unique meaning,
7. Sentence count,
8. Syllable count,
9. Unique word count and Unique word ratio,
10. Average word length, Character count.

These features were integrated into a training dataset, which was used to train multiple machine learning models with the help of a *python* library **Pycaret** – this library serves as a wrapper above various models, simplifying the training and testing process as well as offering an easy comparison of accuracies and precisions of them. After defining the classification experiment, the testing yielded the best results for **Extreme gradient boosting model (XGBC)**, which achieved 70% accuracy and F1-score. This model was therefore selected as the go-to feature-based approach for our task, and after further hyperparameter optimisation, we were able to increase the accuracy to 80%.

label	precision	recall	f1-score	support
AI	0.78	0.79	0.79	1530
Human	0.81	0.79	0.80	1670

Table 1. Classification report from the XGBC model trained with optimal parameters. We can observe a slightly higher accuracy within the human class.

Since we have also converted the dataset into embeddings, a neural network was another suitable classification method we tested. However, even with regularisation and dropout layers, the network suffered from overtraining and was unable to generalise accurately and was only able to achieve 55% accuracy – the reason this could happen might lie within the nature of the training texts, where words converted into embeddings do not carry the information about relationships and meaning within the sentence.

This problem was solved by using a transformer-based model, such as **distilbert-base-uncased**[1], which was trained on the same dataset.

The training process went like this:

Epoch	Validation Loss	Accuracy	Precision	Recall	F1
1	0.593683	0.691542	0.756344	0.691542	0.632225
2	0.498089	0.774876	0.812813	0.774876	0.752618
3	0.588391	0.772388	0.803072	0.772388	0.751733
4	0.440752	0.832090	0.844615	0.832090	0.824615

Table 2. Progress of the distilbert model during four epochs of training.

This shows great potential in using this type of model for text classification.

4. Web application

After experimenting with different machine learning models, the best-performing ones (**XGBC** and **BERT**) were selected and a web application was created. It consists of a backend part, written in *python* with the **FastAPI** library, and a frontend part, written in *java script* with the **ReactJS** library. The backend also uses a variety of other libraries, such as *xgboost*, *transformers*, *torch*, *shap*, *textstat*, *pandas* and *sklearn.feature_extraction.text*.

To make the classification output more user-friendly, the class probabilities are displayed in an interactive pie chart and the user can request a decision explanation. For this, a **SHAP** explainer library was used. It is a game-theoretic approach to explain the output of any machine learning model. It connects optimal credit allocation with local explanations using the classic Shapley values from game theory and their related extensions [2]. **SHAP** offers per-token influence visualisation for the feature-less model and feature importance bar plot for the feature-based model. To get the best of both worlds, a combined classification can be utilised as well, passing the input through both models and computing a weighted average of class probability outputs.

The application is currently being deployed at `ai-detector.lakmoos.com` .

[1] https://huggingface.co/distilbert

[2] Cited from https://shap.readthedocs.io/en/latest/

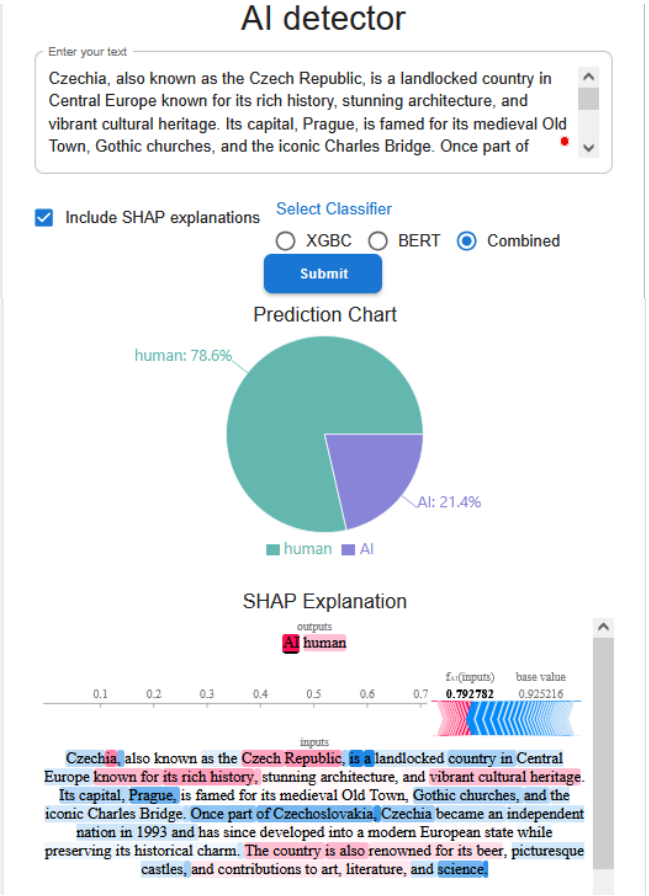

Figure 2. A screenshot taken from the app running locally showcasing the simplicity of the interface and its layout.

5. Conclusions

The project and the master's thesis provide a closer look at how generative models produce text and how they can be successfully and accurately detected with machine learning. It considers various studies done on this topic, gathering insights and inspiration for such classifier and tries to utilize this knowledge to create a detector, that can be used as a standalone application, as well as serve as a additional human-likeness scoring method for responses of a large language model used in the Lakmoos AI company. For this purpose, one feature-based and one feature-less model was chosen, comparing the two approaches as well as combining them to achieve more informed decisions. The result is in a form of a web application, that is publically available and offers interactive user interface with additional classification explanations and which can be accessed with an API call, further expanding its usability not only within the company, but also to the broader world.

References

[1] F. Jelinek, R. L. Mercer, L. R. Bahl, and J. K. Baker. Perplexity—a measure of the difficulty of speech recognition tasks. *The Journal of the Acoustical Society of America*, 62(S1):S63–S63, 08 2005.

[2] Sherice Jacob. Perplexity and burstiness in writing. *Originality.AI Blog*, (1), 2024. Accessed: 2025-03-19.