FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

MASTER THESIS

Adnan Al Ali

Understanding the Detection of Generated Text and its Biases

Department of Theoretical Computer Science and Mathematical Logic

Supervisor of the master thesis: Mgr. Jindřich Helcl, Ph.D.
Study programme: Computer Science – Artificial Intelligence

Prague 2025

I declare that I carried out this master thesis on my own, and only with the cited sources, literature and other professional sources. I understand that my work relates to the rights and obligations under the Act No. 121/2000 Sb., the Copyright Act, as amended, in particular the fact that the Charles University has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 subsection 1 of the Copyright Act.

In .... date ....

Author's signature

I would like to thank my supervisor, Mgr. Jindřich Helcl, Ph.D., for his help and guidance throughout my work on this thesis, especially when the experiments seemed to lead nowhere. I also appreciate my family for their material and emotional support. Finally, I would like to express my gratitude to my friends, who patiently accepted my absence during the busy days.

Title: Understanding the Detection of Generated Text and its Biases

Author: Adnan Al Ali

Department: Department of Theoretical Computer Science and Mathematical Logic

Supervisor: Mgr. Jindřich Helcl, Ph.D., Institute of Formal and Applied Linguistics

Abstract: Following the release of ChatGPT – a chatbot based on GPT-3.5 – and its subsequent growth in popularity, concerns have emerged about the possible misuse of the service, particularly for plagiarism in academia. Given the natural-sounding text generation, the distinction between human-written and generated text is challenging for humans. In contrast, machine-learning methods proved to be accurate to some extent. However, prior work states that these methods rely on perplexity and tend to be biased against non-native speakers, whose texts often have lower perplexities. In this thesis, we revisit these statements after two years, specifically in the Czech language setting. We construct detectors from three separate families and find no systematic bias against non-native speakers. We further show that the perplexity of the texts from Czech non-native speakers is not lower than that of the native ones. Finally, we demonstrate that our detectors function effectively without relying on perplexity.

Keywords: generated text detection, plagiarism detection, GPT, BERT, bias

Název práce: Porozumění detekci generovaného textu a její zaujatosti

Autor: Adnan Al Ali

Katedra: Katedra teoretické informatiky a matematické logiky

Vedoucí diplomové práce: Mgr. Jindřich Helcl, Ph.D., Ústav formální a aplikované lingvistiky

Abstrakt: Po uvedení ChatGPT – chatbota založeného na modelu GPT-3.5 – a jeho následném nárůstu popularity se objevily obavy z možného zneužití této služby, zejména v oblasti plagiátorství v akademickém prostředí. Díky přirozeně znějící generaci textu je pro člověka obtížné rozlišit mezi textem napsaným člověkem a generovaným textem. Naproti tomu metody strojového učení se ukázaly být do určité míry přesné. Dřívější výzkumy však uvádějí, že tyto metody se opírají o míru perplexity a mají tendenci být zaujaté vůči nerodilým mluvčím, jejichž texty mívají nižší perplexitu. V této práci se k těmto tvrzením vracíme po dvou letech, konkrétně v kontextu českého jazyka. Vytváříme detektory ze tří různých skupin a nezjišťujeme žádnou systematickou zaujatost vůči nerodilým mluvčím. Dále ukazujeme, že perplexita textů českých nerodilých mluvčích není nižší než u rodilých mluvčích. Nakonec demonstrujeme, že naše detektory fungují efektivně i bez spoléhání se na perplexitu.

Klíčová slova: detekce generovaného textu, detekce plagiátorství, GPT, BERT, zaujatost

Contents

Introduction

Following the release of LLM-based assistants – most notably ChatGPT, which was based on GPT-3 (Brown et al., 2020) and later upgraded to GPT-4 (OpenAI; Achiam et al., 2024) – and their subsequent growth in popularity, concerns have emerged about the possible misuse of the service, particularly for plagiarism. This concert was largely raised in academic contexts (Susnjak; McIntosh, 2024).

Given the natural-sounding text generation, the distinction between human-written and generated text is challenging for humans (Milička et al., 2025). In contrast, machine-learning methods proved to be accurate to some extent (Junchao Wu et al., 2025). However, according to Liang et al. (2023), some of these methods are perplexity-based (perplexity in this context refers to a measure of 'surprise' of an LLM when attempting to predict the next token of a given text; see Section 4.5.1 for definitions) and tend to be biased against non-native English speakers, whose texts often have lower perplexities. As a result, the texts from non-native speakers are often falsely flagged as AI-generated.

In this thesis, we follow up on the work of Liang et al. (2023), in the Czech-speaking context, and aim to answer three fundamental questions:

1. Is there a bias against non-native speakers in Czech generated text detectors?
2. Is the perplexity of the texts from Czech non-native speakers lower than the texts of their native peers?
3. Is it possible to create generated text detectors without – explicitly or implicitly – relying on perplexity?

To answer the first question, we create a set of generated text detectors and analysers from the commonly used categories: (1) classical machine learning model using a bag-of-words text representation, (2) fine-tuned pre-trained RoBERTa-like (Y. Liu et al., 2019) model, and (3) entropy-based analyser. In addition, we add a commercial detector and an unconventional non-pre-trained Transformer (Vaswani et al., 2017) encoder-only model for examination. We evaluate these models on multiple domains to assess their overall quality and their performance on the texts of native vs. non-native speakers.

For the second question, we use the aforementioned entropy-based analyser. Note that entropy and perplexity have a straightforward exponential relationship. We measure the entropy distributions in all the inspected domains and compare the texts from the non-native speakers with those of their peers, as well as other domains. Furthermore, we visualise the contributions to entropy for each token of a given sample and gain some insight into the factors that affect the entropy of a text.

Finally, to answer the third question, we inspect the correlations within one class (human-written or generated) between the entropy-based analyser and the rest of the analysers, to assess whether they implicitly work with some internal representation of the entropy. We further examine how these correlations change across domains.

Our research finds that the answer to the three proposed questions is considerably different from the work of Liang et al. (2023). We highlight the fact that significant advances in the generated text detection task have been made since 2023. Furthermore, the language setting proved to be an important factor when considering the bias against non-native speakers.

The thesis has the following structure: in Chapter 1, we present the theoretical background for the used concepts. Chapter 2 provides the related literature overview. Chapter 3 discusses the dataset used and the pre-processing pipelines. Chapter 4 contains all of the conducted experiments. In it, we dedicate Sections 4.1, 4.2, 4.3, and 4.4 to creating and examining various types of generated text detectors. Section 4.5 inspects the entropy of the datasets. Finally, Section 4.6 provides a comparison of the presented methods using correlation analysis. For reproducibility, we provide the scripts used in the experiments in a GitLab repository[1].

[1] `https://gitlab.mff.cuni.cz/alalia/detectors-thesis` (accessible for members of the Faculty of Mathematics and Physics only).

1 Theoretical Background

This chapter explains fundamental concepts that the thesis heavily relies on. We aim to provide sufficient context for readers unfamiliar with Machine Learning (ML) or Natural Language Processing (NLP), and only general technical knowledge is expected. Experienced readers may skip most of the sections.

For the purposes of our thesis, we use the following definition of Machine Learning by T. M. Mitchell and T. M. Mitchell (1997):

> *A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.*

Given that the concepts of *experience*, *task*, and *performance* are loosely defined, the definition is applicable to various scenarios, leading to various ML algorithms.

For our purposes, we expect that the *experience* is a *dataset* consisting of (1) a set of *examples* $X = \{\mathbf{x}^{(i)} \mid \mathbf{x}^{(i)} \in X\}$, which are collections of *features* $x_1^{(i)}, x_2^{(i)}, ...$; and (2) a set of labels $Y = \{y^{(i)} \mid y^{(i)} \in Y\}$, corresponding to each example. Note that in our *notation*, we make a distinction between scalars, printed in cursive (a), and vectors, printed in bold ($\mathbf{a}$).

1.1 Probability Theory

The essential concept for probability theory is the *sample space* Ω, which can be viewed as the set of all possible outcomes of an experiment. An *event* E is any subset of the *sample space*. An *event space* $\mathcal{F}$ is a subset of all possible events: $\mathcal{F} \subseteq \mathcal{P}(\Omega)$. A *probability function* is a function $P : \mathcal{F} \to [0, 1]$ satisfying:

1. $P(\Omega) = 1$,
2. for each $E_1, \ldots, E_m \subseteq \mathcal{F}$ such that $E_i \cap E_j = \varnothing$ for $i \neq j$:
$$P\left(\bigcup_i E_i\right) = \sum_i P(E_i)$$
.

The *conditional probability* of an event E given another event F, denoted by $P(E \mid F)$ is defined as:

$$P(E \mid F) = \frac{P(E \cap F)}{P(F)}$$

A useful law regarding the conditional probability is *Bayes' law*, which shows the relationship between $P(E \mid F)$ and $P(F \mid E)$:

$$P(E \mid F) = \frac{P(F \mid E)P(E)}{P(F)}$$

Events E and F are *independent*, if $P(E \cap F) = P(E)P(F)$ and *conditionally independent* given and event G, if $P(E \cap F \mid G) = P(E \mid G)P(F \mid G)$ (Harchol-Balter, 2023, Chapter 2).

A *random variable* is a function $X : \Omega \to \mathbb{R}$. A random variable is *discrete* if the set of all possible values $\mathrm{Im}(X)$ is countable. For random variable X, the Probability Mass Function (PMF) $p_X : \mathrm{Im}(X) \to [0,1]$ is defined as $p_X(a) = P(X = a) = P(\{\omega \in \Omega \mid X(\omega) = a\})$. The PMF defines a *probability distribution.*

The *Bernoulli distribution* is a common distribution, in which the $\mathrm{Im}(X) = \{0, 1\}$. It is parametrised by a single number $p \in [0, 1]$. Then $p_X(1) = p$ and $p_X(0) = 1 - p$ (Harchol-Balter, 2023, Chapter 3). A generalisation of the Bernoulli distribution to $\mathrm{Im}(X) = \{1, \ldots, k\}$ is the *categorical distribution*, parametrised by $p_1, \ldots, p_{k-1} \in [0, 1]$, such that $\sum_{i=1}^{k-1} p_i \leq 1$. Then (Goodfellow et al., 2016, Chapter 3):

$$p_X(i) = \begin{cases} p_i, & \text{for } i < k \\ 1 - \sum_{j=1}^{k-1} p_j \stackrel{\text{def}}{=} p_k, & \text{for } i = k \end{cases}$$

The *binomial distribution* arises from the sum[1] of n independent trials of a random variable that follows the Bernoulli distribution with parameter p. The PMF is defined as:

$$p_X(i; n, p) = \binom{n}{i} p^i (1-p)^{n-i}$$

To generalise the binomial distribution to a higher dimension, the definition of a random variable must be generalised first: from $X : \Omega \to \mathbb{R}$ to $X : \Omega \to \mathbb{R}^k$. The *multinomial distribution* arises from counting the outcomes of n independent trials of a random variable that follows the categorical distribution with parameters $p_1, \ldots, p_k$. This results in a vector of counts $\mathbf{c} \in \{0, \ldots, n\}^k$ such that $\sum_i c_i = n$. The PMF is defined as (Bishop, 2006, Chapter 2):

$$p_X(\mathbf{c}; n, p_1, \ldots, p_k) = \frac{n!}{c_1!\, c_2!\, \ldots\, c_k!} \prod_{i=1}^{k} p_i^{c_i}$$

1.2 Classification Task

The classification task is the most relevant task for our thesis. In the task, the computer (specifically, a *model* running on the computer) is required to predict one of k labels (or *classes*) for an example x. This typically involves learning the parameters θ of a prediction function $f_\theta : \mathbb{R}^n \to \{1, ..., k\}$ (Goodfellow et al., 2016, Chapter 5), although the example x is not necessarily a vector of a fixed length nor real numbers, as will be our case.

A special case of the classification task is the *binary* classification task, where $k = 2$. In this case, the labels are often called *positive* (1) and *negative* (0). We view the generated text detection as an instance of such a task: the two classes are *generated* (i.e. positive) and *natural*/human-written (i.e. negative).

1.2.1 Evaluation Metrics

For tasks such as classification, the typical evaluation metric (performance measure) is *accuracy* – i.e. the proportion of correctly classified examples (Goodfellow et al., 2016, Chapter 5). Let $\hat{y}^{(i)}$ be the label predicted by the model for $\mathbf{x}^{(i)}$ and $y^{(i)}$ the true label of $\mathbf{x}^{(i)}$. Then the accuracy on the set X is defined as:

[1] Or, equivalently, the *count* of positive draws (where $X = 1$).

$$\text{Acc} = \frac{\left|\{i \mid 0 < i < |X| \wedge y^{(i)} = \hat{y}^{(i)}\}\right|}{|X|}$$

While accuracy provides an intuitive evaluation, achieving high accuracy is trivial when the classes are unbalanced. Furthermore, it may also be useful to distinguish the model's performance on samples belonging to the individual classes rather than the whole dataset. For these reasons, we introduce alternative metrics: a prediction is *false negative* if the predicted label is 0, while the true label is 1 and *false positive* if the predicted label is 1, while the true label is 0 (Russell; Norvig, 2016, Chapter 19). Subsequently, we define the False Negative Rate (FNR) and False Positive Rate (FPR) as:

$$\text{FNR} = \frac{\left|\{i \mid 0 < i < |X| \wedge y^{(i)} = 0 \wedge \hat{y}^{(i)} = 1\}\right|}{|\{i \mid 0 < i < |X| \wedge y^{(i)} = 1\}|}$$

$$\text{FPR} = \frac{\left|\{i \mid 0 < i < |X| \wedge y^{(i)} = 1 \wedge \hat{y}^{(i)} = 0\}\right|}{|\{i \mid 0 < i < |X| \wedge y^{(i)} = 0\}|}$$

In other words, FPR is the proportion of false positives relative to all negative samples and FNR is the proportion of false negatives relative to all positive samples. In the context of generated text detection, a false positive is a sample that is human-written but classified as generated. A false negative is a sample that is generated but incorrectly classified as human-written.

1.3 Optimisation

The process of optimisation involves finding the minimum[2] of a *loss function* $f(\mathbf{x})$ (Goodfellow et al., 2016, Chapter 4). A typical loss function for the classification task is the *Negative Log-Likelihood* (NLL) defined as follows (Goodfellow et al., 2016, Chapter 5):

$$-\mathbb{E}_{\mathbf{x},y\sim\hat{p}_{\text{data}}} \log p_{\text{model}}(y \mid \mathbf{x})$$

1.3.1 Optimisers

The loss function is commonly optimised numerically, using *gradient descent*. In order to find $\mathbf{x}^* = \arg\min_{\mathbf{x}} f(\mathbf{x})$, the method iteratively computes the gradient $\nabla_{\mathbf{x}} f(\mathbf{x})$ and updates $\mathbf{x}$ to a new value:

$$\mathbf{x}' = \mathbf{x} - \epsilon \nabla_{\mathbf{x}} f(\mathbf{x})$$

where $\epsilon > 0$ is the *Learning Rate* (LR).

A drawback of the gradient descent method is that it is not guaranteed to reach the *global minimum* and can converge to any local minimum. The optimisation objective of finding the global optimum is therefore often relaxed to finding a 'satisfactory' local minimum (Goodfellow et al., 2016, Chapter 4).

[2] In other contexts, optimisation can also refer to finding the maximum. In the general case, the optimised function is called the *objective* function.

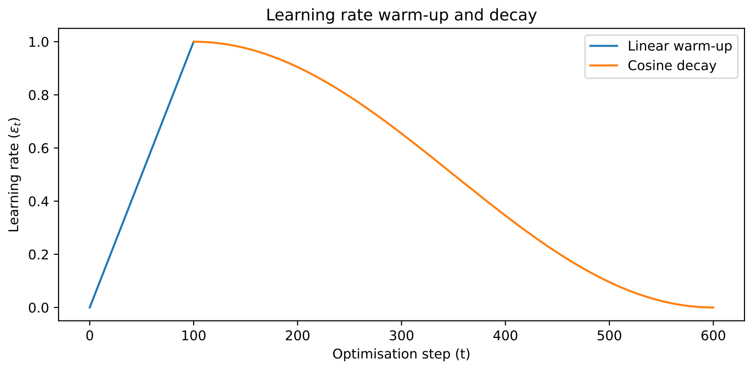


Figure 1.1 Example learning rate scheduling plot for $T_1 = 100$ linear warm-up steps, $T_2 = 500$ cosine decay steps, and $\epsilon_T = 1$.

To allow working with a large training dataset, the gradient is often estimated on a small subset drawn uniformly from the training data called a *minibatch* or just *batch*. This required the loss function to be decomposable into per-example loss computations, which holds for the NLL. Such extension of the gradient descent method is called the *Stochastic Gradient Descent* (SGD; Goodfellow et al., 2016, Chapter 5). A more advanced version of the SGD algorithm is *AdamW* (Loshchilov; Hutter, 2019), which adapts a learning rate for each parameter and supports a straightforward choice of the *weight decay* (which is a method of controlling overfitting).

1.3.2 Learning Rate Scheduling

In numerous scenarios, a variable Learning Rate is used, where a *scheduler* determines the LR for each optimisation step (Zhang et al., 2023, Sec. 12.11). The LR is typically scheduled in two phases: *warm-up* and *decay*[3].

In the warm-up phase, the goal is to ensure stability at the beginning of the training, when the performance is poor and the loss gradients are high. For the first T_1 steps, the learning rate will gradually increase from 0 to ϵ_T. A straightforward approach is to use a *linear* warm-up, which defines the LR at time t as:

$$\epsilon_t = \frac{t}{T_1}\epsilon_T$$

At the end of the training, the aim is to have a small LR to 'refine' the weights, while avoiding too drastic changes to the LR and keeping it reasonably high during most of the decay phase. Cosine decay (Loshchilov; Hutter, 2017) can be used to achieve this, using the following formula below to decay from T_2 to 0:

$$\epsilon_t = \frac{1}{2}\epsilon_T\left(1 + \cos\frac{\pi t}{T_2}\right)$$

Figure 1.1 shows an example relationship of ϵ_t on t when using the combination of linear warm-up and cosine decay.

[3]Not to be confused with weight decay.

1.3.3 Underfitting, Overfitting, Regulatisation

The ability of a model to perform well on previously unseen data is called *generalisation*. In ML, the objective is to minimise the *generalisation error*, alongside the *training error*. To measure the generalisation error, the original dataset is divided into a *train set* and a *validation set*, and only the former is used during training.

The situation where the training error is high is called *underfitting*. The opposite phenomenon, in which the generalisation error is considerably larger than the training error, is called *overfitting*. Altering the model's *capacity* – i.e. its ability to fit a wide variety of functions – is one of the ways to control underfitting and overfitting (Goodfellow et al., 2016, Chapter 5).

Aside from directly changing the number of parameters in the models, overfitting can be prevented by choosing certain approaches during training, including (Goodfellow et al., 2016, Chapter 7):

- L^2 parameter regularisation, commonly known as *weight decay*. This involves adding the term $\alpha\frac{1}{2}\|w\|_2^2$ to the loss function, where w is the vector of all weights of the model and α is a constant (hyperparameter). This, in turn, limits the model's capacity.

- *Early stopping*, which is the practice of ending the training process once the performance no longer improves on the validation set.

- *Dataset augmentation* – creating *new* samples by alternating the original ones slightly. An example would be to shift, rotate or scale images in an image classification task.

- *Dropout* (Srivastava et al., 2014), during which, with the probability of p the output of each hidden unit is replaced with 0 (masked). On inference, no masking is applied, and all the weights where dropout was applied are scaled down by a factor of p to account for the larger number of inputs. This is done to prevent the units from co-adapting 'too much'.

- *Label smoothing*, which softens the hard category targets by taking $\beta < 1$ from the true target and redistributing it towards the other labels uniformly. For binary distribution, this would mean that, after smoothing, a positive sample would have the target probability $1 - \beta$ for the positive label and β for the negative label.

1.4 Naïve Bayes Model

The Naïve Bayes (NB) model is a model for classification which exploits the Bayes' law (see Section 1.1) and introduces the naïve assumption of conditional independence of the features given the class, which significantly simplifies the model. Suppose a vector of features $\mathbf{x} = (x_1, \ldots, x_k)^\top$ and l classes $C_1, \ldots, C_l$. The goal is to predict the probability of a class given the features: $p(C_i \mid x_1, \ldots, x_k)$. Using Bayes' law, this can be modelled as:

$$p(C_i \mid x_1, \ldots, x_k) = \frac{p(x_1, \ldots, x_k \mid C_i)p(C_i)}{p(x_1, \ldots, x_k)}$$

Furthermore, given a fixed sample $\mathbf{x}$ when computing the probabilities of the classes, the denominator can be viewed as a normalisation constant (Russell; Norvig, 2016, Chapter 13):

$$p(C_i \mid x_1, \ldots, x_k) = \alpha p(x_1, \ldots, x_k \mid C_i) p(C_i)$$

where

$$\alpha = \frac{1}{\sum_{j=1}^{l} p(x_1, \ldots, x_k \mid C_j) p(C_j)}$$

The assumption of conditional independence of the features given the class states that $p(x_1, \ldots, x_k \mid C_i) = \prod_{j=1}^{k} p(x_j \mid C_i)$, which yields the prediction formula:

$$p(C_i \mid x_1, \ldots, x_k) = \alpha p(C_i) \prod_{j=1}^{k} p(x_j \mid C_i)$$

Unlike the methods mentioned later in this chapter, the NB model is typically trained analytically, using Maximum a Posteriori (MAP) hypothesis learning, rather than using numerical optimisation methods. This involves setting the parameters such that the product of a posteriori probabilities over the entire training dataset is maximised. The a posteriori probability of a single sample $\mathbf{x}$ belonging to a class C_y is defined as $p(y) \prod_{j=1}^{k} p(x_j \mid C_y)$.

To model the MAP probability, a distribution of the features must be assumed (Russell; Norvig, 2016, Chapter 20). For text classification, the vector of counts of each word in the vocabulary can be used in combination with the *multinomial NB*. In that case, the probability $p(x_i \mid C_y)$ is modelled with a single parameter $\theta_{y,i}$, estimated as

$$\hat{\theta}_{y,i} = \frac{N_{y,i} + \alpha}{N_y + \alpha k}$$

where $N_{y,i}$ is the total number of times the feature i occurs in all samples belonging to the class C_y

$$N_{y,i} = \sum_{\{\mathbf{x} \mid \mathbf{class}(\mathbf{x}) = \mathbf{C_y}\}} x_i$$

$N_y = \sum_i N_{y,i}$ is the count of all features in class C_y, and α is a smoothing coefficient to prevent zero probabilities (scikit-learn developers, [n.d.]).

1.4.1 Naïve Bayes Model with TF-IDF Features

Term Frequency-Inverse Document Frequency (TF-IDF) is a form of representing a text using a fixed-length vector $\mathbf{x} \in \mathbb{R}^n$. It is an instance of the *bag-of-words* model, in which the order of the words is disregarded, while the word counts are essential. Aside from text classification, the TF-IDF model is used in information retrieval. The *term frequency* $\mathrm{tf}_{t,d}$ of a term t in a document d is the number of occurrences of t in the document.

To assign a lower weight to terms that are common across a large number of documents (and, therefore, likely have a smaller significance in the classification of the documents), the *inverse document frequency* is introduced: let df_t be the document frequency of the term t, i.e. the number of documents in the training set that contain the term t. Then, the inverse document frequency is defined as

$$\mathrm{idf}_t = \log \frac{N}{\mathrm{df}_t}$$

where N is the number of documents in the training set. The TF-IDF score is then computed as the product of the two: $\text{tf-idf}_{t,d} = \text{tf}_{t,d} \cdot \text{idf}_t$ (Manning et al., 2008, Chapter 6).

The inverse document frequency vector $(\text{idf}_1, \ldots, \text{idf}_{|\text{vocab\_size}|})^\top$, containing the idf scores for each term in the vocabulary, is computed during training. On inference, when presented with a document d, the term frequency vector $(\text{tf}_{1,\text{d}}, \ldots, \text{tf}_{|\text{vocab\_size}|,\text{d}})^\top$, containing the document's term frequencies for each term in the vocabulary, is computed. The final TF-IDF representation is computed as the element-wise product of the two vectors.

While the features obtained using TF-IDF are not discrete, as required in a multinomial distribution (see Section 1.1), it has been shown empirically (Kibriya et al., 2005) that including the idf component improves the performance of the multinomial NB classifier compared to using the discrete term counts (tf) alone.

1.5 Feed-Forward Networks

Feed-forward networks are essential parts of ML architectures. They are composed of one or more feed-forward layers. The final layer is called the *output layer*, and the inner layers are called *hidden*. If a hidden layer is present, the network is called a *Multilayer Perceptron* (MLP). The output of each layer is computed using the previous one as:

$$\mathbf{x}^{(i+1)} = a\left(W^{(i+1)^\top}\mathbf{x}^{(i)} + \mathbf{b}^{(i+1)}\right)$$

where $a : \mathbb{R}^k \to \mathbb{R}^k$ is an *activation function*.

1.5.1 Activation Functions

The standard activation function in the hidden layers of modern MLPs is the *Rectified Linear Unit* (ReLU), which is defined pointwise as $\text{ReLU}(x) = \max\{0, x\}$. The activation function in the output layer depends on the specific task. Typically, one of three functions is used: (1) the identity function for regression, (2) the *sigmoid* (σ) function for binary classification, and (3) the *softmax* function for general classification (Goodfellow et al., 2016, Chapter 6).

The pointwise sigmoid function $\sigma : \mathbb{R} \to (0, 1)$ is defined as

$$\sigma(x) = \frac{1}{1 + \exp(-x)}$$

The $\text{softmax} : \mathbb{R}^k \to (0, 1)^k$ function is defined as:

$$\text{softmax}(\mathbf{x})_i = \frac{\exp(x_i)}{\sum_j \exp(x_j)}$$

Note that the outputs of the softmax function sum up to 1, representing a valid categorical distribution.

1.6 Tokenisation

Neural networks, such as the Transformer architecture (described in Section 1.8), internally work with continuous vectors of numbers rather than with text. Converting text inputs to numbers is a fundamental task of NLP. A typical approach is to learn a vector – embedding (see Section 1.7) – for each term of the vocabulary. Implicitly, this requires the vocabulary to be a finite set.

However, many NLP tasks, such as Machine Translation, are *open-vocabulary*. Sennrich et al. (2016) introduced a way of dealing with out-of-vocabulary tokens using *subword units*. The intuition is that such units can hold partial meaning, and the meaning of an unobserved word can be, to some extent, reconstructed from its known subwords.

The tokeniser training procedure is based on the *Byte Pair Encoding* (BPE) algorithm (Gage, 1994). For the purposes of training a tokeniser, the algorithm can run on two levels: character level or UTF-8 byte level (Radford; Jeffrey Wu et al., 2019a). The tokenisation is trained on one text of arbitrary length (e.g. the concatenation of all training data).

The algorithm starts with an initial vocabulary of symbols – the individual characters or bytes, depending on the level. For a given number of iterations, it counts each unique consecutive pair of symbols in the text and finds the most common one. The identified pair of symbols (A, B) is then merged into a single symbol AB. To prevent the merging of two words, a special word boundary symbol is introduced, which is not eligible for merging.

The final number of characters is the initial number of symbols (256 for the byte level) plus the number of iterations. This allows the size of the vocabulary to be set to any number larger than the initial vocabulary. Note that when working on the byte level, the problem of unknown symbols is eliminated, as any character can be encoded using 1–4 UTF-8 bytes. This is known as the *byte fallback*.

SentencePiece (Kudo; Richardson, 2018) provides an extension to the BPE algorithm by dropping the requirement for the text to be pre-tokenised into words, which makes the system language-independent. Furthermore, it makes the tokenisation *lossless*, i.e. the tokens can be detokenised (reconstructed) into the original text without losing any information. This is achieved by retaining the information about whitespace in the tokenised form.

1.7 Embeddings

To convert the discrete tokens to their initial continuous representation, a learnt matrix $E \in \mathbb{R}^{\text{vocab\_size} \times d_{\text{model}}}$ is used. In the matrix, the i-th row corresponds to the (non-contextual) embedding of the i-th term in the vocabulary. When converting the final continuous representation to a probability distribution over the terms for the next token, the transposition of the same matrix $E^\top$ is used, followed by the softmax activation.

1.7.1 Positional Embeddings

In the paragraph above, we described how a token – regardless of its position in the text – is converted into a continuous representation. In this section, we describe how the information about the *position/order* of a token, regardless of its underlying subword, is converted into a continuous representation of a fixed-length vector. The motivation behind this stems from the fact that by default, the Transformer architecture (described in Section 1.8) is unaware of the order of the tokens.

As a result, information about the order – *positional embeddings* – must be added to the tokens manually. The positional embedding for each token is a vector of dimension d_{model}, and it is added to the token embedding before the first attention block. The original article (Vaswani et al., 2017) proposed the use of the sine and cosine functions, which we do not discuss, as alternative solutions have been more widely adopted in contemporary models.

Trained embeddings. In case the *context window* (the maximum number of tokens the model works with at a time) is limited, a position embedding vector $\mathbf{p}_i$ can be trained for each position i, similarly to the token embeddings. The encoder-only models described in Section 1.9 use this approach, although some modern encoder-only models (Warner et al., 2024) use the approach described below.

Rotary Positional Embeddings (RoPE) RoPE (Jianlin Su et al., 2024) is a technique of encoding the absolute positions of the tokens while explicitly incorporating the *relative position* of two tokens in the self-attention mechanism (described in Section 1.8.1). Suppose two intermediate token representations $\mathbf{x}_m$ and $\mathbf{x}_n$ at positions m and n respectively. The relative position refers to the difference $m - n$. The motivation behind RoPE is to find functions f_q, f_k transforming the intermediate representations and their positions into query and key vectors $\mathbf{q}$, $\mathbf{k}$ such that $\mathbf{q}^\top \mathbf{k} = \langle \mathbf{q}, \mathbf{k} \rangle$ is a function g of the original vectors $\mathbf{x}_m$, $\mathbf{x}_n$ and their relative position $m - n$[4]:

$$\langle f_q(\mathbf{x}_m, m), f_k(\mathbf{x}_n, n) \rangle = g(\mathbf{x}_m, \mathbf{x}_n, m - n)$$

The authors proved that such behaviour can be achieved in the following way: The d-dimensional space (for an even d) is split into $d/2$ consecutive subspaces of dimension 2, and the components within the subspaces are rotated by $m\theta_i$ where θ is a constant for each subspace. The whole transformation for queries (and keys analogously) is calculated as follows:

$$f_q(\mathbf{x}_m, m) = R^d_{\Theta,m} W_q \mathbf{x}_m$$

where

$$R^d_{\Theta,m} = \begin{pmatrix} \cos(m\theta_1) & -\sin(m\theta_1) & 0 & 0 & \cdots & 0 & 0 \\ \sin(m\theta_1) & \cos(m\theta_1) & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cos(m\theta_2) & -\sin(m\theta_2) & \cdots & 0 & 0 \\ 0 & 0 & \sin(m\theta_2) & \cos(m\theta_2) & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & \cos(m\theta_{\frac{d}{2}}) & -\sin(m\theta_{\frac{d}{2}}) \\ 0 & 0 & 0 & 0 & \cdots & \sin(m\theta_{\frac{d}{2}}) & \cos(m\theta_{\frac{d}{2}}) \end{pmatrix}$$

[4] Note that this reduces the information about absolute positions to information about relative positions, from which the absolute position cannot be recovered.

and

$$\Theta = \left\{\theta_i = 10\,000^{\frac{-2(i-1)}{d}} \mid i \in \{1, 2, \ldots, d/2\}\right\}$$

Consequently, it holds that:

$$\mathbf{q}_m^\top \mathbf{k} = (R^d_{\Theta,m} W_q \mathbf{x}_m)^\top (R^d_{\Theta,n} W_k \mathbf{x}_n) = \mathbf{x}^\top W_q^\top R^d_{\Theta,m-n} W_k \mathbf{x}_n$$

which satisfies the condition formulated above.

Note that the RoPE mechanism is applied before each computation of the attention, only to the query and key vectors, as opposed to the one-time application to the token embeddings used in the methods mentioned above. Furthermore, similarly to the originally proposed sine and cosine embeddings, the RoPE has the property of *long-term decay*, i.e. the dot product decays with position increase, following the intuition that two distant tokens are less likely to be related.

1.8 The Transformer

The Transformer (Vaswani et al., 2017) is an architecture widely used for NLP tasks (Devlin et al., 2019; Straka et al., 2021; Brown et al., 2020; Grattafiori et al., 2024). Originally proposed for sequence transduction (such as Machine Translation; MT), it consisted of two components shown in Figure A.1: the encoder (left) and the decoder (right). As a modification of the model, two architectures have emerged: the encoder-only models (described in Section 1.9) and the decoder-only models (described in Section 1.10).

The main building block of the Transformer architecture is the *Transformer block*. The blocks consist of a *multi-head self-attention*[5] module and an element-wise feed-forward network with one hidden layer. Furthermore, the self-attention in the decoder is modified by masking, such that each element can only attend to previous elements. Finally, both of the modules are augmented by *residual connections* and normalised using layer normalisation as $\mathrm{LayerNorm}(x + \mathrm{module}(x))$.

1.8.1 Scaled Dot-Product Attention

The attention mechanism is used to add information from other relevant tokens to a given token. Specifically, the attention used in the Transformer is defined as a function of queries ($Q \in \mathbb{R}^{n \times d_q}$), keys ($K \in \mathbb{R}^{n \times d_q}$), and values ($V \in \mathbb{R}^{n \times d_v}$). Where n is the sequence length and d_q, d_v are fixed hyperparameters of the model. The keys and values are used to compute the compatibility weights between the elements, and the output for each element is computed as the weighted sum of the values, specifically:

$$\mathrm{Attention}(Q, K, V) = \mathrm{softmax}\left(\frac{QK^\top}{\sqrt{d_q}}\right) V$$

The Q, K, and V matrices are obtained as projections of the intermediate continuous representations using the respective projection matrices $W^Q \in \mathbb{R}^{d_{\mathrm{model}} \times d_q}$,

[5] In the case of the full encoder-decoder architecture, the *cross-attention* module connecting the encoder to the decoder is also present, as shown in Figure A.1.

$W^K \in \mathbb{R}^{d_{\text{model}} \times d_q}$, and $W^V \in \mathbb{R}^{d_{\text{model}} \times d_v}$. The multi-head attention mechanism involves doing h such operations in parallel (each with its separate projection matrices), concatenating the results and projecting them back to the dimension of the model d_{model}.

In the case of self-attention, all three projections are computed from the sequence[6]. Therefore, the self-attention acts as a contextualization measure within the encoder or the decoder.

1.9 Encoder-Only Transformer Models

Encoder-only Transformer models only contain the encoder component of the Transformer architecture. While they are typically not used to generate text, they are often used in various NLP tasks, such as named entity recognition, sentiment analysis, question answering, etc. The typical workflow involves pre-training the model on a large corpus of unlabeled text and later fine-tuning it for specific downstream tasks, with a smaller set of labelled data.

1.9.1 BERT

Bidirectional Encoder Representations from Transformers (BERT; Devlin et al., 2019) utilises the encoder-only architecture and proposes a way for pre-training a *bidirectional* language model. In this context, 'bidirectional' means that each token has full information about all the other tokens. This makes more straightforward training methods (such as the next token prediction) unusable.

Pre-training setup. During pre-training, two text segments are always presented to the model. Two special tokens are added to the token sequence: (1) the `[CLS]` token is added as the first token and serves as a way to aggregate the information about all the tokens, which can be used for text classification and (2) the `[SEP]` token is added between the two segments as a separator. Aside from the trained positional embeddings, one of two trained embedding vectors $\mathbf{a}$ or $\mathbf{b}$ is added to the tokens' embeddings depending on whether they belong to the first or the second segment. This is done to further distinguish the two. The training has two objectives described below.

Masked Language Model objective. In the MLM objective, a random subset of tokens is selected for a perturbation, and the model is trained to predict the original underlying tokens. More specifically, 15% of tokens are selected for prediction. Out of those, 80% are replaced by the special `[MASK]` token, while 10% are replaced with a random token and 10% are kept unchanged. The model then predicts the true token on each of the 15% of selected positions with NLL loss. This 'partial' masking is done to make the pre-training more similar to downstream tasks, in which the `[MASK]` token is typically not present.

[6]In contrast, cross-attention computes the queries as projections from the decoder, while the keys and the values are computed from the decoder.

Next Sentence Prediction objective. The NSP[7] objective is used to train the model to understand the relationship between two segments of text. In the training data, the two segments can either be contiguous from the same text or two different texts. The model is then trained for binary classification: `IsNext` or `NotNext` using a classification head connected to the final continuous representation of the `[CLS]` token.

1.9.2 RoBERTa

RoBERTa (robustly optimised BERT pre-training approach; Y. Liu et al., 2019) proposed multiple changes to the BERT pre-training process and demonstrated that the performance can be significantly improved. The changes included:

- *Larger pre-training dataset:* the dataset used has 160 GB, as opposed to the 13 GB used for BERT.
- *Larger batch size:* from 256 used in BERT to 8 k.
- *Dynamic masking:* while static masking involves generating a fixed number (10 used in BERT) of masks for a sample, dynamic masking generates a new mask every time a sample is presented to the model.
- *Omitting the NSP objective:* using only the MLM objective alone improved the performance on downstream tasks.
- *Using byte-level BPE:* as opposed to the character-level tokenisation used in BERT.
- *Larger vocabulary:* from 30 k used in BERT to 50 k.
- *Refined hyper-parameter tuning.*

1.9.3 RobeCzech

RobeCzech (Straka et al., 2021) is a monolingual RoBERTa-like model for Czech. It outperformed larger multilingual models, and monolingual Czech models available at the time on Czech-specific NLP tasks. It was trained on Czech-only data from the following sources:

- SYNv4 (Křen; Cvrček; Čapka; Čermáková; Hnátková; Chlumská; Jelínek; Kovářiková; Petkevič; Procházka; Skoumalová; Škrabal; Truneček; Vondřička; A. Zasina, 2016): a corpus of contemporary Czech; part of the Czech National Corpus (4 188 M tokens),
- Czes (author, 2011): a collection of newspaper and magazine articles (432 M tokens),
- a subset of the Czech part of the W2C (Majliš, 2011) web corpus (16 M tokens),
- plain texts from Czech Wikipedia (123 M tokens).

[7] By 'sentence' the authors mean an arbitrary span of contiguous text rather than a sentence in the linguistic sense. For clarity, we use the term 'text segment' to refer to the same concept.

1.10 Decoder-Only Transformer Models

As a complement to the encoder-only models, the *decoder-only* models contain only the decoder, also omitting the cross-attention module. The main difference from the encoder-only models is the attention masking, in which each token can only attend to previous tokens in the sequence. The architecture can be used to generate text given some context, as first demonstrated by P. J. Liu et al. (2018). The decoder-only models are used to create 'AI assistants' that can be prompted for various tasks that the models were not explicitly trained for.

1.10.1 Decoding Strategies

When generating a text, the model is presented with a text prefix and returns a probabilistic distribution over the entire vocabulary as the candidates for the next token. While a straightforward solution – *greedy sampling*, where the most probable next token is selected each time – exists, it is not guaranteed to output the most probable sequence. Furthermore, some applications – such as generating multiple answers for the same prompt for RLHF (see the end of Section 1.10.2) – benefit from nondeterminism.

Beam search. *Beam search* partially addresses the sub-optimality of greedy sampling by maintaining $\beta \in \mathbb{N}$ prefixes that are iteratively updated. In each step, the next token distribution is computed for each of the β prefixes, and the top β tokens with the highest probability are considered for each prefix (yielding β^2 hypotheses). Out of the new possible hypotheses, the top β most probable are selected for the next iteration (Massarelli et al., 2020). Although this method is used in MT, it is not preferred for decoder-only models, as it often leads to text degeneration (Holtzman et al., 2020).

Top-k sampling. The *top-k* sampling strategy (Fan et al., 2018) in each iteration only considers the top k ($0 < k \leq \text{vocab\_size}$) most probable tokens. The top k probabilities are normalised such that their sum equals 1, and the next token is sampled out of the k candidates following the new probability.

Top-p sampling. *Top-p* sampling (Holtzman et al., 2020), also referred to as *nucleus sampling*, is an extension of the top-k sampling strategy that works with a dynamic number of tokens in each iteration. For a given $p \in [0, 1]$, the minimum number of the most probable tokens is considered, such that the sum of their probabilities is $\geq p$. Similarly to top-k, the probabilities are normalised and a new token is sampled.

Sampling with temperature. Introducing temperature $t > 0$ to a sampling procedure involves changing the final softmax activation (see Section 1.5.1) to

$$\text{softmax}(\mathbf{x}; t)_i = \frac{\exp(x_i/t)}{\sum_j \exp(x_j/t)}$$

Setting the temperature to 1 has no effect on the distribution, while setting it to a number smaller than 1 (which is the typical use case) skews the distribution towards more probable tokens (Holtzman et al., 2020).

1.10.2 GPT

Generative Pre-trained Transformer (GPT) is a family of decoder-only Transformer models developed by OpenAI. The earlier models – GPT-1 (Radford; Narasimhan et al., 2018), GPT-2 (Radford; Jeffrey Wu et al., 2019b) – are open-source, while the later models – GPT-3 (Brown et al., 2020), GPT-4 (OpenAI; Achiam et al., 2024), etc. – are commercial and closed-source.

The pre-training procedure introduced for GPT-1 utilised the *next token prediction*: given tokens $\mathcal{U} = \{u_1, \dots, u_n\}$, at position i, the model is trained to predict the distribution $p(u_i \mid u_{i-k}, u_{i-k+1}, \dots, u_{i-1})$ where k is the context window. Despite the generative pre-training, GPT-1 was not primarily used to generate text; it was rather fine-tuned with a special `[EXTRACT]` token added to the end[8] and a task-specific classification head connected to it.

In GPT-2, the model's capacity was increased significantly, from 117 M parameters to 1 542 M. The model was able to perform on downstream tasks, such as question answering or machine translation, without any further training, i.e. *zero-shot*.

The creation of GPT-3 involved an even greater increase in the model's capacity, to 175 B parameters. The authors demonstrate how the model was able to generate the correct text after being presented with a few examples – i.e. *few-shot learning*[9]. The model was later *instruction-tuned* to be used for ChatGPT, an 'AI assistant'.

InstructGPT (Ouyang et al., 2022) describes the process of tuning a pre-trained LLM to respond to user prompts in a dialogic way and aligning its response with human preferences. The process is composed of three steps:

1. A human-written diverse dataset of (user prompt, assistant response) pairs is created, and the model is fine-tuned similarly to pre-training.

2. Multiple responses are generated for the same prompt, and human annotators label the preference between them. The collected data is then used to train a smaller, GPT-based *reward model* that predicts the preference (reward) of a given (user prompt, assistant response) pair.

3. The reward model is used to fine-tune the model resulting from step 1 using Proximal Policy Optimisation[10] (PPO; Schulman et al., 2017).

The first step is referred to as *instruction tuning*, while the final two steps are referred to as Reinforcement Learning From Human Feedback (RLHF).

The later models, including GPT-4o (OpenAI; Hurst et al., 2024), which we used in our dataset generation (see Chapter 3), are entirely closed-source, and the number of parameters, as well as the specific training procedures, are unknown.

[8] Note that unlike in the BERT architecture, the token aggregating the information needs to be added to the *end* of the sequence, as tokens in the decoder can only attend to previous tokens.

[9] Learning in this case does not involve any change to the model's parameters; only including a few examples in the context window.

[10] An algorithm for reinforcement learning using gradient methods.

1.10.3 Llama

Llama (also spelled as *LLaMa*) is a family of *open-source* decoder-only Transformer models developed by Meta AI. The models roughly follow the training procedure described above. Some implementation details of Llama 3 include (Grattafiori et al., 2024):

1. Model size of 405 B parameters (the largest model in the family).
2. The largest model in te family trained on 15.6 T tokens.
3. Trained with context window gradually increased from 8 k to 128 k.
4. For alignment with human preferences, using Direct Preference Optimization (DPO; Rafailov et al., 2023), in which the preferences are learned directly, without a separate reward model, as opposed to PPO.

2 Related Work

The text generated by LLMs has been studied intensively since the release of ChatGPT (based on GPT-3; Brown et al., 2020) and the subsequent growth in popularity of LLM assistants. In this chapter, we provide an overview of previous work regarding the use of LLMs for plagiarism, generated text detection and its biases.

2.1 Generated Text and Plagiarism

The concern of using LLM-generated text for academic misconduct has been raised in academia. Susnjak and McIntosh (2024)[1] provided an early examination of the capabilities of ChatGPT in answering open-ended questions from different academic domains that require complex reasoning and knowledge. The authors raised a concern that LLMs may pose a threat to academic integrity, especially in online examinations.

In the study, ChatGPT was first prompted to generate a question for a given domain and then to answer it using 500 words, in several paragraphs. The quality of the responses was evaluated in the following criteria: relevance, clarity, accuracy, precision, depth, breadth, logic, persuasiveness, and originality. Upon the analysis of the responses, the authors reported satisfactory results in all the criteria but two: accuracy and originality. There was a 'tradeoff' observed between the two: the model can either answer truthfully but without much originality, or it can *hallucinate* an untruthful answer, which is inherently original.

In the final part, the authors provide certain strategies to prevent academic misconduct using ChatGPT. Notably, using multimodal questions – with images embedded – was believed to make cheating more challenging. However, with the introduction of multimodal LLMs (H. Liu et al. (2023), Gemini Team et al. (2023), OpenAI, Achiam et al. (2024), Grattafiori et al. (2024)), this suggestion became outdated. The authors further suggest using AI detectors or, as a more radical solution, returning to oral exams. AI detectors, therefore, remain the only feasible solution mentioned for misconduct in online exams.

2.2 Detecting Generated Text

To our knowledge, no work was published specifically on the automatic detection of AI-generated text in Czech at the time of writing this thesis. As an important contribution, the work of Milička et al. (2025), which focuses on humans' ability to detect generated text, shows that humans were able to detect generated text better than random baseline (55.4% correctness) and improved to 65.1% when presented with feedback about the true author of the text, which we do not find sufficient.

Junchao Wu et al. (2025) provide a survey on the problem of generated text detection, including the motivation, suggested high-quality datasets, an overview of common methods, and their performance. The following subsections are largely

[1]Preprint published in 2022.

based on the survey, covering: (1) classical ML classifiers working with the the bag-of-words model, (2) logit-based detectors working with the logits of a reference LLM when trying to predict a given document, (3) linguistec feature-based detectors working with vectors of linguistic features such as the diversity of vocabulary, and (4) pre-trained classifiers typically based on a BERT-like model.

2.2.1 Classical ML Classifiers

Various works attempted to create an approach that uses simple, classical ML methods, either to be used as a baseline or for simple deployment. Solaiman et al. (2019) describe creating a logistic regression classifier with TF-IDF (see Section 1.4.1) features, which achieved better results than simple logit-based methods. Similarly, Najjar et al. (2025) used TF-IDF features in combination with ML algorithms such as random forests (Ho, 1995) and the SVM (Cortes; Vapnik, 1995). The study found that these methods achieved a near-perfect accuracy for entire articles, but the accuracy decreased with individual paragraphs, which are considerably shorter.

2.2.2 Logit-Based Detectors

The *logit-based detectors* are a family of detectors which use the logits (the outputs of the final layer before the softmax activation) of a reference LLM to detect generated text. The reference model can either be the same as the generated text source model or a 'similar' model, since the source model is often unavailable. For each prefix and each term in the vocabulary, the logits indicate the confidence of the term being generated as the next token. This method does not require any training.

The most straightforward method of utilising the logits is computing the *log-likelihood* (the average token-wise log probability) of a text. This method was implemented by Solaiman et al. (2019) but was found to be rather inaccurate. Regardless, more complex logit-based methods have been used successfully.

GLTR (Gehrmann et al., 2019) computes the *Rank-measure* for each token. Given a document prefix $d_1, \ldots, d_{i-1}$, the rank assigned to position i is the rank of the true i-th token d_i when sorted by the probability distribution over tokens $p(x \mid d_1, \ldots, d_{i-1})$. The ranks are then visualised by dividing them into categories such as Top-10, Top-100, and Top-1000. Using this visualisation, humans improved their accuracy at detecting generated text from 54.2% to 72.3%. As an automated and improved extension of the Rank measure, Jinyan Su et al. (2023) suggest classifying documents using the ratio of log-likelihood and mean log-rank and show that the performance of such a classifier is considerably better than using one of the methods alone.

Vasilatos et al. (2023) use the *perplexity*, which has a straightforward relationship with log-likelihood: $PPL = \exp(-\text{log\_lik})$. The authors found that, on average, the perplexity of generated texts was lower than that of human-written ones. However, the optimal perplexity threshold for classification differed among various 'flavours' of documents, such as code and math.

DetectGPT (E. Mitchell et al., 2023) introduced a non-trivial method that combines the logit-based methods with text perturbations. The key observation is that when introducing minor perturbations to a text using a gap-filling pre-trained model such as the T5 (Raffel et al., 2020), the unmodified AI-generated text tends to have a larger log-likelihood than its perturbations. In contrast, perturbing a human-written text does not systematically decrease its log-likelihood. The proposed method outperformed the simpler methods mentioned above.

2.2.3 Linguistic Feature-Based Detectors

Linguistic feature-based detectors are based on the extraction of certain linguistic features and representing the documents as vectors of such features. Subsequently, the vectors are classified using ML methods. The extraction of the specific features requires expertise in the task. This method was adapted from studies on fake news detection, such as Aich et al. (2022).

Shah et al. (2023) constructed a detector based on three categories of linguistic features: (1) *lexical features*, including the frequency of functional words, word length distribution, sentence length distribution, and the correctness of punctuation, (2) *readability scores*, which utilises multiple established classical metrics for text readability, such as the Flesch Reading Ease and Gunning Fog Index, and (3) *diversity of the vocabulary*, which, again, employs various metrics of vocabulary diversity, including Yule's Characteristic K (Yule, 1944), Herdan's C (Herdan; Sharvit, 2006), Type-Token Ratio (TTR), etc. The authors applied various ML methods on the feature vectors, including logistic regression, decision trees, SVM, etc. Consequently, the study used explanation tools to find the most contributing features and found the most dominant one to be Herdan's C.

Corizzo and Leal-Arenas (2023) created a classifier based on the fusion of LSTM networks (Hochreiter; Schmidhuber, 1997) and linguistic features. Three feature categories of linguistic features were proposed: text, repetitiveness, and emotional semantics. The fused representation was subsequently processed with multiple hidden feed-forward layers, leading to a deep-learning architecture. The model outperformed all other baseline architectures inspected in the article.

2.2.4 Pre-Trained Classifiers

Fine-tuning a pre-trained language model is a straightforward method for the detection of generated text, employing non-trivial models. Typically, encoder-only (see Section 1.9) models are used, such as BERT (Devlin et al., 2019) and RoBERTa (Y. Liu et al., 2019). These models are often considered for baseline solutions and have been used since the beginning of the generated text detection problem (Solaiman et al., 2019).

While the detectors perform well within the domain, they lack cross-domain robustness. Antoun et al. (2023) experimented with the robustness of BERT-like detection models on English and French datasets and found that the accuracy dropped from $> 90\%$ to $< 30\%$ when changing domains and introducing misspellings. The authors suggest including various domains and writing styles in training data to combat this. While the authors work with a domain defined by the writing style and the topic, changing the source model of the generated text can also be considered a change of domain.

2.3 Bias in Generated Text Detection

Liang et al. (2023) examined how AI detectors work on text written by non-native speakers and found concerning results. This study is the most relevant work for our thesis, and we aim to follow up on its findings and expand them to Czech texts. The authors evaluated 7 'widely used' generated text detectors on two groups of documents: (1) Test of English as a Foreign Language (TOEFL) essays written by Chinese students (91 documents), and (2) Hewlett Foundation's ASAP dataset (Hamner et al., 2012), containing US eight-graders' essays (88 documents).

2.3.1 Original Study

Liang et al. (2023) found that, in most cases, the evaluated detectors correctly labelled the essays from US students as human-written, with the mean False Positive Rate (FPR) being 5.1%. In contrast, the TOEFL essays written by Chinese students were often misclassified as AI-generated, with the mean FPR of 61.3%. Furthermore, all 7 detectors unanimously flagged 19.8% of the TOEFL essays as AI-generated.

Upon inspection, the TOEFL essays that were unanimously flagged as AI-generated were shown to have a small perplexity (see Section 4.5.1 for more details on perplexity). The authors proceed to present a claim that *'most GPT detectors use text perplexity to detect AI-generated text'*. As a follow-up, in our thesis, we test whether the texts from non-native Czech speakers have smaller perplexities and whether we can create a classifier that does not rely on the perplexity. Our findings in the Czech setting differ considerably from those of Liang et al. (2023).

Afterwards, the study continued to demonstrate the lack of robustness of the detectors: Liang et al. (2023) prompted ChatGPT to enhance the vocabulary of the TOEFL. This automated change led to a drastic decrease in the mean FPR to 11.6%. This also led to an increase in perplexity. Moreover, simplifying the vocabulary in the US students' speakers led to an increase in FPR to 56.9%.

2.3.2 Leveraging Bias with Representation

Jiang et al. (2024) followed up on the study above. For the human-written documents, they used the Graduate Record Examinations (GRE)[2] essays submitted before the release of ChatGPT. In the GRE, participants self-report whether they are native English speakers, which was used to categorise the essays into two groups based on their nativity. The authors created two types of detectors: (1) feature-based, with features extracted using e-rater® (Attali; Burstein, 2006), which can supposedly extract features regarding the text quality, and (2) perplexity-based features, computed w.r.t. GPT-2 (Radford; Jeffrey Wu et al., 2019b), with features including overall perplexity, the mean, the median, and the interquartile range of sentence-level perplexities, etc.

Given the two sets of feature vectors, the ML classifiers such as SVM, linear regression, and random forests, were trained using one vector per sample (*feature* or *perplexity*) or both concatenated (*feature+perplexity*). The classifiers were then

[2] https://www.ets.org/gre.html

trained on data with equal representation of both native and non-native speakers. While all classifiers achieved near-perfect scores, the best classifier was found to be SVM working with *feature+perplexity* features.

Contrary to Liang et al. (2023), Jiang et al. (2024) found that the FPR was slightly larger in native speakers compared to the non-native ones, although the difference was negligible. The authors hypothesise that this was caused by more frequent grammatical errors in non-native speakers' theses, which made the classification simpler. As a downside of this study, we point out that the authors did not examine the situation where non-native speakers are underrepresented in the training data, which is often the case and would contribute more to the preliminaries for this thesis.

2.4 Model Explanation

Deep learning models, such as the Transformer-based (Vaswani et al., 2017) models, are known to be difficult to explain, i.e. to attribute the results to features that are understandable to humans. As a result, Explainable AI (xAI)[3] emerged as a major field in AI research (Longo et al., 2024). In this section, we present two xAI tools that we use later in the thesis.

2.4.1 Integrated Gradients

Integrated Gradients (IG; Sundararajan et al., 2017) is an explanation method defined specifically for deep learning models, where it is possible to calculate the gradient of the output w.r.t. the input. The authors formulate the attribution problem relative to some baseline input, to which the models behaves 'neutrally': suppose a prediction function $F : \mathbb{R}^n \to [0, 1]$, some input $\mathbf{x} \in \mathbb{R}^n$ and a baseline input $\mathbf{x}' \in \mathbb{R}^n$. Then, an attribution of the prediction F at an input $\mathbf{x}$, relative to the baseline $\mathbf{x}'$ is a vector $A_F(\mathbf{x}; \mathbf{x}') = (a_1, \dots, a_n)^\top \in \mathbb{R}^n$, where a_i is the contribution of the feature x_i.

The authors formulated two axioms that an explanation method should satisfy: (1) *sensitivity:* stating that for every input/baseline pair $(\mathbf{x}, \mathbf{x}')$ such that $\mathbf{x}$ and $\mathbf{x}'$ differ in exactly one feature $(x_i \neq x_i')$ and $F(\mathbf{x}) \neq F(\mathbf{x}')$, the differing feature's attribution a_i should be non-zero; and (2) *implementation invariance:* two networks F and F', regardless of their implementation, are functionally equivalent, if for each $\mathbf{x}$, it holds that $F(\mathbf{x}) = F'(\mathbf{x})$. As such, the attributions should be identical for F and F'.

In their work, Sundararajan et al. (2017) criticised the existing methods for not satisfying these axioms and proposed the IG method, which they prove to satisfy the axioms. The geometrical interpretation of the method is that a line segment in $\mathbb{R}^n$ is created between $\mathbf{x}$ and $\mathbf{x}'$, and the gradients along this path are computed and cumulated into an integral. The formal definition is as follows:

$$\mathrm{IG}_i^{(F)}(\mathbf{x}; \mathbf{x}') = (\mathbf{x} - \mathbf{x}') \int_{\alpha=0}^{1} \frac{\partial F(\mathbf{x}' + \alpha(\mathbf{x} - \mathbf{x}'))}{\partial x_i} d\alpha$$

In practice, the integral is approximated using a sum over small intervals, following the Riemann approximation. This reduces the problem of attribution to the

[3]Not to be confused with the xAI company.

problem of calculating the gradients, which is already implemented in libraries such as PyTorch (Paszke et al., 2019).

2.4.2 Local Interpretable Model-Agnostic Explanations

Local Interpretable Model-agnostic Explanations (LIME; Ribeiro et al., 2016) is a general framework for architecture-agnostic ('black-box') model explanation. The authors define the explanation task as providing a qualitative understanding between the sample's components and the model's prediction.

A key concept for LIME is the *local faithfulness*, which states that an explainer should faithfully correspond to the behaviour of the model in some local neighbourhood around a sample. This requirement is a relaxed version of complete faithfulness. Then, an explainer G is an explainable model (such as a linear model, where $G(\mathbf{x}') = \mathbf{w}_G^\top \mathbf{x}'$) that is locally faithful to the original model F around a sample $\mathbf{x}$. Given the trade-off between faithfulness and interpretability (which is an inverse measure of complexity), the task of finding the best explainer is the task of minimising the complexity, while maximising the faithfulness.

The explainer's domain ($\mathbb{R}^{d'}$) is not necessarily the same as the original model's domain ($\mathbb{R}^d$). For example, the explainer may work with full words (for better interpretability), while the model may work with subword units. For this reason, a distinction is made between the original sample $\mathbf{x}$ and its corresponding representation for the explainer $\mathbf{x}'$. For a given sample $\mathbf{x}$, LIME creates a dataset $\mathcal{Z}$ of pairs $(\mathbf{z}, \mathbf{z}')$, where $\mathbf{z}'$ is a randomly perturbed sample in the neighbourhood of $\mathbf{x}'$, and $\mathbf{z}$ is its corresponding reconstructed sample in the domain of F.

Given the original model F, the dataset $\mathcal{Z}$, and $\pi_\mathbf{x} : \mathbb{R}^d \to \mathbb{R}$ – a proxemity metric from $\mathbf{x}$, the new explainer model G, whose specific architecture and complexity depends on the specific task and can be controlled using hyperparameters, is trained by optimising the loss function:

$$\mathcal{L}(F, G, \pi_\mathrm{x}) = \sum_{\mathbf{z},\mathbf{z}' \in \mathcal{Z}} \pi_\mathrm{x}(\mathbf{z})(F(\mathbf{z}) - G(\mathbf{z}'))^2$$

After this step, the explainable features of the explainer model G (such as the weights of a linear model) are returned as the explanations.

3 Datasets

To train and evaluate our models, we sourced multiple datasets, aiming to include data from diverse domains. Some of the datasets were specifically created to inspect low-resource domains, such as texts produced by non-native speakers, pupils, and academics. Table 3.1 provides a brief overview of the datasets. Detailed descriptions are provided in the sections below.

Name	Description	Generated?
SYNv9$_{\text{NAT}}$	Texts from the Czech National Corpus	No
SYNv9$_{\text{GPT4o}}$	GPT-4o-generated complement for SYNv9$_{\text{NAT}}$	Yes
SYNv9$^{\text{TRAIN}}$	SYNv9$^{\text{TRAIN}}_{\text{GPT4o}}$ and its complement from SYNv9$_{\text{NAT}}$	Mixed
SYNv9$^{\text{VAL}}$	SYNv9$^{\text{VAL}}_{\text{GPT4o}}$ and its complement from SYNv9$_{\text{NAT}}$	Mixed
SYNv9$^{\text{VAL}}_{\text{4oMini}}$	GPT-4o mini-generated complement for SYNv9$_{\text{NAT}}$	Yes
SYNv9$^{\text{VAL}}_{\text{LLAMA}}$	Llama-generated complement for SYNv9$_{\text{NAT}}$	Yes
Wiki$_{\text{NAT}}$	Wikipedia articles	No
Wiki$_{\text{GPT4o}}$	GPT-4o-generated complement for Wiki$_{\text{NAT}}$	Yes
Wiki	Wiki$_{\text{GPT4o}}$ $\cup$ Wiki$_{\text{NAT}}$	Mixed
News$_{\text{NAT}}$	Czech internet newspapers	No
News$_{\text{GPT4o}}$	GPT-4o-generated complement for News$_{\text{NAT}}$	Yes
News	News$_{\text{GPT4o}}$ $\cup$ News$_{\text{NAT}}$	Mixed
NonNative	Text from non-native Czech speakers	No
NonNativeC1	Text from advanced non-native Czech speakers	No
NatYouth	School texts from native Czech youth	No
NatAdv	School texts from native Czech speakers aged 16–18	No
Abs2020	Pre-ChatGPT theses abstracts	No
AbsNew	Post-ChatGPT theses abstracts	Unclear

Table 3.1 Overview of the used datasets.

3.1 Conteporary Czech Corpus

SYNv9 (Křen; Cvrček; Henyš et al., 2021) is a collection of synchronic corpora from SYN2000 to SYN2020, with an addition of a journalistic component predominantly from 2010–2019. For our purposes, we only used the text published between the years 2009 and 2019 (which is the latest year included). Therefore, we included texts from:

- SYN2015 (Křen; Cvrček; Čapka; Čermáková; Hnátková; Chlumská; Jelínek; Kováříková; Petkevič; Procházka; Skoumalová; Škrabal; Truneček; Vondřička; A. J. Zasina, 2016), a representative corpus containing 114 492 written texts published between the years 2010 and 2014, composed of three equally sized parts: fiction, non-fiction, and newspapers/magazines.
- SYN2020 (Jelínek et al., 2021), a representative corpus containing 114 211 written texts published between the years 2015 and 2019; with the same composition as SYN2015.

```
[CS]
Napište dalších 2000 slov tohoto textu. Pište pouze
samotný text.

<text sample>

[EN]
Write the following 2000 words of this text. Write the
actual text only.

<text sample>
```

Listing 3.1 Prompt for the synthetic text generation (only the `CS` version was used). The `<text sample>` is either the first paragraph, if its length is between 100 and 1000 characters or the n sentences ended by a full stop, such that $n \in \mathbb{N}$ is that smallest number that makes the number of characters in the sample at least 150.

- The journalistic component, containing texts from national daily newspapers, regional daily newspapers, and non-specialised magazines.

We included 7460 texts from the corpus (published in 2009–2019), selected at random. We truncated the texts to 2000 tokens, using the tokenisation provided in the corpus annotation. This dataset is referred to as $\text{SYNv9}_{\text{NAT}}$.

3.1.1 Generated Complement

The positive samples were generated using an LLM. To match the structure and vocabulary of the natural corpus in the positive samples, we used the prompt in Listing 3.1 for instruction-tuned language models.

We chose to use instruction-tuned models because we observed that their outputs are less often repeated, too short, or nonsensical, even though text completion is a typical task for non-instruction-tuned models. Additionally, using the prior allows us to consistently use closed-source models, too. Note that the prompt in Listing 3.1 results in a text that starts with an 'out-of-context' paragraph, since the model only continues the text after it is presented with the context. This could be exploited by the detectors to trivially distinguish the generated texts if the human-written text started with an introductory paragraph. Thankfully, the texts in SYNv9 also start out of context.

GPT-4o. As the main source of generated text, we use GPT-4o, which has been the most well-known model to the general public. The version used was `gpt-4o-2024-05-13` (OpenAI; Hurst et al., 2024), with the temperature 0.7 and number of tokens limited to 1024. We generated the data, discarded the files smaller than 2 kB, and split them into two subsets: $\text{SYNv9}^{\text{TRAIN}}_{\text{GPT4o}}$ (2383 samples) and $\text{SYNv9}^{\text{VAL}}_{\text{GPT4o}}$ (299 samples).

For training, we matched the $\text{SYNv9}^{\text{TRAIN}}_{\text{GPT4o}}$ samples with the natural samples that were used to generate them. This subset of $\text{SYNv9}_{\text{NAT}}$ is denoted by $\text{SYNv9}^{\text{TRAIN}}_{\text{NAT}}$. Analogously, we define $\text{SYNv9}^{\text{VAL}}_{\text{NAT}}$. We further define $\text{SYNv9}^{\text{TRAIN}} := \text{SYNv9}^{\text{TRAIN}}_{\text{NAT}} \cup \text{SYNv9}^{\text{TRAIN}}_{\text{GPT4o}}$ and similarly its VAL counterpart.

```
[CS]
Napište Wikipedia článek na téma <article name>

[EN]
Write a Wikipedia article on the topic <article name>
```

Listing 3.2 Prompt for the synthetic Wikipedia articles generation (only the `CS` version was used). The <article name> was substituted for the corresponding article name from the crawled articles.

GPT-4o mini. For additional validation, we generated a dataset using `gpt-4o-mini-2024-07-18`[1] in a similar way: $\text{SYNv9}^{\text{VAL}}_{\text{4oMini}}$ (287 samples).

Llama 3.1 405B. To explore models from different families, we generated data using `llama3.1:405b-instruct-q5_K_S`[2]. We used the same workflow as above. Upon inspection, the quality of this dataset was found to be lower, with shorter paragraphs, occasional language switching, and repeated sentences. We denote this dataset by $\text{SYNv9}^{\text{VAL}}_{\text{Llama}}$ (301 samples).

3.2 Czech Wikipedia Crawl

To include more domains, we added Wikipedia articles for evaluation. The dataset was created by crawling Wikipedia using `pywikibot`[3]. Articles were chosen randomly, retrieved and parsed using `mwparserfromhell`[4]. Articles that contained less than 1000 `<SPACE>` characters were discarded. This resulted in the dataset denoted by Wiki_{nat} (286 samples).

3.2.1 Generated Complement

Similarly to the corpus above, we created the generated complement to match the vocabulary of the natural texts. We used `gpt-4o-2024-05-13` with the same parameters and workflow as before, and the prompt displayed in Listing 3.2. This dataset is denoted by $\text{Wiki}_{\text{GPT4o}}$ (211 samples). By Wiki we denote the union of $\text{Wiki}_{\text{GPT4o}}$ and its corresponding subset of Wiki_{nat}.

3.3 Czech News Crawl

Another domain included in our test data is the news crawl[5], crawled from internet newspaper sites, for the year 2021. Although journalistic texts are

[1] `https://openai.com/index/gpt-4o-mini-advancing-cost-efficient-intelligence`

[2] Llama 3.1, 405 B parameters, quantised to `Q5_K_S` (Grattafiori et al., 2024), downloaded from and hosted using Ollama (`https://ollama.com/`), version from 2024-09-16. Quantisation is the process of reducing the model's weight precision. Gholami et al. (2022) showed that quantising a model to 4 bits only decreases its performance by $\sim 2\%$.

[3] `https://www.mediawiki.org/wiki/Manual:Pywikibot`

[4] `https://github.com/earwig/mwparserfromhell`

[5] `https://data.statmt.org/news-crawl/doc/cs/`

```
[CS]
Napište novinový článek na téma '<article name>'

[EN]
Write a news article on the topic '<article name>'
```

Listing 3.3 Prompt for the synthetic news articles generation (only the `CS` version was used). The <article name> was substituted for the corresponding article name from the selected new articles. Compared to the prompt for Wikipedia, the news article names were more complex, so we introduced quotes to distinguish them from the rest of the prompt.

contained in SYNv9, the structure of online news articles may likely be different from the printed ones. The format of the texts in the crawl is one sentence per line, which loses information about paragraphs. We selected 381 articles from the crawl at random. We denote this dataset by News_{NAT}.

3.3.1 Generated Complement

As before, we created the generated complement. We used `gpt-4o-2024-05-13` with the same parameters and workflow as before, and the prompt displayed in Listing 3.3. This resulted in the dataset denoted by $\text{News}_{\text{GPT4o}}$. All 381 generated articles were sufficiently long. Again, we define $\text{News} := \text{News}_{\text{GPT4o}} \cup \text{News}_{\text{NAT}}$.

3.4 Non-Native Czech Speakers Corpus

As a crucial dataset in determining the performance of our classifiers on text by non-native speakers, we used the AKCES 3 corpus (Šebesta; Bedřichová et al., 2012), a corpus of texts produced by non-native students of the Czech language. To avoid texts with too frequent mistakes, we only included speakers who had studied Czech for at least 24 months at the time of writing. The dataset is referred to as NonNative (450 samples).

While we recognise that 24 months is a relatively short period to acquire a language and that prior work examined texts from learners with significantly longer experience with the language, finding a better-suited Czech corpus proved to be challenging. To partially address this issue, we used the AKCES-GEC corpus (Náplava; Straka, 2019), which, among others, contains a version of AKCES 3 extended by newer articles.

From there, we extracted 118 articles and 1 article, with the speaker's language level marked as C1 and C2, respectively. However, upon inspection, these texts did not exhibit significantly better quality compared to AKCES 3 and instead commonly contained ungrammatical or missing words. Furthermore, their extent was often rather short – only 29 being larger than 2 kB. Subsequently, we created a dataset containing the mentioned 29 documents, referred to as NonNativeC1.

3.5 Native Czech Youth Corpus

To roughly match the domain of NonNative, we utilised the AKCES 1 corpus (Šebesta; Goláňová et al., 2016), a collection of texts produced by native Czech speakers at primary (from 5th grade) and secondary schools. Furthermore, we attempted to match the distribution of the file size of the native texts to the NonNative dataset. This was done by selecting the most similar (in file size) text from AKCES 1 for each text in the NonNative dataset. The resulting subset of AKCES 1 is denoted by NatYouth (450 samples).

To match the advanced non-native texts from NonNativeC1, we randomly selected 29 texts from AKCES 1 with the age category labelled as 'over 15 years' – i.e. 16–18 years. We denote this dataset by NatAdv.

3.6 Corpus of pre-ChatGPT Abstracts

To evaluate the models on academic texts, we created a corpus of theses abstracts, crawled from the Charles University Digital Repository[6]. We include abstracts from the students of the Faculty of Arts (which is the faculty that published the most theses), between the years 2020 and 2021 – i.e., before ChatGPT was introduced. We denote this dataset by Abs2020 (1655 samples).

3.7 Corpus of post-ChatGPT Abstracts

For comparison, we also included the abstracts written in 2023–2025 – after ChatGPT became widely popular. By using the same criteria as before, we created the dataset AbsNew (2050 samples).

[6] `https://dspace.cuni.cz/`

3.8 Data Normalisation and Augmentation

We preprocessed the data before feeding it into the models using various pipelines for different purposes. In this section, we discuss their role and functionality. Table 3.2 contains a brief overview of the normalisation and augmentation pipelines. The pipelines are described in detail in the sections below.

Name	Description
SpaceNorm	Replacing any sequence of whitespace with a single space.
PunctNorm	SpaceNorm + replacing diacritics with their standardised variant.
LowNorm	Lowercasing each character.
RandAug	Adding random Unicode 'words' and whitespace between the words.

Table 3.2 Overview of the used normalisation and augmentation.

3.8.1 Normalisation

Whitespace normalisation. Whitespace normalisation is the process of replacing any sequence of whitespace characters with a single space. The aim is to hide the text's surface structure (paragraphs, double spaces after sentences), which might be a trivial giveaway for detecting generated text. We refer to this approach as SpaceNorm.

While the whitespace-based tokenisers, such as `unitok` (Suchomel et al., 2014), are invariant to whitespace normalisation, the RobeCzech (Straka et al., 2021) tokeniser includes the whitespace in the tokenised representation.

Punctuation normalisation. Alongside whitespace, the use of specific punctuation symbols may be used to detect generated text based on the surface structure trivially. This is demonstrated in Section 4.1.3. To avoid learning trivial patterns, we utilise the `MosesPunctNormalizer` from the `sacremoses`[7] library on top of SpaceNorm. This pipeline is referred to as PunctNorm.

The normaliser follows a rule-based approach to replace punctuation symbols with their standardised equivalent, creating equivalence classes on punctuation symbols. For example, both `„word"` and `«word»` will be normalized to `"word"`.

Lowercasing. Lowercasing a text involves replacing every text character with its lowercase equivalent. This, in turn, creates equivalence classes on words. For whole-word tokenisers, such as `unitok`, this reduces the total number of unique tokens (i.e. terms), which reduces the total number of learned parameters of a model. This process is referred to as LowNorm.

3.8.2 Random Noise Augmentation

As we observed in Section 4.2.4, some models may exploit the 'rarity' of tokens for classification. To better understand this, we add random noise to the text as a form of augmentation. The process first employs the PunctNorm pipeline

[7] `https://github.com/hplt-project/sacremoses`

and subsequently adds random Unicode and whitespace noise. This pipeline is referred to as RandAug.

Unicode Noise. Adding random unicode noise involves inserting random 'words' – sequences of randomly generated printable Unicode symbols. First, the number of words to add is determined: for a sequence of w words and the expected inflation factor of 0.02, the number of added words will be $\max(0, \lfloor x \rceil)$, where $x \sim \mathcal{N}\left(0.02w, \frac{0.02w}{5}\right)$. Next, each word is generated by determining its length as $max(0, \lfloor y \rceil)$, where $y \sim \mathcal{N}(1, 1)$, and generating such a sequence of characters. The words are then inserted into positions generated at random, with repetition.

Whitespace Noise. After adding random words, we join both the original and the inserted words with random whitespace. With the probability of 0.97, we use a single space character. Otherwise, we use a sequence of $max(0, \lfloor z \rceil)$ where $z \sim \mathcal{N}(1, 0.2)$, whitespace characters from the following list: `\n`, `\t`, `\r\n`, `\n\n`, `\r\n\r\n`, and `\u00a0` (The *no-break space* – ` `).

4 Experiments

This chapter contains all the experiments we conducted for the thesis. It is structured as follows: Sections 4.1, 4.2, 4.3, and 4.4 discuss the creation and examination of various types of generated text detectors. Section 4.5 presents a logit-based analyser, which we use to analyse the entropies of the datasets. Finally, section 4.6 provides a comparison of the presented methods using correlation analysis. For reproducibility, we provide the scripts used in the experiments in a GitLab repository[1].

4.1 TF-IDF Naïve Bayes Classifier

Before attempting to construct a robust classifier, we aimed to create a baseline model for comparison. We chose a lexical-based approach (see Section 2.2.1), specifically with TF-IDF (see Section 1.4.1) features. As a typical approach (see Section 1.4.1), we chose the Multinomial NB Classifier. The section discusses the training and evaluation of the classifier with two tokenisation approaches in Sections 4.1.1 and 4.1.2, and inspects their functionality in Section 4.1.3.

4.1.1 Training and Evaluation with the Unitok Tokeniser

For the feature-engineering pipeline, we employed the `unitok` (Suchomel et al., 2014) tokeniser, LowNorm, and the `TfidfVectorizer`[2] from the `scikit-learn` library (Pedregosa et al., 2011) with the default parameters except for the `tokenizer` parameter. Following the default parameters, the vectoriser does not include tokens with the absolute document frequency ≤ 1 and relative document frequency of 100% in the vocabulary (see Section 1.4.1 for definitions).

For the classifier, we used the `MultinomialNB` from the same library, with default parameters – i.e., Laplace smoothing $\alpha = 1$. We trained the model on the dataset SYNv9$^{\text{TRAIN}}$. The number of unique tokens after normalisation (i.e. terms), which is also the dimension of the TF-IDF vector, was 162 109. The results are shown in Table 4.1.

From Table 4.1 we can see that the classifier reached a near-perfect (> 99%) accuracy on training data SYNv9$^{\text{TRAIN}}$ and maintained it on unseen data from the same domain SYNv9$^{\text{VAL}}$ and similarly on SYNv9$^{\text{VAL}}_{\text{4oMini}}$, suggesting that GPT-4o (OpenAI; Hurst et al., 2024) and GPT-4o mini[4] produce texts that are in some sense similar. On the other hand, the classification model performed considerably worse on a dataset from the same domain generated by an LLM from an entirely different family (SYNv9$^{\text{VAL}}_{\text{Llama}}$).

Furthermore, the classifier performed worse on domains different from the training domain, such as Wiki and performed catastrophically – i.e. considerably

[1] https://gitlab.mff.cuni.cz/alalia/detectors-thesis

[2] https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html

[3] The metrics suppose that all the abstracts are human-written, which is not guaranteed.

[4] https://openai.com/index/gpt-4o-mini-advancing-cost-efficient-intelligence

Dataset	Acc	FPR	FNR	Unk Rate	Pre-Processing
SYNv9$^{\text{TRAIN}}$	99.12%	0.80%	0.97%	13.07%	
SYNv9$^{\text{VAL}}$	99.17%	1.00%	0.67%	15.44%	
SYNv9$^{\text{VAL}}_{\text{4oMini}}$	98.95%	–	1.05%	9.38%	
SYNv9$^{\text{VAL}}_{\text{Llama}}$	73.09%	–	26.91%	13.84%	
Wiki	88.93%	9.44%	13.27%	25.12%	
News	98.17%	3.41%	0.26%	14.98%	LowNorm
NonNative	91.33%	8.67%	–	22.18%	
NonNativeC1	75.86%	24.14%	–	14.83%	
NatYouth	93.78%	6.22%	–	17.78%	
NatAdv	75.86%	24.14%	–	15.78%	
Abs2020	19.82%	80.18%	–	17.02%	
AbsNew[3]	17.90%	82.10%	–	17.24%	

Table 4.1 Evaluation results of the TF-IDF NB Classifier. The 'Unk Rate' column corresponds to the ratio of unknown tokens (not in the vocabulary) in the dataset.

worse than random – on Abs2020 and AbsNew.

As for the potential bias against non-native speakers, the increase in FPR for NonNative compared to NatYouth was 2.45%, which is arguably a negligible difference, possibly caused by a slight difference in the datasets' domains. There was no difference in performance between NonNativeC1 and NatAdv. Therefore, we conclude that this classifier does not exhibit a systematic bias against non-native speakers.

4.1.2 Training and Evaluation with the RobeCzech Tokeniser

For comparison, we further trained a similar model with the RobeCzech (Straka et al., 2021) tokeniser and no normalisation. This bounds the dimension of the feature vector to a maximum of 52 000, which is the vocabulary size for RobeCzech. The actual feature vector dimension was 49 998.

While the use of the RobeCzech tokeniser significantly decreased the unknown token rate (thanks to subword tokenisation, which also lowered the vocabulary size to less than a third of that from Section 4.1.1), it did not achieve a consistent improvement across all the datasets. Specifically, the FNR on the Wiki dataset increased significantly. The detailed results are shown in Table 4.2.

Regarding the potential bias against non-native speakers, the difference in performance between NonNative and NatYouth was, again, negligible. However, there was a considerable difference between NonNativeC1 and NatAdv (13.79% increase in FPR for non-native speakers), suggesting a possible bias.

4.1.3 Explaining the Classifier

One of the benefits of using a straightforward classifier such as the NB is its explainability: the classifier works with a bag-of-words feature vector, and an observable weight is assigned independently to each feature. In this section, we

Dataset	Acc	FPR	FNR	Unk Rate	Pre-Processing
SYNv9$^{\text{TRAIN}}$	98.70%	0.55%	2.06%	0.00%	
SYNv9$^{\text{VAL}}$	98.83%	.33%	2.01%	0.13%	
SYNv9$^{\text{VAL}}_{\text{4oMini}}$	98.95%	–	1.05%	0.05%	
SYNv9$^{\text{VAL}}_{\text{Llama}}$	72.43%	–	27.57%	0.13%	
Wiki	80.28%	4.9%	39.81%	0.45%	
News	96.72%	6.3%	0.26%	0.17%	–
NonNative	93.11%	6.89%	–	0.26%	
NonNativeC1	65.52%	34.48%	–	0.20%	
NatYouth	93.56%	6.44%	–	0.21%	
NatAdv	79.31%	20.69%	–	0.21%	
Abs2020	39.46%	60.54%	–	0.18%	
AbsNew	38.49%	61.51%	–	0.19%	

Table 4.2 Evaluation results of the TF-IDF NB Classifier with the RobeCzech tokeniser. The 'Unk Rate' column corresponds to the ratio of unknown tokens (not in the vocabulary) in the dataset.

attempt to explain the classifier. First, on the classifier level, we show the most contributing terms and induce the possible general rules that apply. Next, we introduce an intuitive method for sample-level explanation.

Most contributing terms – unitok tokeniser. Given the functionality of the NB Classifier, we can extract the k terms contributing the most to positive and negative classification as

$$\underset{t \in \text{vocab}}{\text{Topk}} \left(\log p\left(t|c=1\right) - \log p\left(t|c=0\right)\right)$$

and

$$\underset{t \in \text{vocab}}{\text{Topk}} \left(\log p\left(t|c=0\right) - \log p\left(t|c=1\right)\right)$$

respectively, where c is the class (1 for positive classification). For $k = 20$, we list the terms contributing the most to positive (Table 4.3a, left) and negative (Table 4.3b, right) classification.

We can conclude from the comparison that while the generated texts included predominantly adjectives, nouns, and verbs expressing exactness and enumeration, the natural texts more often contained adverbs, conjunctions, and nouns expressing time. Furthermore, the terms in the natural dataset frequently expressed inexactness and had informal or literary tone.

Most contributing terms – RobeCzech tokeniser. We applied the equivalent procedure to the modified classifier from Section 4.1.2. The results are shown in Tables below: 4.4a left for the most positive contributions and 4.4b right for the most negative contributions.

While some terms are the same or equivalent to Table 4.3, there are three significant observations to be made: (1) the newline character (and subsequently the paragraph structure) is the most contributing factor for positive classification, (2) a large part of the positively contributing terms begin with a capital letter,

#	Term	Translation
1	důležitým	important$_{ins}$
2	zlepšit	to improve
3	aspektem	aspect$_{ins}$
4	důležité	important$_{nom}$
5	mezitím	meanwhile
6	klíčové	key$^{adj}_{nom}$
7	výzvám	challenges$_{loc,pl}$
8	přispět	to contribute
9	odhodlání	determination$_{nom}$
10	zahrnovat	to include
11	klíčových	key$^{adj}_{gen,pl}$
12	jedním	one (of)
13	zahrnuje	includes
14	výzvy	challenges$_{nom}$
15	zlepšení	improvement$_{nom}$
16	dosáhnout	to achieve
17	dopad	impact$_{nom}$
18	novák	Novák$_{nom}$
19	rovnováhu	balance$_{acc}$
20	vést	to lead

(a) Terms contributing the most to positive classification.

#	Term	Translation
1	prý	supposedly [inf.]
2	zhruba	rougly
3	loni	last year
4	ovšem	however
5	procent	per cent
6	korun	crowns$_{gen}$
7	asi	approximately [lit.]
8	včera	yesterday
9	roku	year$_{gen}$
10	tisíc	thousand
11	miliardy	billion$_{gen}$
12	jenže	however
13	půl	half
14	čtyři	four
15	šest	six
16	velice	very [lit.]
17	totiž	namely
18	kč	CZK
19	či	or [lit.]
20	sobotu	Saturday$_{acc}$

(b) Terms contributing the most to negative classification.

Table 4.3 Terms contributing to classification. Novák (Table a, row 18) is a Czech name commonly used as a fictional character, e.g. in math puzzles.

suggesting that the beginnings of sentences are used for detection, and (3) a large part of the negatively contributing terms are punctuation, specifically symbols that are rather rare in Czech, placed in the top 4 positions.

Sample-level explanation. The method described in the previous subsections is useful for a general understanding of the classifier. However, a more in-depth analysis may require an explanation of how individual samples are classified. We present a method of visualising each token's contribution to the final prediction.

Suppose a TF-IDF vector sample $\mathbf{s}$. Then the *log-odds* of $\mathbf{s}$ being classified as positive are:

$$\begin{aligned}
\log \text{odds}\,(c=1|\mathbf{s}) &= \log \frac{P(c=1)\prod_{t\in\text{vocab}} p(t|c=1)^{s_t}}{P(c=0)\prod_{t\in\text{vocab}} p(t|c=0)^{s_t}} \\
&= \log \frac{\prod_{t\in\text{vocab}} p(t|c=1)^{s_t}}{\prod_{t\in\text{vocab}} p(t|c=0)^{s_t}} \\
&= \sum_{t\in\text{vocab}} (s_t \cdot \log p(t|c=1)) - \sum_{t\in\text{vocab}} (s_t \cdot \log p(t|c=0)) \\
&= \sum_{t\in\text{vocab}} (s_t \cdot (\log p(t|c=1) - \log p(t|c=0)))
\end{aligned}$$

#	Term	Translation
1	`\n`	`\n`
2	Jedním	One of
3	Dalším	Another
4	Tento	This
5	Mezitím	Meanwhile
6	aspe	aspe
7	důležité	important$_{nom,sg,neut}$
8	důležitým	important$_{ins}$
9	Musíme	We must
10	Tyto	These
11	klíčových	key$^{adj}_{gen,pl}$
12	zlepšit	to improve
13	Bylo	It was
14	Každý	Each
15	zahrnuje	includes
16	odhodlání	determination$_{nom}$
17	vést	to lead
18	Jednoho	One^{adv}
19	klíčové	key$^{adj}_{nom,pl}$
20	zahrnovat	to include

(a) Terms contributing the most to positive classification.

#	Term	Translation
1	»	»
2	. . .	. . .
3	*	*
4	•	•
5	(	(
6	prý	supposedly [inf.]
7	ovšem	however
8	…	…
9	PRAHA	PRAGUE
10	procent	per cent
11	roku	year$_{gen}$
12	zhruba	rougly
13	loni	last year
14	-	-
15	!	!
16	včera	yesterday
17	PO	MON
18	/	/
19	asi	approximately [lit.]
20	korun	crowns$_{gen}$

(b) Terms contributing the most to negative classification.

Table 4.4 Terms contributing to classification in the NB Classifier with the RobeCzech classifier. `\n` (Table a, row 1) is the newline (`LF`) character. 'aspe' (Table a, row 6) is a sub-word token, likely occurring mostly in the word ***aspe**kt* (aspect).

In the second line, we reduced the fraction by the prior probability $P(c = 1) = P(c = 0)$, since the classes are balanced. As a result, every term t contributes to positive classification odds by

$$o(s_t) := \exp(s_t \cdot (\log p(t|c = 1) - \log p(t|c = 0)))$$

To get a number from the interval $[0, 1]$, we use the standard formula for converting odds to probabilities, yielding:

$$p_t := \frac{o(s_t)}{1 + o(s_t)}$$

Next, to visualise the contributions, we assign a colour to each token t in the sequence:

$$\text{colour}(p_t) = \begin{cases} \text{RGBA}(255, 50, 50, a(p_t)), & \text{if } p_t \geq 0.5 \\ \text{RGBA}(50, 50, 255, a(p_t)), & \text{if } p_t < 0.5 \\ \text{RGB}(200, 200, 200), & \text{if } t \notin \text{vocab} \end{cases}$$

where

$$a(p_t) = \sqrt[4]{2\,|p_t - 0.5|}$$

$a : [0, 1] \to [0, 1]$ is symmetrical around 0.5, with $a(0) = a(1) = 1$ and $a(0.5) = 0$. It is proportional to the absolute confidence of the model, regardless of the predicted class. Since most of the contributions are close to the midpoint 0.5, we propose a 4th root transformation that stretches the region near 0.5 closer to the extremes. We provide a visualisation of the functions above in Figure A.2 and an example of visualisation of contributions of tokens in a text sample in Figure 4.1.

Ne jed nala jsem nejlépe , jak jsem mohla , a za to se omlouvám . Měla jsem se
i později uji stit , že dané instrukce jsou stále platné . Měla jsem lépe zvážit své
chování , abych se uji stila , že nemám jít do klubu a zůstat doma ," vz kázala
finské veřejnosti premiér ka , která sama čeká na výsledky testů . Není jediná , která
v současnosti čelí nař čením z porušování pan dem i ckých pravidel . V Británii ve
středu rezign ovala mluvčí premiéra Borise John sona Al le gra Stra t ton ová . Na
uni klé nahrávce s dalšími úředníky kabinetu vtip kovala o večírku v Down ing Street
. Kon al se loni v době , kdy byly podobné akce zakázané .

Figure 4.1 Example contribution visualisation for a correctly classified human-written news article (with the model's output probability 0.48), using the NB classifier with the RobeCzech tokeniser. Red tokens contribute more to positive classification.

4.1.4 Conclusion and Discussion

In this section, we described the construction and training of a NB classifier with TF-IDF features. While the classifier achieved unexpectedly high, near-perfect train and validation accuracy, it struggled significantly with different domains (particularly with theses abstracts) and texts generated by a different family of LLMs. There was no significant bias towards non-native speakers.

For comparison, we constructed an analogous classifier that works with the RobeCzech tokeniser rather than a typical word tokeniser. While this significantly decreased the unknown token rate, the accuracy did not consistently improve across the datasets. Using this classifier, we observed a considerable increase in FPR between NatAdv and NonNativeC1, although this increase was not present between NatYouth and NonNative. Considering the small size of the prior pair of datasets, we argue that the evidence of bias against non-native speakers is inconclusive.

In the last part, we exploited the classifiers' straightforwardness to explain their functionality. From the original classifier, we observed that the terms contributing to positive/negative classification differ in their part of speech and tone. In the classifier modified by the RobeCzech tokeniser, the most contributing tokens partially overlapped with the prior, but, more importantly, showed that the paragraph structure, first words in a sentence, and punctuation are significant for classification.

4.2 RobeCzech Classifier

In this section, we describe our attempt to create a robust generated text classifier using contemporary approaches. As our base model, we chose RobeCzech (Straka et al., 2021; see Section 1.9.3), which falls into the pre-trained classifiers category (see Section 2.2.4). We first introduce its architecture and training approach in Sections 4.2.1 and 4.2.2. We present the results of the evaluation in Section 4.2.3 and inspect some unexpected behaviour with rare tokens in Section 4.2.4. We attempt to mitigate the behaviour and repeat the experiments in Sections 4.2.5 and 4.2.6. Finally, we attempt to gain some insight into the model's functionality using three sample-level explanation methods in Section 4.2.7.

4.2.1 Architecture

The architecture is based on the architecture for sentiment analysis described by Straka et al. (2021, Sec. 4.6). The prediction works as follows:

1. The input text is tokenised and special tokens are added, importantly, the `[CLS]` token in the beginning.
2. The tokens are passed through RobeCzech, and the output of the last hidden layer (i.e. the contextualised embeddings) is extracted.
3. The embedding of the `[CLS]` token is linearly projected to dimension 1. The rest of the embeddings are disregarded.
4. The linear projection is followed by a sigmoid activation, resulting in output $\in [0, 1]$, the probability of positive classification.

The architecture is illustrated in Figure 4.2.

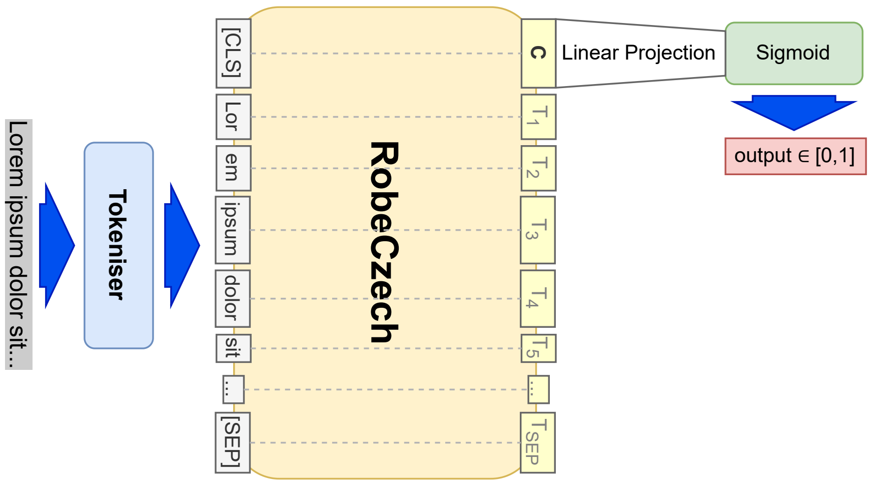


Figure 4.2 The architecture of the RobeCzech classifier.

4.2.2 Training

The classifier was trained on the SYNv9$^{\text{TRAIN}}$ dataset with no text normalisation. We used the SYNv9$^{\text{VAL}}$ dataset for hyperparameter tuning. The model was trained on a single NVIDIA RTX A4000 GPU (16 GB VRAM). We used the PyTorch (Paszke et al., 2019) implementation of the standard training procedures.

The training procedure consisted of three phases: in the first phase, we froze the RobeCzech parameters and only trained the linear projection layer (classification head) at a constant learning rate. In the second phase, we unfroze the RobeCzech weights and trained them with a linear learning rate warmup from 0 to a specified value. Finally, in the third phase, we kept the weights unfrozen and trained with cosine decay to 0.

The following hyperparameters were used in all the phases: batch size: 32, optimiser: AdamW (with the default $\beta_1 = 0.9$, $\beta_2 = 0.999$), weight decay: 10^{-3}, label smoothing: 0.1.

The classification head was trained by itself for one epoch at a learning rate 5×10^{-4}. We were able to reach accuracy on SYNv9$^{\text{VAL}}$ of 89.80% after this epoch alone. Next, the whole model was trained with a linear learning rate warmup from 0 to 3×10^{-7} over one epoch. Finally, the model was trained for an additional 3 epochs with cosine learning rate decay to 0.

4.2.3 Evaluation

First, we evaluated the model on the datasets with no text pre-processing. The results are shown in Table 4.5. For comparison, we evaluated the same model with PunctNorm applied to data on inference. The results with the normalisation applied are shown in Table 4.6.

Comparing Table 4.5 (resp. 4.6) with the results of the TF-IDF NB Classifier displayed in Table 4.1, we observe that the NB classifier (initially intended as a baseline classifier) outperforms the supposedly robust, more complex RobeCzech classifier on almost all datasets – except for SYNv9$^{\text{VAL}}_{\text{Llama}}$, Abs2020, and AbsNew.

Inspecting Table 4.6, we conclude that applying PunctNorm during inference consistently shifts the prediction towards the positive label (i.e. increases the FPR and decreases the FNR). This is most significant in NatYouth, where the FPR increased by 29.34%.

4.2.4 No-Break Space Anomaly

While we originally speculated that the large difference in the FPR was caused by the loss of information about paragraph structure or punctuation, upon closer inspection, we discovered that numerous texts in the dataset contain the *no-break space* (`U+00a0`) character (which was added during the transcription from handwritten texts). Simply replacing this character with a regular space leads to the high FPR as observed before (specifically 57.56%).

Paradoxically, the *no-break space* did not occur in SYNv9$^{\text{TRAIN}}$, which was used to train the model. In fact, the character breaks into two tokens upon tokenisation – utilising the *byte fallback* – suggesting that the character was infrequent or non-existent in the pre-training data as well. Therefore, the model likely developed some generalised rules, possibly about rare tokens.

We tested this hypothesis by introducing RandAug to the data on inference, expecting the predictions to be shifted towards the positive label – i.e. an increase

[5] The metrics suppose that all the abstracts are human-written, which is not guaranteed.

Dataset	Acc	FPR	FNR	Train Prep.	Infer. Prep.
SYNv9$^{\text{TRAIN}}$	99.35%	0.42%	0.88%	–	
SYNv9$^{\text{VAL}}$	99.33%	0.33%	1.00%		
SYNv9$^{\text{VAL}}_{\text{4oMini}}$	99.65%	–	0.35%		
SYNv9$^{\text{VAL}}_{\text{Llama}}$	92.69%	–	7.31%		
Wiki	86.73%	15.17%	11.37%		
News	85.96%	27.82%	0.26%		–
NonNative	52.44%	47.56%	–		
NonNativeC1	10.34%	89.66%	–		
NatYouth	71.56%	28.44%	–		
NatAdv	37.93%	62.07%	–		
Abs2020	33.78%	66.22%	–		
AbsNew[5]	32.44%	67.56%	–		

Table 4.5 Evaluation results of the RobeCzech Classifier.

Dataset	Acc	FPR	FNR	Train Prep.	Infer. Prep.
SYNv9$^{\text{TRAIN}}$	99.01%	1.17%	0.80%	–	
SYNv9$^{\text{VAL}}$	98.83%	1.67%	0.67%		
SYNv9$^{\text{VAL}}_{\text{4oMini}}$	99.65%	–	0.35%		
SYNv9$^{\text{VAL}}_{\text{Llama}}$	95.68%	–	4.32%		
Wiki	83.41%	23.70%	9.48%		
News	80.31%	39.11%	0.26%		PunctNorm
NonNative	50.00%	50.00%	–		
NonNativeC1	3.45%	96.55%	–		
NatYouth	42.22%	57.78%	–		
NatAdv	13.79%	86.21%	–		
Abs2020	33.72%	66.28%	–		
AbsNew	32.29%	67.71%	–		

Table 4.6 Evaluation results of the RobeCzech Classifier with PunctNorm preprocessing on inference.

Dataset	Acc	FPR	FNR	Train Prep.	Infer. Prep.
SYNv9$^{\text{TRAIN}}$	99.14%	0.17%	1.55%	–	
SYNv9$^{\text{VAL}}$	99.33%	0.00%	1.34%		
SYNv9$^{\text{VAL}}_{\text{4oMini}}$	99.65%	–	0.35%		
SYNv9$^{\text{VAL}}_{\text{Llama}}$	88.7%	–	11.30%		
Wiki	85.55%	9.00%	19.91%		
News	90.42%	18.37%	0.79%		RandAug
NonNative	76.00%	24.00%	–		
NonNativeC1	34.48%	65.52%	–		
NatYouth	72.67%	27.33%	–		
NatAdv	44.83%	55.17%	–		
Abs2020	69.61%	30.39%	–		
AbsNew	66.39%	33.61%	–		

Table 4.7 Evaluation results of the RobeCzech Classifier with RandAug preprocessing on inference only.

in FNR and a decrease in FPR. The results were consistent with our hypothesis, as demonstrated in Table 4.7.

4.2.5 Training with Augmentation

In an attempt to mitigate the model classifying based on token frequency, likely observed in the pre-training phase, we trained another classifier with RandAug during training rather than just on inference.

Compared to Section 4.2.2, we changed the learning rate in the first (classification head) phase to 10^{-3}. This was followed by the second (warmup) phase with learning-rate linearly decreasing from 0 to 5×10^{-7} over one epoch. Finally, in the third (cosine decay) phase, we trained for only one additional epoch, with learning rate decay to 0. Interestingly, a validation accuracy of 97.99% was achieved just by training the classification head in the first phase.

4.2.6 Evaluation After Training with Augmentation

We started the evaluation with the same pre-processing as during training, i.e. RandAug. The results are shown in Table 4.8. For comparison, we also evaluated the model with PunctNorm on inference, as shown in Table 4.9.

Dataset	Acc	FPR	FNR	Train Prep.	Infer. Prep.
$\text{SYNv9}^{\text{TRAIN}}$	99.16%	0.55%	1.13%	RandAug	RandAug
$\text{SYNv9}^{\text{VAL}}$	99.00%	1.00%	1.00%		
$\text{SYNv9}^{\text{VAL}}_{\text{4oMini}}$	99.65%	–	0.35%		
$\text{SYNv9}^{\text{VAL}}_{\text{Llama}}$	90.03%	–	9.97%		
Wiki	85.78%	13.74%	14.69%		
News	91.86%	16.01%	0.26%		
NonNative	67.11%	32.89%	–		
NonNativeC1	27.59%	72.41%	–		
NatYouth	62.22%	37.78%	–		
NatAdv	48.28%	51.72%	–		
Abs2020	67.01%	32.99%	–		
AbsNew	64.88%	35.12%	–		

Table 4.8 Evaluation results of the RobeCzech Classifier trained with RandAug, with RandAug also applied on inference.

While training with RandAug led to significant improvement compared to previous attempts, the classifier still associated rare tokens with negative classification. This is visible when comparing Tables 4.8 and 4.9. When the RandAug noise is no longer present on inference, the classifier shifts towards positive classification, although this shift is less dramatic than before.

4.2.7 Explaining the Classifier

Unlike the NB classifier, which is straightforward to explain (as demonstrated in Section 4.1.3), the complex architecture of the RobeCzech classifier poses

Dataset	Acc	FPR	FNR	Train Prep.	Infer. Prep.
SYNv9$^{\text{TRAIN}}$	99.03%	0.84%	1.09%	RandAug	PunctNorm
SYNv9$^{\text{VAL}}$	99.00%	1.00%	1.00%		
SYNv9$^{\text{VAL}}_{\text{4oMini}}$	99.65%	–	0.35%		
SYNv9$^{\text{VAL}}_{\text{Llama}}$	93.02%	–	6.98%		
Wiki	84.60%	18.48%	12.32%		
News	88.85%	22.05%	0.26%		
NonNative	54.00%	46.00%	–		
NonNativeC1	17.24%	82.76%	–		
NatYouth	41.56%	58.44%	–		
NatAdv	17.24%	82.76%	–		
Abs2020	51.42%	48.58%	–		
AbsNew	48.49%	51.51%	–		

Table 4.9 Evaluation results of the RobeCzech Classifier trained with RandAug, with PunctNorm applied on inference.

challenges. In this section, we attempt to explain the classifier using various methods. All of the presented results in the figures were constructed from the variant with no pre-processing on training and PunctNorm pre-processing on inference.

Selective Masking Given the pre-training approach (see Section 1.9.1), the model recognises a special `[MASK]` token, which can replace any tokens in the sequence. We exploit it for input perturbation, to hide certain tokens from the classifier and compare the results.

Given a sequence of tokens $(t_1, \ldots, t_n)$, the model normally predicts $p(c = 1 \mid (t_1, \ldots, t_n))$. By masking the i-th token, we aimed to get the probability $p(c = 1 \mid (t_1, \ldots, t_{i-1}, t_{i+1}, t_n))$. Furthermore, since some words are broken into multiple tokens, we also implemented masking of k consequent tokens, aiming to calculate $p(c = 1 \mid (t_1, \ldots, t_{i-1}, t_{i+k}, t_n))$. For a given k, we mask all consequent subsequences of length k, one subsequence at a time, and observe how the prediction probability changes for every masking.

The interpretation of the masked result is to some extent counterintuitive: consider the original baseline output with no masking b and the output after masking the i-th token $m_i = p(c = 1 \mid (t_1, \ldots, t_{i-1}, t_{i+1}, t_n))$. If $m_i < b$, the probability of positive classification decreased after the masking of the i-th token, which means that the token originally contributed to positive classification.

To create a more intuitive visualisation, we propose the following colouring with red/blue inversion: first, we consider the relative change $m_i - b$ rather than m_i. We scale the differences to $[-1, 1]$ using:

$$M_i = \begin{cases} \frac{m_i - b}{-\min_i(m_i - bi)}, & \text{if } m_i < b \\ \frac{m_i - b}{\max_i(m_i - bi)}, & \text{if } m_i > b \end{cases}$$

and use M_i for colouring:

$$\text{colour}(m_i) = \begin{cases} \text{RGBA}(255, 50, 50, M_i), & \text{if } M_i < 0 \\ \text{RGBA}(50, 50, 255, M_i), & \text{if } M_i > 0 \end{cases}$$

Figure 4.3 shows an example visualisation of the masking. Upon experimenting with various texts and masked window lengths, we were unable to find any conclusive repeated patterns in the visualisations. Therefore, we consider this attempt at model explanation unsuccessful.

[CLS] Ne jed nala jsem nejlépe , jak jsem mohla , a za to se omlouvám . Měla jsem se i později uji stit , že dané instrukce jsou stále platné . Měla jsem lépe zvážit své chování , abych se uji stila , že nemám jít do klubu a zůstat doma ," vz kázala finské veřejnosti premiér ka , která sama čeká na výsledky testů . Není jediná , která v současnosti čelí nař čením z porušování pan dem i ckých pravidel . [SEP]
-0.052 0.031

Figure 4.3 Example result of selective token masking in the RobeCzech classifier with masking window length of 1. The baseline probability was $b = 0.484$, and the true class was 0 (human-written). Red tokens contribute more to positive classification. The two boxes on the last line show the maximum negative/positive change w.r.t. b.

Integrated Gradients As an alternative method of explaining the outputs of the model, we experimented with Integrated Gradients (Sundararajan et al., 2017; see Section 2.4.1), implemented in the Captum library (Kokhlikyan et al., 2020). In our approach, we followed the instructions from Ruben Winastwan[6]. As the input layer for gradient integration, we used the *embeddings* layer of the RobeCzech model and computed the mean contribution across the embedding dimension for each token. As the baseline input, we used a sequence of the `[PAD]` token with the same length as the actual input tokens, with special tokens added to the beginning and the end:

$$(\texttt{[CLS]}, \texttt{[PAD]}, \ldots, \texttt{[PAD]}, \texttt{[SEP]})$$

For colouring, we used a similar approach as above, with $b = 0$ and no blue/red colour inversion. Figure 4.4 displays an example result of the explanations. Again, the visualisation did not bring satisfactory results: the token contributing to positive classification by far the most was supposedly a full stop. The tokens supposedly contributing the most to negative classification were seemingly random prepositions and a comma.

LIME As the final attempt, we utilised the text module from the Local Interpretable Model-agnostic Explanations (LIME) approach (Ribeiro et al., 2016; see Section 2.4.2). The `mask_string` parameter was set to `[MASK]`, which gets converted to the corresponding special token upon tokenisation. We used the same colouring approach as with IG.

[6] `https://towardsdatascience.com/interpreting-the-prediction-of-bert-model-for-text-classification-5ab09f8ef074/`

[CLS] Ne jed nala jsem nejlépe , jak jsem mohla , a za to se omlouvám . Měla jsem se i později uji stit , že dané instrukce jsou stále platné . Měla jsem lépe zvážit své chování , abych se uji stila , že nemám jít do klubu a zůstat doma ," vz kázala finské veřejnosti premiér ka , která sama čeká na výsledky testů . Není jediná , která v současnosti čelí nař čením z porušování pan dem i ckých pravidel . V Británii ve středu rezign ovala mluvčí premiéra Borise John sona Al le gra Stra t ton ová . Na uni klé nahrávce s dalšími úředníky kabinetu vtip kovala o večírku v Down ing Street . Kon al se loni v době , kdy byly podobné akce zakázané . [SEP]
0.418 -0.344

Figure 4.4 Example result of the Integrated Gradients explanation method applied on the RobeCzech classifier. The true label was 0 (human-written), and the model's output was 0.48. Red tokens contribute more to positive classification. The two boxes on the last line show the maximum negative/positive contributions.

Nejednala jsem nejlépe jak jsem mohla a za to se omlouvám Měla jsem se i později ujistit že dané instrukce jsou stále platné Měla jsem lépe zvážit své chování abych se ujistila že nemám jít do klubu a zůstat doma vzkázala finské veřejnosti premiérka která sama čeká na výsledky testů Není jediná která v současnosti čelí nařčením z porušování pandemických pravidel V Británii ve středu rezignovala mluvčí premiéra Borise Johnsona Allegra Strattonová Na uniklé nahrávce s dalšími úředníky kabinetu vtipkovala o večírku v Downing Street Konal se loni v době kdy byly podobné akce zakázané
0.023 -0.015

Figure 4.5 Example result of the LIME explanation method applied on the RobeCzech classifier. The true label was 0 (human-written), and the model's output was 0.48. Red tokens contribute more to positive classification. The two boxes on the last line show the maximum negative/positive contributions.

A visualisation of an example document is shown in Figure 4.5. Compared to previous attempts, we consider this method to be the most informative. From the figure, we can see that the model largely works with the initial words in sentences, and the token contributing the most to negative classification was *'loni'* ('last year'). This is similar to our findings in Section 4.1.3, in which we explained the NB classifier with the RobeCzech tokeniser. Note that in Table 4.4b, the term *'loni'* was also among the terms contributing the most to negative classification. This suggests some extent of similarity between the two classifiers. We further examine this similarity in Section 4.6.

4.2.8 Conclusion and Discussion

In this section, we introduced a classifier from the pre-trained models family, specifically based on the RobeCzech classifier. We proposed the architecture and the fine-tuning approach. Upon evaluation, we discovered that, similarly to the classifier from Section 4.1, our classifier achieved a near-perfect accuracy on the training domains, but struggled considerably with other domains. Moreover, it did not consistently outperform the aforementioned baseline classifier. While training on more domains would likely achieve more robustness, it would not necessarily guarantee good performance on previously unseen domains.

Using various evaluation scenarios, we found that the model likely associates infrequent characters with human-written texts, which was difficult to 'unlearn' even by introducing random rare characters to all training data. This was particularly significant when the model achieved a high accuracy on the NatYouth corpus, which we later found to be caused by the no-break space character. Disregarding that result, the models performed better on the NonNative dataset compared to NatYouth. In contrast, all the variants performed better or equally good on NatAdv compared to NonNativeC1. Therefore, we do not conclude any systematic bias against non-native speakers.

In the last part, we attempted to utilise certain sample-level explanation frameworks on the model. While we were unable to obtain conclusive observations from the selective masking and the Integrated Gradients methods, the LIME method provided more promising results. The results suggested that the first words of sentences have great importance for the classification. This finding, together with some contributing keywords, is comparable to the findings in Section 4.1.3 (particularly when using the RobeCzech tokeniser), suggesting some similarity between the NB classifier and the RobeCzech classifier. We further examine this potential similarity in Section 4.6. As for the unsuccessful explanation attempts, we believe that the model also uses more complex structures in the text, which are not possible to visualise on a token level.

4.3 Transformer Encoder Classifier from Scratch

In this chapter, we propose a non-conventional approach to Transformer models (Vaswani et al., 2017): custom Transformer encoder-only (see Section 1.9) classifier (CTEC) from scratch – i.e. with no pre-training [7]. The objective was also to create a model with a smaller number of parameters, since we do not require it to have all the complex capabilities that pre-trained BERT (Devlin et al., 2019) models have.

This section tries to inspect the similarities and differences of the CTEC classifier and RobeCzech, and is structured similarly to Section 4.2. First, we introduce the architecture and changes in the training procedure in Section 4.3.1 and 4.3.2. We evaluate the classifier in the same environment as RobeCzech in Section 4.3.3. Finally, we apply the same explanation methods as before to the CTEC in Section 4.3.4.

4.3.1 Architecture

The outline of the architecture was the same as described in Section 4.2.1, with the Tokeniser and the RobeCzech components replaced with components trained from scratch:

- `ByteLevelBPETokenizer` from Huggingface's Tokenizers[8] library was used for the tokeniser. As the name suggests, this is a byte-level BPE tokeniser (see Section 1.6). The tokeniser was configured to a vocabulary size of 16 000 and a minimum frequency of 2.

- For the backbone model, we constructed a RoBERTa model (see Section 1.9.2) using the `RobertaConfig` from Huggingface's Transformers library (Wolf et al., 2020). The specific configuration is displayed in Table 4.10, with the respective values of the RobeCzech model for comparison.

4.3.2 Training

The training procedure from Section 4.2.2 was modified for the needs of the randomly initialised model. Firstly, the first phase – in which the RobeCzech model's weights were frozen and only the classification head was trained – was omitted. Secondly, the gradient norm was clipped to 1.0 to ensure stability even with large errors.

Following previous experiments, two classifiers were trained on SYNv9$^{\text{TRAIN}}$ – one with no text pre-processing and one with RandAug during training – both trained with the same hyperparameters. Similarly, SYNv9$^{\text{VAL}}$ was used to tune the hyperparameters. The batch size was increased to 128, while the remaining

[7]The method was originally implemented by accident: due to a mistake in the code, we loaded the RobeCzech (see Section 1.9.3) model without the pre-trained weights. Therefore, only the pre-trained tokeniser and the model architecture were loaded. Despite this, after tuning the hyperparameters, we were able to achieve comparable validation accuracy.

[8]`https://huggingface.co/docs/tokenizers/index`

[9]Including the classification head.

Parameter	Value (CTEC)	Value (RobeCzech)
`hidden_size`	360	768
`num_attention_heads`	6	12
`intermediate_size`	1024	3072
`num_hidden_layers`	6	12
`vocab_size`	16 000	51 997
`layer_norm_eps`	10^{-5}	10^{-5}
`max_position_embeddings`	514	514
Total Parameters[9]	13 506 505	125 976 577

Table 4.10 Overview of the configuration of the CTEC and comparison with the RobeCzech model.

parameters, other than the learning rate, were preserved – AdamW optimiser (with the default $\beta_1 = 0.9$, $\beta_2 = 0.999$), weight decay: 10^{-3}, label smoothing: 0.1.

The whole model was trained with linear LR warmup from 0 to 10^{-4} for 10 epochs and additional 23 epochs with cosine LR decay to 0.

4.3.3 Evaluation

To reproduce the previous experiments, the classifier trained without pre-processing was evaluated using no pre-processing (Table 4.11), PunctNorm (Table 4.12), and RandAug (Table 4.13). The classifier trained with RandAug was evaluated using RandAug (Table 4.14) and PunctNorm (Table 4.15).

The results in Table 4.11 are quite satisfactory, with two considerable drawbacks: 47.33% FPR on the NatYouth dataset and 67.98% FPR in the News dataset. Compared to the results with PunctNorm on inference (Table 4.12), we observe that the model is still sensitive to specific punctuation and whitespace, although normalising the text had the opposite effect on the CTEC than on the RobeCzech classifier: while normalising the whitespace and punctuation shifts the prediction towards the positive label in Robeczech, the CTEC prediction shifts towards the negative label.

The results of evaluation with RandAug noise (Table 4.13) do not show a systematic shift towards either of the labels. Its effects were rather inconsistent: on some datasets, the performance generally deteriorated, while on others it improved. This behaviour contrasts with the RobeCzech model's behaviour, where the new RandAug shifted the prediction towards the positive label. We hypothesise that the CTEC does not exploit the prior frequencies of the tokens because it did not learn the frequencies during pre-training, unlike RobeCzech.

Training with RandAug, we observed more balance between FPR and FNR. Using RandAug (Table 4.14) or PunctNorm (Table 4.15) on inference made little difference, with only a slight shift towards the negative label when using PunctNorm. This suggests that the classifier is, to some extent, robust against random noise.

As the final comparison, we return to the first CTEC trained with no pre-processing and test it with PunctNorm, while decreasing the positive label

[10]The metrics suppose that all the abstracts are human-written, which is not guaranteed.

Dataset	Acc	FPR	FNR	Train Prep.	Infer. Prep.
SYNv9$^{\text{TRAIN}}$	100.00%	0.00%	0.00%	–	–
SYNv9$^{\text{VAL}}$	99.16%	0.67%	1.00%		
SYNv9$^{\text{VAL}}_{\text{4oMini}}$	100%	–	0.00%		
SYNv9$^{\text{VAL}}_{\text{Llama}}$	84.72%	–	15.28%		
Wiki	91.0%	18.01%	0.00%		
News	66.01%	67.98%	0.00%		
NonNative	95.33%	4.67%	–		
NonNativeC1	89.66%	10.34%	–		
NatYouth	52.67%	47.33%	–		
NatAdv	72.41%	27.59%	–		
Abs2020	94.50%	5.50%	–		
AbsNew[10]	93.46%	6.54%	–		

Table 4.11 Evaluation results of the CTEC.

Dataset	Acc	FPR	FNR	Train Prep.	Infer. Prep.
SYNv9$^{\text{TRAIN}}$	98.49%	0.00%	3.02%	–	PunctNorm
SYNv9$^{\text{VAL}}$	94.98%	0.00%	10.03%		
SYNv9$^{\text{VAL}}_{\text{4oMini}}$	96.86%	–	3.14%		
SYNv9$^{\text{VAL}}_{\text{Llama}}$	44.52%	–	55.48%		
Wiki	69.19%	0.00%	61.61%		
News	93.31%	0.00%	13.39%		
NonNative	98.67%	1.33%	–		
NonNativeC1	96.55%	3.45%	–		
NatYouth	99.33%	0.67%	–		
NatAdv	96.55%	3.45%	–		
Abs2020	94.56%	5.44%	–		
AbsNew	93.41%	6.59%	–		

Table 4.12 Evaluation results of the CTEC with PunctNorm pre-processing on inference.

Dataset	Acc	FPR	FNR	Train Prep.	Infer. Prep.
SYNv9$^{\text{TRAIN}}$	99.16%	1.05%	0.63%	–	RandAug
SYNv9$^{\text{VAL}}$	97.49%	3.01%	2.01%		
SYNv9$^{\text{VAL}}_{\text{4oMini}}$	100.00%	–	0.00%		
SYNv9$^{\text{VAL}}_{\text{Llama}}$	72.76%	–	27.24%		
Wiki	85.31%	9.95%	19.43%		
News	96.85%	4.72%	1.57%		
NonNative	71.11%	28.89%	–		
NonNativeC1	79.31%	20.69%	–		
NatYouth	81.11%	18.89%	–		
NatAdv	75.86%	24.14%	–		
Abs2020	62.11%	37.89%	–		
AbsNew	61.90%	38.10%	–		

Table 4.13 Evaluation results of the CTEC with RandAug pre-processing on inference.

threshold. Table 4.16 shows that by applying this change, we achieved results very similar to the CTEC trained with RandAug and tested with PunctNorm (Table 4.15). The exact value of the threshold was chosen to 0.37 to achieve the most similar results. This comparison demonstrates that the two training approaches do not lead to fundamentally different classifiers, and their differences can be largely mitigated simply by changing the classification threshold.

4.3.4 Explaining the Model

Given the similar architecture to the RobeCzech model, we opted for the same explanation techniques as in Section 4.2.7: selective masking, Integrated Gradients, and LIME. The experiments were done on the classifier trained with RandAug, with PunctNorm applied on inference. The example visualisations are shown in Figures A.3, A.4, and A.5 in the attachment. In the case of CTEC, none of the methods provided satisfactory results. A common occurrence was that the explainers considered conjunctions and punctuation to be important for classification. We were unable to identify any further patterns.

4.3.5 Conclusion and Discussion

In this section, we introduced an unconventional approach to training a Transformer model that does not involve pre-training. We were able to create a detector with almost 10 times fewer parameters than the RobeCzech classifier and achieve comparable results on the training domain and a stronger robustness. For the first time, the detector's performance did not drop below the random baseline on any of the datasets. During the evaluation, we observed that the model partially relies on the punctuation, which can lead to incorrect classification. However, unlike the RobeCzech model, our CTEC did not make any assumptions about previously unseen tokens.

The good-performing variants of the classifier had somewhat ($\leq 12\%$) better performance on the NatYouth dataset compared to NonNative, and the difference was less significant or non-existent when comparing NonNativeC1 to NatAdv. Therefore, some suspicions can be raised about the bias against non-native speakers in this classifier, although its extent is not nearly as drastic as described by Liang et al. (2023).

Lastly, we again attempted to explain the model. We used the same three methods used for the RobeCzech classifier: selective masking, Integrated Gradients, and LIME. We were unable to draw any non-trivial conclusions from the explanations provided by either of the explored methods.

Dataset	Acc	FPR	FNR	Train Prep.	Infer. Prep.
SYNv9$^{\text{TRAIN}}$	99.96%	0.08%	0.00%	RandAug	
SYNv9$^{\text{VAL}}$	99.16%	0.33%	1.34%		
SYNv9$^{\text{VAL}}_{\text{4oMini}}$	100.00%	–	0.00%		
SYNv9$^{\text{VAL}}_{\text{Llama}}$	75.75%	–	24.25%		
Wiki	87.68%	6.16%	18.48%		
News	99.21%	1.57%	0.00%		RandAug
NonNative	67.78%	32.22%	–		
NonNativeC1	79.31%	20.69%	–		
NatYouth	79.78%	20.22%	–		
NatAdv	79.31%	20.69%	–		
Abs2020	62.11%	37.89%	–		
AbsNew	61.07%	38.93%	–		

Table 4.14 Evaluation results of the CTEC with RandAug pre-processing during training and inference.

Dataset	Acc	FPR	FNR	Train Prep.	Infer. Prep.
SYNv9$^{\text{TRAIN}}$	99.96%	0.08%	0.00%	RandAug	
SYNv9$^{\text{VAL}}$	99.50%	0.00%	1.00%		
SYNv9$^{\text{VAL}}_{\text{4oMini}}$	100.00%	–	0.00%		
SYNv9$^{\text{VAL}}_{\text{Llama}}$	72.76%	–	27.24%		
Wiki	87.91%	4.74%	19.43%		
News	99.21%	1.57%	0.00%		PunctNorm
NonNative	71.33%	28.67%	–		
NonNativeC1	82.76%	17.24%	–		
NatYouth	82.22%	17.78%	–		
NatAdv	82.76%	17.24%	–		
Abs2020	62.42%	37.58%	–		
AbsNew	62.68%	37.32%	–		

Table 4.15 Evaluation results of the CTEC with RandAug pre-processing during training and PunctNorm on inference.

Dataset	Acc	FPR	FNR	Train Prep.	Infer. Prep.
SYNv9$^{\text{TRAIN}}$	99.69%	0.00%	0.63%	–	
SYNv9$^{\text{VAL}}$	99.16%	0.33%	1.34%		
SYNv9$^{\text{VAL}}_{\text{4oMini}}$	100.00%	–	0.00%		
SYNv9$^{\text{VAL}}_{\text{Llama}}$	70.43%	–	29.57%		
Wiki	86.26%	8.06%	19.43%		
News	98.69%	1.57%	1.05%		PunctNorm
NonNative	82.44%	17.56%	–		
NonNativeC1	82.76%	17.24%	–		
NatYouth	87.67%	11.33%	–		
NatAdv	89.66%	10.34%	–		
Abs2020	62.66%	37.34%	–		
AbsNew	60.78%	39.22%	–		
Positive threshold decreased to 0.37					

Table 4.16 Evaluation results of the CTEC with no pre-processing during training and PunctNorm on inference; positive threshold decreased to 0.37.

4.4 Commercial Generated Text Detector

To provide more realistic detection results, we included a commercial, closed-source model for comparison. At the time of writing this, there is no monolingual detector for AI-generated text in Czech. Instead, we resorted to multilingual detectors. After an informal survey of the available options, we found that *Plagramme*[11] performs well on the tested documents. The company generously provided access to their API free of charge for academic research[12].

The specific inner mechanisms of Plagramme's AI detector are proprietary. The product webpage states the following:

> *A set of tools helps develop and provide the AI text checking service. The AI detector uses machine learning, natural language processing, web development tools, and cloud services to ensure accurate checking and reliable detection of AI-generated content.*

The tool works on a *sentence* level, returning the probability p_i of each of the S sentences in a document being generated. To obtain the same format as our previous detectors, we compute the average of the probabilities (each sentence with the same weight, regardless of its length):

$$p = \frac{1}{S} \sum_{i=1}^{S} p_i$$

In order not to overload Plagramme's servers, we limited the datasets larger than 100 documents to a randomly selected subset of size 100. The documents were clipped to the first 512 words, similarly to earlier experiments. SpaceNorm normalisation was then applied. The results are shown in Table 4.17.

Dataset	Acc	FPR	FNR	Pre-Processing
SYNv9$^{\text{VAL}}$	97.5.%	1.00%	4.00%	
SYNv9$^{\text{VAL}}_{\text{4oMini}}$	99.00%	–	1.00%	
SYNv9$^{\text{VAL}}_{\text{Llama}}$	65.00%	–	35.00%	
Wiki	93.00%	2.00%	12.00%	
News	96.50%	6.00%	1.00%	
NonNative	98.00%	2.00%	–	SpaceNorm
NonNativeC1	100.00%	0.00%	–	
NatYouth	99.00%	1.00%	–	
NatAdv	96.55%	3.45%	–	
Abs2020	96.00%	4.00%	–	
AbsNew[13]	89.00%	11.00%	–	

Table 4.17 Evaluation results of the commercial classifier with at most 100 samples per dataset.

[11] `https://www.plagramme.com/services/ai`

[12] We hereby declare that the choice of the tool was *not* influenced by this offer and that every communication with the company happened *after* it was decided to use this tool.

[13] The metrics suppose that all the abstracts are human-written, which is not guaranteed.

The results show considerably better results than the classifiers that we created and presented before. The detector struggled the most on the SYNv9$^{\text{VAL}}_{\text{LLAMA}}$ dataset, suggesting that it was not trained on Llama-generated (Grattafiori et al., 2024) documents and does not generalise well across the models. Disregarding this dataset, the only two datasets on which the error rate was higher than 10% were Wiki (where the FNR was 12%) and AbsNew (where the FPR was 11%). Given the considerably smaller FPR on Abs2020, this might suggest that some of the new abstracts were in fact generated.

The difference in performance between NonNative and NatYouth was negligible. Despite that, the difference between NonNativeC1 and NatAdv was greater (1.00% vs. 3.45%), but given the small sample size, we do not consider this difference significant.

4.4.1 Conclusion and Discussion

In this section, we introduced and tested a commercial multilingual generated text detector, which we believe will provide a more realistic insight into real-world applications. We observed that the commercial detector consistently outperforms our detectors except for correctly classifying Llama-generated text, which it was likely not trained on.

We did not conclude any systematic bias against non-native speakers, with the error being $< 3.5\%$ on each native/non-native dataset. Given the near-perfect performance on most datasets, the outliers may provide some information about the datasets themselves, rather than the classifier. We suspect that an increased 'FPR' on AbsNew might suggest that some of the documents in the dataset are generated (hence the quotation marks). Furthermore, the increased FNR on Wiki could mean that generating natural-sounding Wikipadia-style articles is a simple task for LLMs, as they likely observed the reference article in the training data (this will be further demonstrated in Section 4.5)

4.5 Llama Logit-Based Analyser

In this section, we introduce our approaches for analysing the entropy of a text with respect to a decoder-only Transformer model (see Section 1.10). The analyser proposed in this section plays a key role when comparing our findings to the findings of Liang et al. (2023), who suggest that the texts created by non-native speakers have lower perplexities and that the generated text detectors rely on the perplexity. The latter will be discussed in Section 4.6.

The section is structured as follows: first, we introduce the concepts of entropy and perplexity of a document w.r.t. an LLM in Section 4.5.1. We discuss how we applied the concept to our specific case and what results we obtained in Section 4.5.2. Finally, we introduce an approach for visualising the tokens' attribution to entropy in Section 4.5.3 and use it to explain the entropy of a text from a non-native speaker.

4.5.1 Entropy and Perplexity w.r.t. a Model

For our purposes, we define[14] the entropy of a document $\mathbf{d}$ of length N w.r.t. a language model as:

$$H(\mathbf{d}; \text{model}) = -\frac{1}{N}\sum_{i=1}^{N} \log p_{\text{model}}(d_i \mid d_1, \ldots, d_{i-1})$$

This can be interpreted as the *cross-entropy* $H(q, p_{\text{model}})$ where q is a uniform distribution that assigns the probability $1/N$ to each token. Another interpretation is that the formula is the NLL of the document under the model, normalised by its length. The entropy is inversely related to the likelihood of a sequence being generated by the model: the larger the entropy, the lower the probability of generation. We illustrate how the $\log p_{\text{model}}(d_i \mid d_1, \ldots, d_{i-1})$ is extracted from a model in Figure 4.6.

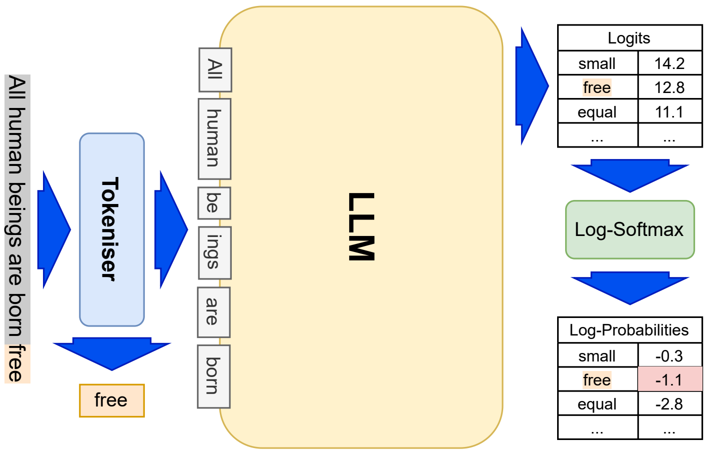


Figure 4.6 Example extraction of the log-probability of a token (‘`free`’) given a context. Note that in practice, the log-probabilities can be calculated simultaneously for each token, thanks to attention masking.

[14]Our definition stems from the established notion of perplexity of a token sequence (as used by Jiang et al., 2024), omitting the exponentiation.

The entropy $H(\mathbf{d}; \text{model})$ and the perplexity $PPL(\mathbf{d}; \text{model})$ of a document have a straightforward exponential relationship:

$$PPL(\mathbf{d}; \text{model}) = \exp(H(\mathbf{d}; \text{model}))$$

Therefore, the two properties are positively correlated. We chose to work with the entropy rather than the perplexity, as its distribution is simpler to model (see Figure 4.7).

4.5.2 Entropy Analysis

For our reference model, we chose Llama 3.2 1B base (Grattafiori et al., 2024). Unlike other authors (Jiang et al., 2024), who used GPT-2 (Radford; Jeffrey Wu et al., 2019b) as the reference model, we chose the Llama family, as we believe that a contemporary model represents the current GPT models more accurately than the somewhat outdated GPT-2, despite belonging to a different family. We worked with 16-bit weights and used the Unsloth library (Daniel Han; Unsloth team, 2023) for efficient inference. The small model was chosen for efficiency, and the results below suggest that it was sufficient. The number of samples for each dataset was clipped to 1000, and the subset was selected at random.

We truncated each sample to 512 tokens and disregarded the predictions on the first 100 tokens to provide sufficient context. Let $M = \min(N, 512)$, then our modified entropy formula is as follows:

$$H(\mathbf{d}; \text{model}) = -\frac{1}{M - 100} \sum_{i=101}^{M} \log P_{\text{model}}(d_i \mid d_1, \ldots, d_{i-1})$$

Samples where $M - 100 \leq 0$ were discarded.

First, we plotted the histogram of the entropy distribution for SYNv9$^{\text{TRAIN}}$, displayed in Figure 4.7. As expected, the generated documents have smaller entropies than the natural ones in most cases, although there are some overlaps in the histograms. Furthermore, we observed that the histograms resemble the normal distribution density. Upon this observation, we estimated their parameters μ_{GPT4o}, σ_{GPT4o}, μ_{NAT}, and σ_{NAT} using the maximum likelihood estimate, and plotted their respective density functions ($\varphi_{\mu_{\text{GPT4o}},\sigma_{\text{GPT4o}}}$ and $\varphi_{\mu_{\text{NAT}},\sigma_{\text{NAT}}}$) into the figure.

This density estimate $h = H(\mathbf{d}; model)$ could further be used for classification: let

$$p(h \mid c = 1) = \varphi_{\mu_{\text{GPT4o}},\sigma_{\text{GPT4o}}}(h)$$

and similarly for $p(\mathrm{x} \mid c = 0)$. Bayes' law states that

$$p(c = 1 \mid h) \propto \varphi_{\mu_{\text{GPT4o}},\sigma_{\text{GPT4o}}}(h) P(c = 1)$$

Since the classes are balanced, the $P(c = 1)$ component can be discarded. The final prediction would then be calculated as follows:

$$p(c = 1 \mid \mathrm{x}) = \frac{\varphi_{\mu_{\text{GPT4o}},\sigma_{\text{GPT4o}}}(h)}{\varphi_{\mu_{\text{GPT4o}},\sigma_{\text{GPT4o}}}(h) + \varphi_{\mu_{\text{NAT}},\sigma_{\text{NAT}}}(h)}$$

However, we found that this classification method is not practically applicable, as the densities differ considerably among different domains (as demonstrated

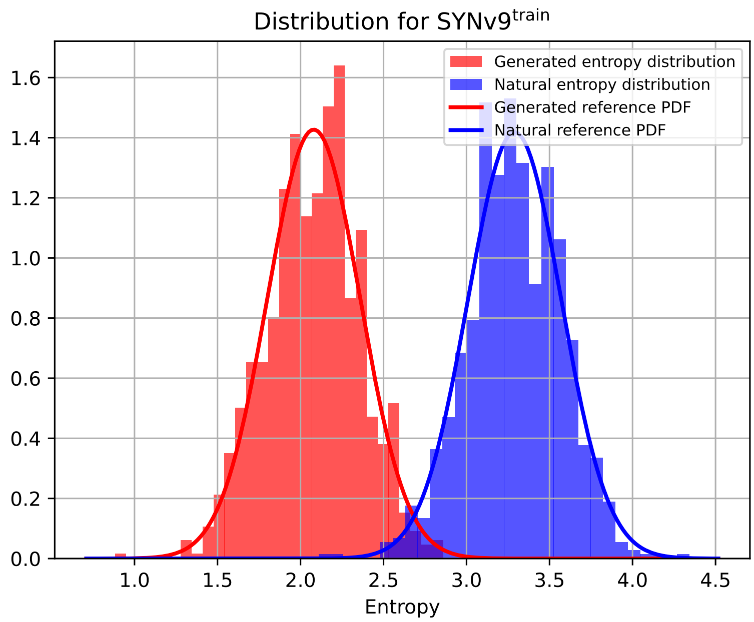


Figure 4.7 Entropy distribution for SYNv9$^{\text{TRAIN}}$ and their estimated densities.

below). Therefore, we do not present a classifier in this section but rather a tool for analysing the logit-based classifiers.

Figure 4.8 displays the entropy distributions of all the datasets, and Table 4.18 shows their respective mean and Standard Deviation (SD). In Figure 4.8, comparing subplots (i.a), (i.b), (i.c) with (ii.a), we see that the texts generated by GPT-4o (OpenAI; Hurst et al., 2024) and GPT-4o-mini[15] have similar entropy distributions, while the entropy for Llama 3.1 405B (Grattafiori et al., 2024) was smaller, on average. This is likely because the model against which we measured the entropy was also from the Llama 3 family.

Subplots (ii.b) and (ii.c) provide important insight for cross-domain analysis. While it still holds that, within one domain, generated texts have smaller entropies than natural texts, with little overlap, the mean and standard deviation of the entropies differ significantly across domains. In the case of Wiki and News, the entropies of both generated and natural data were significantly smaller than expected from SYNv9$^{\text{TRAIN}}$. This is likely because the two domains are publicly available and likely contained in the LLMs training data, which makes the models more likely to generate similar documents.

Importantly, contrary to the findings of Liang et al. (2023), the mean entropy of NonNative is *not* less than that of NatYouth. In fact, the opposite holds, although the difference is not dramatic. Similarly, the mean entropy of NonNativeC1 is slightly greater than the mean entropy of NatAdv. Interestingly, the advanced student texts (NonNativeC1 and NatAdv) have smaller mean entropies compared to their respective non-advanced counterparts. This is further examined in Section 4.5.3.

The abstract datasets (Abs2020, AbsNew) have the smallest mean entropies of all the negative datasets. This is likely because of the easily predictable, rigid

[15] https://openai.com/index/gpt-4o-mini-advancing-cost-efficient-intelligence

structure of the texts. Contrary to the results from Section 4.4, we did not find any significant differences in distributions between the two datasets displayed in Figure 4.8 (iv.b) and (iv.c).

Dataset	**Positive**		**Negative**	
	Mean	**SD**	**Mean**	**SD**
SYNv9$^{\text{TRAIN}}$	2.08	0.28	3.29	0.28
SYNv9$^{\text{VAL}}$	2.06	0.29	3.29	0.28
SYNv9$^{\text{VAL}}_{\text{4oMini}}$	2.05	0.25	–	–
SYNv9$^{\text{VAL}}_{\text{Llama}}$	1.90	0.37	–	–
Wiki	1.66	0.21	2.4	0.31
News	1.88	0.18	2.8	0.36
NonNative	–	–	3.45	0.58
NonNativeC1	–	–	2.95	0.39
NatYouth	–	–	3.15	0.50
NatAdv	–	–	2.84	0.20
Abs2020	–	–	2.31	0.41
AbsNew	–	–	2.28	0.38

Table 4.18 The mean and Standard Deviation (SD) of the entropies for each dataset.

4.5.3 Explaining the Analyser

Since the analyser works by computing the average of the log-likelihoods for each token, calculating each token's attributions is straightforward. For better visualisation, we introduce the following changes:

1. *Exponentiation*: we work with $p_i := p_{\text{model}}(d_i \mid d_1, \ldots, d_{i-1})$ rather than the logarithms used in entropy.

2. *Min-max scaling*: to ensure that the probabilities are dispersed in the interval $[0, 1]$ we scale them using the formula:

$$p'_i = \frac{p_i - \min_j(p_j)}{\max_j(p_j) - \min_j(p_j)}$$

3. *Non-linear transformation*: to better distinguish between the probabilities close to 0, we introduce non-linear transformation $a_i = \sqrt[5]{p_i}$ (similarly to the sample-level visualisation in Section 4.1.3).

Finally, we colour each token as $\text{colour}_i = \text{RGBA}(40, 160, 80, a_i)$. Figure 4.9 provides an example visualisation of a text from a non-native speaker. In most cases, the first token of a word has a smaller likelihood, while the following tokens are more likely. This behaviour is intuitive: the first token provides most information about the intended word, while the following tokens can be 'guessed' from the context.

Tokens containing an error are an exception to this rule: even if the first token is known, the following tokens have low likelihoods. Again, this behaviour is expected,

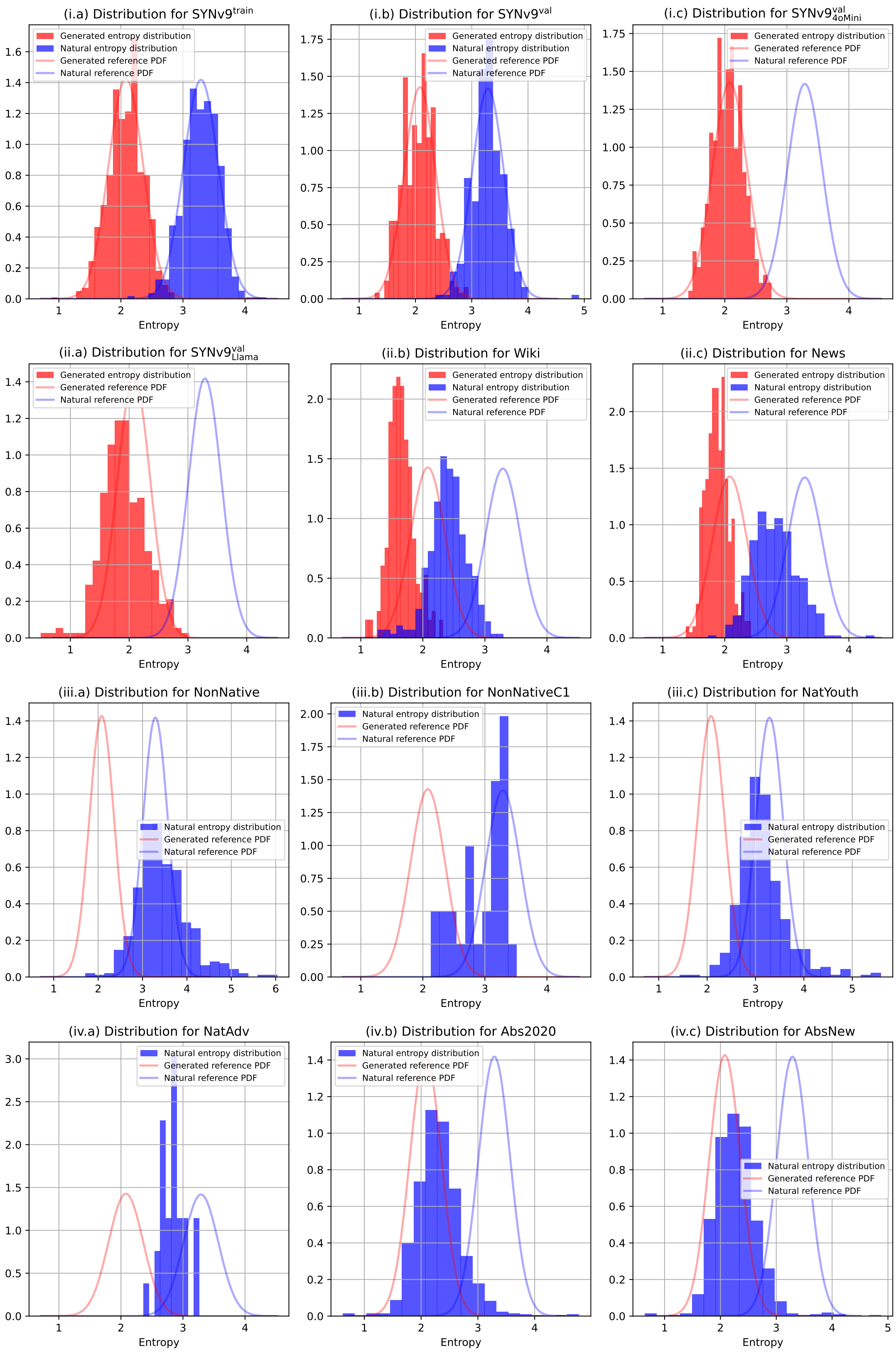


Figure 4.8 Entropy distributions of all the datasets compared to the density estimated on SYNv9$^{\text{TRAIN}}$.

as grammatical errors are not common in training corpora. Furthermore, the grammatical errors that are included are usually not systematic, i.e. different errors are made for the same words. Therefore, this likely explains the higher entropy of the texts from non-native speakers compared to their native counterparts.

Figure 4.9 Tokens' contributions to entropy, written by a non-native speaker. Darker tokens have higher likelihoods. Start-word tokens have a space at the beginning of their boxes. The first 100 tokens (in grey) are used to introduce the context, and their likelihood is not measured. Non-introductory tokens with spelling or grammatical errors have red borders.

4.5.4 Conclusion and Discussion

In this section, we introduced a logit-based approach that analyses the entropy of a text with respect to a language model. The significance of this approach is its relationship to perplexity, which Liang et al. (2023) examined in their study on bias against non-native speakers in generated text detection.

We verified that, within a domain, generated texts have smaller entries than the human-written ones, with only a small overlap. However, the entropies differ significantly across domains, which makes it challenging to use this observation for a straightforward Bayesian classification. The difference likely stems from whether similar documents were included in the training data and from the structure of the text.

Unlike Liang et al. (2023), we did *not* find that the texts produced by non-native speakers have lower entropies: our data suggested the opposite. This is likely because of the frequent grammatical errors in the non-native works. The difference was less significant for non-native speakers who are more advanced (NonNativeC1 vs. NatAdv), which supports this claim.

To provide a deeper understanding of the analyser, we created a method of visualising the per-sample likelihoods of the tokens. The visualisation provided an insight into the likelihood of the tokens in a multi-token word. Next, we used this visualisation to show that grammatical errors lead to a higher entropy of a document.

In Section 4.6, we use this analyser to examine whether our previously introduced models implicitly use the entropy to detect generated text.

4.6 Correlation Analysis

In previous sections, we introduced various methods for detecting generated text. In this section, we examine whether some of these methods use similar features to classify text; i.e. whether their outputs within one class[16] are correlated. The aim of this analysis is to (1) examine whether our detectors (implicitly) rely on entropy, as suggested by Liang et al. (2023), and (2) possibly provide some explanation of the methods we struggled to explain by relating them to more explainable methods.

To compute the correlation, we use the SciPy library (Virtanen et al., 2020), which, for two vectors $\mathbf{x}$ and $\mathbf{y}$ with their respective means $m_{\mathbf{x}}$ and $m_{\mathbf{y}}$, uses the following formula:

$$r = \frac{\sum_i (x_i - m_{\mathbf{x}})(y_i - m_{\mathbf{y}})}{\sqrt{\sum_i (x_i - m_{\mathbf{x}})^2 \sum_i (y_i - m_{\mathbf{y}})^2}}$$

Finally, to interpret the results, we use the thresholds described as 'conventional' by Schober et al. (2018), displayed in Table 4.19:

Absolute Correlation Value	Interpretation
$[0.0, 0.1)$	negligible correlation
$[0.1, 0.4)$	weak correlation
$[0.4, 0.7)$	moderate correlation
$[0.7, 0.9)$	strong correlation
$[0.9, 1.0]$	very strong correlation

Table 4.19 Interpretations of the correlation values.

4.6.1 Model Correlations

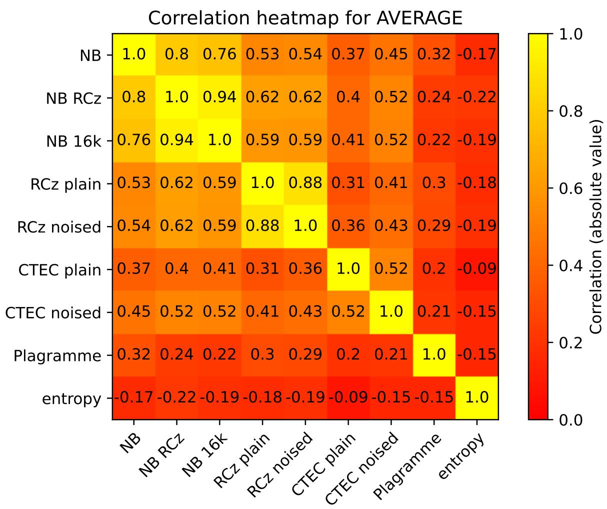
For our comparison, we selected one or more representatives of each category of the introduced models:

- *NB*: the NB classifier with TF-IDF features with word-level tokenisation, using LowNorm during training and inference (introduced in Section 4.1).
- *NB RCz*: the NB classifier with TF-IDF features with RobeCzech tokeniser and no pre-processing (introduced in Section 4.1.2).
- *NB 16k*: the NB classifier with TF-IDF features with the small 16 k terms vocabulary tokeniser trained in Section 4.3 (this model has not been formally introduced before, but its functionality is closely related to the classifier from Section 4.1.2).
- *RCz plain*: the RobeCzech classifier with no pre-processing (introduced in Section 4.2).
- *RCz noised*: the RobeCzech classifier with RandAug applied during training and inference (introduced in Sacetion 4.2.5).

[16] Note that correlations in mixed-class data are trivial for any functional detectors.

- *CTEC plain*: the Custom Transformer Encoder-only Classifier with no pre-processing (introduced in Section 4.3).
- *CTEC noised*: the Custom Transformer Encoder-only Classifier with RandAug applied during training and inference (introduced in Section 4.3).
- *Plagramme*: the commercial generated text detector with SpaceNorm applied on inference (introduced in Section 4.4).
- *entropy*: the logit-based entropy analyser (introduced in Section 4.5).

The per-dataset average correlation heatmap is shown in Figure 4.10, and all datasets are shown in the attachment, in Figure A.6. The figure shows an unsurprising strong correlation between the models from the same family, with the only exception being *CTEC plain* vs. *CTEC noised*. Importantly, the correlation of all the models with *entropy* is negligible or weak, contrary to Liang et al. (2023), although its value is systematically negative, which is expected. For illustration, we provide a sample-level comparison of the outputs of *entropy* vs. *CTEC noised* on the $\text{Wiki}_{\text{GPT4o}}$ dataset[17] in Figure 4.11.



	NB	NB RCz	NB 16k	RCz plain	RCz noised	CTEC plain	CTEC noised	Plagramme	entropy
NB	1.0	0.8	0.76	0.53	0.54	0.37	0.45	0.32	-0.17
NB RCz	0.8	1.0	0.94	0.62	0.62	0.4	0.52	0.24	-0.22
NB 16k	0.76	0.94	1.0	0.59	0.59	0.41	0.52	0.22	-0.19
RCz plain	0.53	0.62	0.59	1.0	0.88	0.31	0.41	0.3	-0.18
RCz noised	0.54	0.62	0.59	0.88	1.0	0.36	0.43	0.29	-0.19
CTEC plain	0.37	0.4	0.41	0.31	0.36	1.0	0.52	0.2	-0.09
CTEC noised	0.45	0.52	0.52	0.41	0.43	0.52	1.0	0.21	-0.15
Plagramme	0.32	0.24	0.22	0.3	0.29	0.2	0.21	1.0	-0.15
entropy	-0.17	-0.22	-0.19	-0.18	-0.19	-0.09	-0.15	-0.15	1.0

Figure 4.10 Per-dataset average correlation heatmap for the compared models.

Furthermore, there was a moderate correlation between the outputs of the RobeCzech classifier and those of the NB classifier with the same tokeniser (and similarly for the CTEC). This suggests that, to some extent, the classifiers work with the same principle. Thanks to this insight, we know that the knowledge from explaining the NB classifier (see Section 4.1.3) also partially applies to the Transformer-based models. Again, we illustrate this correlation on $\text{Wiki}_{\text{GPT4o}}$ in Figure 4.12.

[17]The second model was chosen as one of the best-performing models and the dataset was chosen as one of the datasets on which the model struggled, to obtain more variance in the outputs.

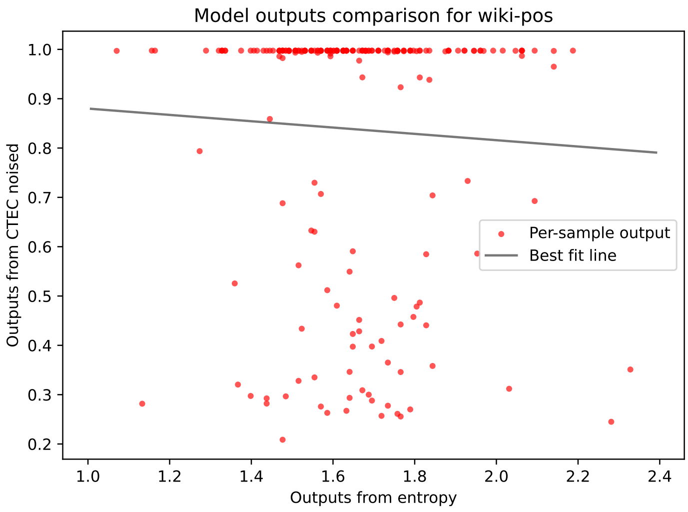


Figure 4.11 Sample-level comparison of the outputs of *entropy* vs. *CTEC noised* on the $\text{WIKI}_{\text{GPT4o}}$ dataset.

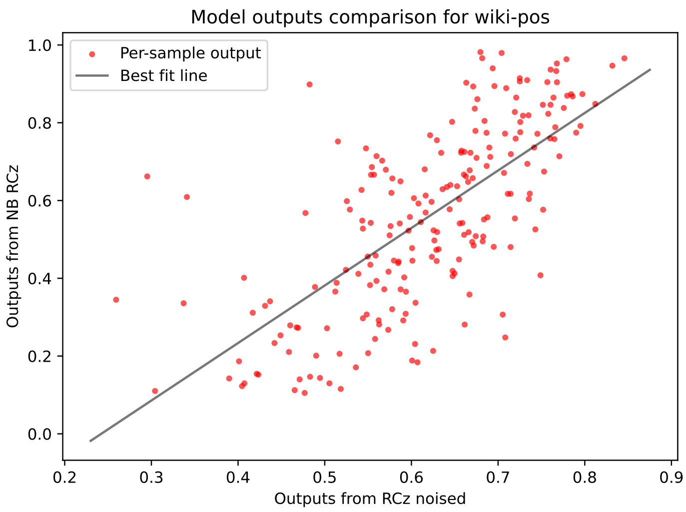


Figure 4.12 Sample-level comparison of the outputs of *RCz noised* vs. *NB RCz* (which both use the same tokeniser) on the $\text{WIKI}_{\text{GPT4o}}$ dataset.

The correlations on the $\text{SYNv9}^{\text{VAL}}_{\text{LLAMA}}$ dataset (shown in Figure 4.13) are considerably higher compared to the other datasets. The correlation rose to 'strong correlation' levels for all models except two: *CTEC plain* (where it remained moderate but higher than average) and *entropy* (where it remained weak). Hypothetically, this could mean that while the more complex models work with more complex, model-specific features, when these features are not present, they resort to lexical features, which are primarily used in NB.

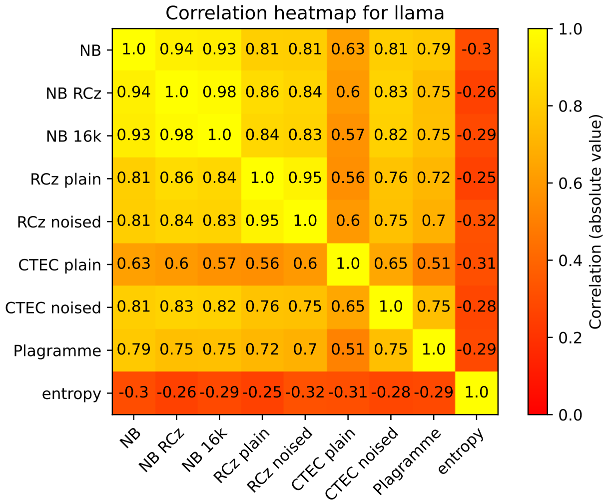


Figure 4.13 Correlation heatmap for the $\text{SYNv9}^{\text{VAL}}_{\text{LLAMA}}$ dataset.

4.6.2 Conclusion and Duscussion

In this section, we provided a critical comparison of the introduced methods using within-class correlations. A key finding was that the outputs of the models are only weakly correlated with the entropy of the texts, showing that the models do *not* implicitly rely on the entropy. This finding is opposed to the findings of Liang et al. (2023).

The correlation analysis also provided a certain insight into the models: it was shown that the correlation between Transformer-based models and TF-IDF/NB models is moderate if they use the same tokeniser, suggesting that the former family of models also works with lexical features, to some extent. This correlation is even stronger when measured on data generated by a model on which the models were not trained. We hypothesise that this is caused by the models learning more complex model-specific structures and falling back on lexical features when they are not present.

Conclusion

In this thesis, we provided a comprehensive follow-up to the work of Liang et al. (2023) – who examine the bias of generated text detectors on non-native speakers – in the context of the Czech language. We introduced the datasets and data pre-processing pipelines. Next, we used these datasets to train and evaluate models based on three commonly used architectures. For comparison, we further included a commercial AI detector. We found that while achieving high performance on the training domain was trivial, even with simple models, all of our models performed considerably worse when the domain was significantly different. We attempted to explain the models, which was only partially successful.

To answer the first question raised in the Introduction – i.e. whether Czech generated text detectors are biased against non-native speakers – we evaluated the detectors on two pairs of comparable datasets of text from non-native/native speakers: NonNative/NatYouth and NonNativeC1/NatAdv. We did *not* find a systematic bias against the non-native speakers: some detectors performed better on non-native texts, while others performed better on those from the native speakers.

As for the second question, whether the texts from non-native Czech speakers have lower perplexities, we created an entropy-based analyser and found the opposite: on average, the entropy (and, therefore, the perplexity) of the texts from non-native Czech speakers is *greater* than that of the comparable texts from native speakers. We attribute this phenomenon to frequent grammatical errors, particularly in declension. This explains the difference from the experiments on English texts, where the declension is considerably less complex.

Finally, we answered the third question – whether it is possible to create generated text detectors that do not rely on perplexity – by first creating architectures that do not explicitly work with perplexity. Next, to ensure that these classifiers do not have some implicit representation of perplexity, we compared the correlations of their outputs within one class to our entropy-based analyser and confirmed that the correlation was insignificant. Therefore, we concluded that our detectors do *not* rely on perplexity. Moreover, the correlation analysis provided further insight into the functionality of the models.

Limitations

Despite our efforts, the approaches presented in this work have some limitations that are worth considering. We consider the sourcing of the non-native texts to be the main limitation of this work. While we were able to source the 'NonNative' dataset of 450 documents, the speakers' proficiency in Czech was insufficient. To mitigate this, we obtained a dataset of non-native speakers with a higher proficiency: 'NonNativeC1'. However, this dataset was considerably smaller (29 documents), and grammar mistakes were still prevalent. As a result, the prediction of the detectors was largely impacted by grammar and spelling errors, rather than lexical and syntactic properties of the texts.

Another limitation is the fact that we were unable to create an industry-level generated text detector that is robust across domains and source language models. We partially addressed this issue by employing a commercial detector and evaluating it in the same way as our detectors. However, given the closed-source nature of the detector, our understanding of its functionality is limited to 'black box' observations.

Finally, the scope of the architectures of the detectors and the scope of the source models of the generated datasets are limited. For example, detectors based on linguistic features were not considered, and the source models for the generated text were limited to the GPT-4o family (OpenAI; Hurst et al., 2024) and Llama 3 (Grattafiori et al., 2024). Expanding those scopes could provide more insight into the problem of generated text detection.

Future Work

For future work, we suggest sourcing a more representative dataset of texts from highly proficient Czech non-native speakers to obtain more comparable results to related work. Next, a more comprehensive training dataset should be created that covers diverse domains and language models, and a new generation of detectors should be trained on the dataset. The correlation analysis could be used to identify models whose error is not correlated to create an ensemble model, which could achieve industry standards. The improved model could then be re-evaluated for a possible bias against non-native speakers.

A significant finding of this thesis is that the bias against non-native speakers is largely language-dependent. Consequently, future work might explore languages other than English and Czech to examine how the conclusions differ and whether there are any general, language-independent characteristics of the generated text detection problem.

Bibliography

AICH, Ankit; BHATTACHARYA, Souvik; PARDE, Natalie, 2022. Demystifying neural fake news via linguistic feature-based interpretation. In: *Proceedings of the 29th International Conference on Computational Linguistics*, pp. 6586–6599.

ANTOUN, Wissam; MOUILLERON, Virginie; SAGOT, Benoît; SEDDAH, Djamé, 2023. Towards a Robust Detection of Language Model-Generated Text: Is ChatGPT that easy to detect? In: SERVAN, Christophe; VILNAT, Anne (eds.). *Actes de CORIA-TALN 2023. Actes de la 30e Conférence sur le Traitement Automatique des Langues Naturelles (TALN), volume 1 : travaux de recherche originaux – articles longs*. Paris, France: ATALA, pp. 14–27. Available also from: `https://aclanthology.org/2023.jeptalnrecital-long.2/`.

ATTALI, Yigal; BURSTEIN, Jill, 2006. Attali, Y., & Burstein, J. (2006). Automated essay scoring with e-rater® V.2. Journal of Technology, Learning, and Assessment, 4(3). *Journal of Technology, Learning, and Assessment*. Vol. 4.

AUTHOR, (:unav) Unknown, 2011. *czes*. Available also from: `http://hdl.handle.net/11858/00-097C-0000-0001-CCCF-C`. LINDAT/CLARIAH-CZ digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

BISHOP, Christopher, 2006. *Pattern Recognition and Machine Learning*. Springer. Available also from: `https://www.microsoft.com/en-us/research/publication/pattern-recognition-machine-learning/`.

BROWN, Tom B.; MANN, Benjamin; RYDER, Nick; SUBBIAH, Melanie; KAPLAN, Jared; DHARIWAL, Prafulla; NEELAKANTAN, Arvind; SHYAM, Pranav; SASTRY, Girish; ASKELL, Amanda; AGARWAL, Sandhini; HERBERT-VOSS, Ariel; KRUEGER, Gretchen; HENIGHAN, Tom; CHILD, Rewon; RAMESH, Aditya; ZIEGLER, Daniel M.; WU, Jeffrey; WINTER, Clemens; HESSE, Christopher; CHEN, Mark; SIGLER, Eric; LITWIN, Mateusz; GRAY, Scott; CHESS, Benjamin; CLARK, Jack; BERNER, Christopher; MCCANDLISH, Sam; RADFORD, Alec; SUTSKEVER, Ilya; AMODEI, Dario, 2020. Language models are few-shot learners. In: *Proceedings of the 34th International Conference on Neural Information Processing Systems*. Vancouver, BC, Canada: Curran Associates Inc. NIPS '20. ISBN 9781713829546.

CORIZZO, Roberto; LEAL-ARENAS, Sebastian, 2023. A Deep Fusion Model for Human *vs*. Machine-Generated Essay Classification. In: *2023 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–10. Available from DOI: `10.1109/IJCNN54540.2023.10191322`.

CORTES, Corinna; VAPNIK, Vladimir, 1995. Support-Vector Networks. *Mach. Learn.* Vol. 20, no. 3, pp. 273–297. ISSN 0885-6125. Available from DOI: `10.1023/A:1022627411411`.

DANIEL HAN, Michael Han; UNSLOTH TEAM, 2023. *Unsloth*. Available also from: `http://github.com/unslothai/unsloth`.

DEVLIN, Jacob; CHANG, Ming-Wei; LEE, Kenton; TOUTANOVA, Kristina, 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In: BURSTEIN, Jill; DORAN, Christy; SOLORIO, Thamar (eds.). *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, pp. 4171–4186. Available from DOI: `10.18653/v1/N19-1423`.

FAN, Angela; LEWIS, Mike; DAUPHIN, Yann, 2018. Hierarchical Neural Story Generation. In: GUREVYCH, Iryna; MIYAO, Yusuke (eds.). *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Melbourne, Australia: Association for Computational Linguistics, pp. 889–898. Available from DOI: `10.18653/v1/P18-1082`.

GAGE, Philip, 1994. A new algorithm for data compression. *C Users J.* Vol. 12, no. 2, pp. 23–38. ISSN 0898-9788.

GEHRMANN, Sebastian; STROBELT, Hendrik; RUSH, Alexander, 2019. GLTR: Statistical Detection and Visualization of Generated Text. In: COSTA-JUSSÀ, Marta R.; ALFONSECA, Enrique (eds.). *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. Florence, Italy: Association for Computational Linguistics, pp. 111–116. Available from DOI: `10.18653/v1/P19-3019`.

GEMINI TEAM; ANIL, Rohan; BORGEAUD, Sebastian; ALAYRAC, Jean-Baptiste; YU, Jiahui; SORICUT, Radu; SCHALKWYK, Johan; DAI, Andrew M; HAUTH, Anja; MILLICAN, Katie et al., 2023. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*.

GHOLAMI, Amir; KIM, Sehoon; DONG, Zhen; YAO, Zhewei; MAHONEY, Michael W; KEUTZER, Kurt, 2022. A survey of quantization methods for efficient neural network inference. In: *Low-power computer vision*. Chapman and Hall/CRC, pp. 291–326.

GOODFELLOW, Ian; BENGIO, Yoshua; COURVILLE, Aaron, 2016. *Deep Learning*. MIT Press. `http://www.deeplearningbook.org`.

GRATTAFIORI, Aaron et al., 2024. *The Llama 3 Herd of Models*. Available from arXiv: `2407.21783 [cs.AI]`.

HAMNER, Ben; MORGAN, Jaison; LYNNVANDEV; SHERMIS, Mark; ARK, Tom Vander, 2012. *The Hewlett Foundation: Automated Essay Scoring* [`https://kaggle.com/competitions/asap-aes`]. Kaggle.

HARCHOL-BALTER, Mor, 2023. *Introduction to Probability for Computing*. Cambridge University Press. ISBN 9781009309073. Available from DOI: `10.1017/9781009309097`.

HERDAN, Simona; SHARVIT, Yael, 2006. Definite and Nondefinite Superlatives and NPI Licensing. *Syntax*. Vol. 9, pp. 1–31. Available from DOI: `10.1111/j.1467-9612.2006.00082.x`.

HO, Tin Kam, 1995. Random Decision Forest. In: *Proceedings of the 3rd International Conference on Document Analysis and Recognition (ICDAR)*. Montreal, Canada, pp. 278–282.

HOCHREITER, Sepp; SCHMIDHUBER, Jürgen, 1997. Long Short-Term Memory. *Neural Comput.* Vol. 9, no. 8, pp. 1735–1780. ISSN 0899-7667. Available from DOI: 10.1162/neco.1997.9.8.1735.

HOLTZMAN, Ari; BUYS, Jan; DU, Li; FORBES, Maxwell; CHOI, Yejin, 2020. The Curious Case of Neural Text Degeneration. In: *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net. Available also from: https://openreview.net/forum?id=rygGQyrFvH.

JELÍNEK, Tomáš; KŘIVAN, Jan; PETKEVIČ, Vladimír; SKOUMALOVÁ, Hana; SINDLEROVA, Jana, 2021. SYN2020: A New Corpus of Czech with an Innovated Annotation. In: pp. 48–59. ISBN 978-3-030-83526-2. Available from DOI: 10.1007/978-3-030-83527-9_4.

JIANG, Yang; HAO, Jiangang; FAUSS, Michael; LI, Chen, 2024. Detecting ChatGPT-generated essays in a large-scale writing assessment: Is there a bias against non-native English speakers? *Computers & Education*. Vol. 217, p. 105070. ISSN 0360-1315. Available from DOI: https://doi.org/10.1016/j.compedu.2024.105070.

KIBRIYA, Ashraf M.; FRANK, Eibe; PFAHRINGER, Bernhard; HOLMES, Geoffrey, 2005. Multinomial Naive Bayes for Text Categorization Revisited. In: WEBB, Geoffrey I.; YU, Xinghuo (eds.). *AI 2004: Advances in Artificial Intelligence*. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 488–499. ISBN 978-3-540-30549-1.

KOKHLIKYAN, Narine; MIGLANI, Vivek; MARTIN, Miguel; WANG, Edward; ALSALLAKH, Bilal; REYNOLDS, Jonathan; MELNIKOV, Alexander; KLIUSHKINA, Natalia; ARAYA, Carlos; YAN, Siqi; REBLITZ-RICHARDSON, Orion, 2020. *Captum: A unified and generic model interpretability library for PyTorch*. Available from arXiv: 2009.07896 [cs.LG].

KŘEN, Michal; CVRČEK, Václav; ČAPKA, Tomáš; ČERMÁKOVÁ, Anna; HNÁTKOVÁ, Milena; CHLUMSKÁ, Lucie; JELÍNEK, Tomáš; KOVÁŘÍKOVÁ, Dominika; PETKEVIČ, Vladimír; PROCHÁZKA, Pavel; SKOUMALOVÁ, Hana; ŠKRABAL, Michal; TRUNEČEK, Petr; VONDŘIČKA, Pavel; ZASINA, Adrian, 2016. *SYN v4: large corpus of written Czech*. Available also from: http://hdl.handle.net/11234/1-1846. LINDAT/CLARIAH-CZ digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

KŘEN, Michal; CVRČEK, Václav; ČAPKA, Tomáš; ČERMÁKOVÁ, Anna; HNÁTKOVÁ, Milena; CHLUMSKÁ, Lucie; JELÍNEK, Tomáš; KOVÁŘÍKOVÁ, Dominika; PETKEVIČ, Vladimír; PROCHÁZKA, Pavel; SKOUMALOVÁ, Hana; ŠKRABAL, Michal; TRUNEČEK, Petr; VONDŘIČKA, Pavel; ZASINA, Adrian Jan, 2016. SYN2015: Representative Corpus of Contemporary Written Czech. In: CALZOLARI, Nicoletta; CHOUKRI, Khalid; DECLERCK, Thierry; GOGGI, Sara; GROBELNIK, Marko; MAEGAARD, Bente; MARIANI, Joseph; MAZO, Helene; MORENO, Asuncion; ODIJK, Jan; PIPERIDIS, Stelios (eds.). *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*. Portorož, Slovenia: European Language Resources Association

(ELRA), pp. 2522–2528. Available also from: https://aclanthology.org/L16-1400/.

KŘEN, Michal; CVRČEK, Václav; HENYŠ, Jan; HNÁTKOVÁ, Milena; JELÍNEK, Tomáš; KOCEK, Jan; KOVÁŘÍKOVÁ, Dominika; KŘIVAN, Jan; MILIČKA, Jiří; PETKEVIČ, Vladimír; PROCHÁZKA, Pavel; SKOUMALOVÁ, Hana; ŠINDLEROVÁ, Jana; ŠKRABAL, Michal, 2021. *SYN v9: large corpus of written Czech*. Available also from: http://hdl.handle.net/11234/1-4635. LINDAT/CLARIAH-CZ digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

KUDO, Taku; RICHARDSON, John, 2018. SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing. In: BLANCO, Eduardo; LU, Wei (eds.). *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Brussels, Belgium: Association for Computational Linguistics, pp. 66–71. Available from DOI: 10.18653/v1/D18-2012.

LIANG, Weixin; YUKSEKGONUL, Mert; MAO, Yining; WU, Eric; ZOU, James, 2023. GPT detectors are biased against non-native English writers. *Patterns*. Vol. 4, no. 7, p. 100779. ISSN 2666-3899. Available from DOI: https://doi.org/10.1016/j.patter.2023.100779.

LIU, Haotian; LI, Chunyuan; WU, Qingyang; LEE, Yong Jae, 2023. Visual Instruction Tuning. In: OH, A.; NAUMANN, T.; GLOBERSON, A.; SAENKO, K.; HARDT, M.; LEVINE, S. (eds.). *Advances in Neural Information Processing Systems*. Curran Associates, Inc. Vol. 36, pp. 34892–34916. Available also from: https://proceedings.neurips.cc/paper_files/paper/2023/file/6dcf277ea32ce3288914faf369fe6de0-Paper-Conference.pdf.

LIU, Peter J.; SALEH, Mohammad; POT, Etienne; GOODRICH, Ben; SEPASSI, Ryan; KAISER, Lukasz; SHAZEER, Noam, 2018. *Generating Wikipedia by Summarizing Long Sequences*. Available from arXiv: 1801.10198 [cs.CL].

LIU, Yinhan; OTT, Myle; GOYAL, Naman; DU, Jingfei; JOSHI, Mandar; CHEN, Danqi; LEVY, Omer; LEWIS, Mike; ZETTLEMOYER, Luke; STOYANOV, Veselin, 2019. *RoBERTa: A Robustly Optimized BERT Pretraining Approach*. Available from arXiv: 1907.11692 [cs.CL].

LONGO, Luca; BRCIC, Mario; CABITZA, Federico; CHOI, Jaesik; CONFALONIERI, Roberto; SER, Javier Del; GUIDOTTI, Riccardo; HAYASHI, Yoichi; HERRERA, Francisco; HOLZINGER, Andreas; JIANG, Richard; KHOSRAVI, Hassan; LECUE, Freddy; MALGIERI, Gianclaudio; PÁEZ, Andrés; SAMEK, Wojciech; SCHNEIDER, Johannes; SPEITH, Timo; STUMPF, Simone, 2024. Explainable Artificial Intelligence (XAI) 2.0: A manifesto of open challenges and interdisciplinary research directions. *Information Fusion*. Vol. 106, p. 102301. ISSN 1566-2535. Available from DOI: https://doi.org/10.1016/j.inffus.2024.102301.

LOSHCHILOV, Ilya; HUTTER, Frank, 2017. *SGDR: Stochastic Gradient Descent with Warm Restarts*. Available from arXiv: 1608.03983 [cs.LG].

LOSHCHILOV, Ilya; HUTTER, Frank, 2019. *Decoupled Weight Decay Regularization*. Available from arXiv: 1711.05101 [cs.LG].

Majliš, Martin, 2011. *W2C – Web to Corpus – Corpora*. Available also from: `http://hdl.handle.net/11858/00-097C-0000-0022-6133-9`. LINDAT/CLARIAH-CZ digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Manning, Christopher D.; Raghavan, Prabhakar; Schütze, Hinrich, 2008. *Introduction to Information Retrieval*. USA: Cambridge University Press. isbn 0521865719.

Massarelli, Luca; Petroni, Fabio; Piktus, Aleksandra; Ott, Myle; Rocktäschel, Tim; Plachouras, Vassilis; Silvestri, Fabrizio; Riedel, Sebastian, 2020. How Decoding Strategies Affect the Verifiability of Generated Text. *Findings of ACL: EMNLP 2020*.

Milička, Jiří; Marklová, Anna; Drobil, Ondřej; Pospíšilová, Eva, 2025. *Humans can learn to detect AI-generated texts, or at least learn when they can't*. Available from arXiv: `2505.01877 [cs.CL]`.

Mitchell, Eric; Lee, Yoonho; Khazatsky, Alexander; Manning, Christopher D.; Finn, Chelsea, 2023. DetectGPT: zero-shot machine-generated text detection using probability curvature. In: *Proceedings of the 40th International Conference on Machine Learning*. Honolulu, Hawaii, USA: JMLR.org. ICML'23.

Mitchell, Tom M; Mitchell, Tom M, 1997. *Machine learning*. Vol. 1. McGraw-hill New York. No. 9.

Najjar, Ayat A.; Ashqar, Huthaifa I.; Darwish, Omar A.; Hammad, Eman, 2025. *Detecting AI-Generated Text in Educational Content: Leveraging Machine Learning and Explainable AI for Academic Integrity*. Available from arXiv: `2501.03203 [cs.CL]`.

Náplava, Jakub; Straka, Milan, 2019. Grammatical Error Correction in Low-Resource Scenarios. *arXiv preprint arXiv:1910.00353*.

OpenAI; Achiam, Josh et al., 2024. *GPT-4 Technical Report*. Available from arXiv: `2303.08774 [cs.CL]`.

OpenAI; Hurst, Aaron et al., 2024. *GPT-4o System Card*. Available from arXiv: `2410.21276 [cs.CL]`.

Ouyang, Long; Wu, Jeff; Jiang, Xu; Almeida, Diogo; Wainwright, Carroll L.; Mishkin, Pamela; Zhang, Chong; Agarwal, Sandhini; Slama, Katarina; Ray, Alex; Schulman, John; Hilton, Jacob; Kelton, Fraser; Miller, Luke; Simens, Maddie; Askell, Amanda; Welinder, Peter; Christiano, Paul; Leike, Jan; Lowe, Ryan, 2022. Training language models to follow instructions with human feedback. In: *Proceedings of the 36th International Conference on Neural Information Processing Systems*. New Orleans, LA, USA: Curran Associates Inc. NIPS '22. isbn 9781713871088.

Paszke, Adam; Gross, Sam; Massa, Francisco; Lerer, Adam; Bradbury, James; Chanan, Gregory; Killeen, Trevor; Lin, Zeming; Gimelshein, Natalia; Antiga, Luca; Desmaison, Alban; Köpf, Andreas; Yang, Edward; DeVito, Zach; Raison, Martin; Tejani, Alykhan; Chilamkurthy, Sasank; Steiner, Benoit; Fang, Lu; Bai, Junjie; Chintala, Soumith, 2019. PyTorch: an imperative style, high-performance deep learning library. In: *Proceedings of* *the 33rd International Conference on Neural Information Processing Systems*. Red Hook, NY, USA: Curran Associates Inc.

Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; Vanderplas, J.; Passos, A.; Cournapeau, D.; Brucher, M.; Perrot, M.; Duchesnay, E., 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*. Vol. 12, pp. 2825–2830.

Radford, Alec; Narasimhan, Karthik; Salimans, Tim; Sutskever, Ilya et al., 2018. *Improving language understanding by generative pre-training.(2018)*.

Radford, Alec; Wu, Jeffrey; Child, Rewon; Luan, David; Amodei, Dario; Sutskever, Ilya, 2019a. Language Models are Unsupervised Multitask Learners. *OpenAI*. Available also from: `https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf`.

Radford, Alec; Wu, Jeffrey; Child, Rewon; Luan, David; Amodei, Dario; Sutskever, Ilya et al., 2019b. Language models are unsupervised multitask learners. *OpenAI blog*. Vol. 1, no. 8, p. 9.

Rafailov, Rafael; Sharma, Archit; Mitchell, Eric; Ermon, Stefano; Manning, Christopher D.; Finn, Chelsea, 2023. Direct preference optimization: your language model is secretly a reward model. In: *Proceedings of the 37th International Conference on Neural Information Processing Systems*. New Orleans, LA, USA: Curran Associates Inc. NIPS '23.

Raffel, Colin; Shazeer, Noam; Roberts, Adam; Lee, Katherine; Narang, Sharan; Matena, Michael; Zhou, Yanqi; Li, Wei; Liu, Peter J., 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.* Vol. 21, no. 1. issn 1532-4435.

Ribeiro, Marco Tulio; Singh, Sameer; Guestrin, Carlos, 2016. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. San Francisco, California, USA: Association for Computing Machinery, pp. 1135–1144. KDD '16. isbn 9781450342322. Available from doi: `10.1145/2939672.2939778`.

Russell, Stuart J; Norvig, Peter, 2016. *Artificial intelligence: a modern approach*. pearson.

Schober, Patrick; Boer, Christa; Schwarte, Lothar A, 2018. Correlation Coefficients: Appropriate Use and Interpretation. *Anesth Analg*. Vol. 126, no. 5, pp. 1763–1768.

Schulman, John; Wolski, Filip; Dhariwal, Prafulla; Radford, Alec; Klimov, Oleg, 2017. *Proximal Policy Optimization Algorithms*. Available from arXiv: `1707.06347 [cs.LG]`.

scikit-learn developers, [n.d.]. *Naive Bayes*. scikit-learn. Available also from: `https://scikit-learn.org/stable/modules/naive_bayes.html`. Accessed: 2025-06-18.

Šebesta, Karel; Bedřichová, Zuzanna; Šormová, Kateřina; Štindlová, Barbora; Hrdlička, Milan; Hrdličková, Tereza; Hana, Jiří; Rosen, Alexandr; Petkevič, Vladimír; Jelínek, Tomáš; Škodová, Svatava; Poláčková, Marie; Janeš, Petr; Lundáková, Kateřina; Skoumalová, Hana; Šťastný, Klement; Sládek, Šimon; Pierscieniak, Piotr, 2012. *AKCES 3*. Available also from: http://hdl.handle.net/11858/00-097C-0000-000C-2112-B. LINDAT/CLARIAH-CZ digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Šebesta, Karel; Goláňová, Hana; Letafková, Jana; Jelínková, Blanka, 2016. *AKCES 1*. Available also from: http://hdl.handle.net/11234/1-1741. LINDAT/CLARIAH-CZ digital library at the Institute of Formal and Applied Linguistics (ÚFAL), Faculty of Mathematics and Physics, Charles University.

Sennrich, Rico; Haddow, Barry; Birch, Alexandra, 2016. Neural Machine Translation of Rare Words with Subword Units. In: Erk, Katrin; Smith, Noah A. (eds.). *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany: Association for Computational Linguistics, pp. 1715–1725. Available from doi: 10.18653/v1/P16-1162.

Shah, Aditya; Ranka, Prateek; Dedhia, Urmi; Prasad, Shruti; Muni, Siddhi; Bhowmick, Kiran, 2023. Detecting and Unmasking AI-Generated Texts through Explainable Artificial Intelligence using Stylistic Features. *International Journal of Advanced Computer Science and Applications*. Vol. 14, no. 10. Available from doi: 10.14569/IJACSA.2023.01410110.

Solaiman, Irene; Brundage, Miles; Clark, Jack; Askell, Amanda; Herbert-Voss, Ariel; Wu, Jeff; Radford, Alec; Krueger, Gretchen; Kim, Jong Wook; Kreps, Sarah; McCain, Miles; Newhouse, Alex; Blazakis, Jason; McGuffie, Kris; Wang, Jasmine, 2019. *Release Strategies and the Social Impacts of Language Models*. Available from arXiv: 1908.09203 [cs.CL].

Srivastava, Nitish; Hinton, Geoffrey; Krizhevsky, Alex; Sutskever, Ilya; Salakhutdinov, Ruslan, 2014. Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.* Vol. 15, no. 1, pp. 1929–1958. issn 1532-4435.

Straka, Milan; Náplava, Jakub; Straková, Jana; Samuel, David, 2021. RobeCzech: Czech RoBERTa, a Monolingual Contextualized Language Representation Model. In: *Text, Speech, and Dialogue: 24th International Conference, TSD 2021, Olomouc, Czech Republic, September 6–9, 2021, Proceedings*. Olomouc, Czech Republic: Springer-Verlag, pp. 197–209. isbn 978-3-030-83526-2. Available from doi: 10.1007/978-3-030-83527-9_17.

Su, Jianlin; Ahmed, Murtadha; Lu, Yu; Pan, Shengfeng; Bo, Wen; Liu, Yunfeng, 2024. RoFormer: Enhanced transformer with Rotary Position Embedding. *Neurocomput.* Vol. 568, no. C. issn 0925-2312. Available from doi: 10.1016/j.neucom.2023.127063.

SU, Jinyan; ZHUO, Terry; WANG, Di; NAKOV, Preslav, 2023. DetectLLM: Leveraging Log Rank Information for Zero-Shot Detection of Machine-Generated Text. In: BOUAMOR, Houda; PINO, Juan; BALI, Kalika (eds.). *Findings of the Association for Computational Linguistics: EMNLP 2023*. Singapore: Association for Computational Linguistics, pp. 12395–12412. Available from DOI: 10.18653/v1/2023.findings-emnlp.827.

SUCHOMEL, Vít; MICHELFEIT, Jan; POMIKÁLEK, Jan, 2014. Text Tokenisation Using unitok. In: *Eighth Workshop on Recent Advances in Slavonic Natural Language Processing* [tištěná verze "print"]. Brno: Tribun EU, pp. 71–75. Available also from: https://nlp.fi.muni.cz/raslan/2014/14.pdf.

SUNDARARAJAN, Mukund; TALY, Ankur; YAN, Qiqi, 2017. Axiomatic attribution for deep networks. In: *Proceedings of the 34th International Conference on Machine Learning - Volume 70*. Sydney, NSW, Australia: JMLR.org, pp. 3319–3328. ICML'17.

SUSNJAK, Teo; MCINTOSH, Timothy R., 2024. ChatGPT: The End of Online Exam Integrity? *Education Sciences*. Vol. 14, no. 6. ISSN 2227-7102. Available from DOI: 10.3390/educsci14060656.

VASILATOS, Christoforos; ALAM, Manaar; RAHWAN, Talal; ZAKI, Yasir; MANIATAKOS, Michail, 2023. Howkgpt: Investigating the detection of chatgpt-generated university student homework through context-aware perplexity analysis. *arXiv preprint arXiv:2305.18226*.

VASWANI, Ashish; SHAZEER, Noam; PARMAR, Niki; USZKOREIT, Jakob; JONES, Llion; GOMEZ, Aidan N; KAISER, Łukasz; POLOSUKHIN, Illia, 2017. Attention is All you Need. In: GUYON, I.; LUXBURG, U. Von; BENGIO, S.; WALLACH, H.; FERGUS, R.; VISHWANATHAN, S.; GARNETT, R. (eds.). *Advances in Neural Information Processing Systems*. Curran Associates, Inc. Vol. 30. Available also from: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

VIRTANEN, Pauli; GOMMERS, Ralf; OLIPHANT, Travis E.; HABERLAND, Matt; REDDY, Tyler; COURNAPEAU, David; BUROVSKI, Evgeni; PETERSON, Pearu; WECKESSER, Warren; BRIGHT, Jonathan; VAN DER WALT, Stéfan J.; BRETT, Matthew; WILSON, Joshua; MILLMAN, K. Jarrod; MAYOROV, Nikolay; NELSON, Andrew R. J.; JONES, Eric; KERN, Robert; LARSON, Eric; CAREY, C J; POLAT, İlhan; FENG, Yu; MOORE, Eric W.; VANDERPLAS, Jake; LAXALDE, Denis; PERKTOLD, Josef; CIMRMAN, Robert; HENRIKSEN, Ian; QUINTERO, E. A.; HARRIS, Charles R.; ARCHIBALD, Anne M.; RIBEIRO, Antônio H.; PEDREGOSA, Fabian; VAN MULBREGT, Paul; SCIPY 1.0 CONTRIBUTORS, 2020. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*. Vol. 17, pp. 261–272. Available from DOI: 10.1038/s41592-019-0686-2.

WARNER, Benjamin; CHAFFIN, Antoine; CLAVIÉ, Benjamin; WELLER, Orion; HALLSTRÖM, Oskar; TAGHADOUINI, Said; GALLAGHER, Alexis; BISWAS, Raja; LADHAK, Faisal; AARSEN, Tom; COOPER, Nathan; ADAMS, Griffin; HOWARD, Jeremy; POLI, Iacopo, 2024. Smarter, Better, Faster, Longer: A Modern Bidirectional Encoder for Fast, Memory Efficient, and Long Context Finetuning and Inference. *CoRR*. Vol. abs/2412.13663. Available from DOI: 10.48550/ARXIV.2412.13663.

WOLF, Thomas; DEBUT, Lysandre; SANH, Victor; CHAUMOND, Julien; DELANGUE, Clement; MOI, Anthony; CISTAC, Pierric; RAULT, Tim; LOUF, Remi; FUNTOWICZ, Morgan; DAVISON, Joe; SHLEIFER, Sam; PLATEN, Patrick von; MA, Clara; JERNITE, Yacine; PLU, Julien; XU, Canwen; LE SCAO, Teven; GUGGER, Sylvain; DRAME, Mariama; LHOEST, Quentin; RUSH, Alexander, 2020. Transformers: State-of-the-Art Natural Language Processing. In: LIU, Qun; SCHLANGEN, David (eds.). *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Online: Association for Computational Linguistics, pp. 38–45. Available from DOI: 10.18653/v1/2020.emnlp-demos.6.

WU, Junchao; YANG, Shu; ZHAN, Runzhe; YUAN, Yulin; CHAO, Lidia Sam; WONG, Derek Fai, 2025. A Survey on LLM-Generated Text Detection: Necessity, Methods, and Future Directions. *Computational Linguistics*. Vol. 51, no. 1, pp. 275–338. ISSN 0891-2017. Available from DOI: 10.1162/coli_a_00549.

YULE, C Udny, 1944. *The statistical study of literary vocabulary*. Cambridge University Press.

ZHANG, Aston; LIPTON, Zachary C.; LI, Mu; SMOLA, Alexander J., 2023. *Dive into Deep Learning*. Cambridge University Press. https://D2L.ai.

List of Figures

List of Tables

List of Abbreviations

Acc Accuracy. 12, 39, 40, 46–48, 54, 56, 57

AI Artificial Intelligence. 8, 25, 27–29, 57, 69

BERT Bidirectional Encoder Representations from Transformers. 20, 21, 23, 26, 27

BPE Byte Pair Encoding. 17, 52

CTEC Custom Transformer Encoder-only Classifier. 52–56, 66–68, 80–82, 87

DPO Direct Preference Optimization. 24

FNR False Negative Rate. 12, 39, 40, 45–48, 53, 54, 56–58

FPR False Positive Rate. 12, 28, 29, 39, 40, 43, 45–48, 53, 54, 56–58

GPT Generative Pre-trained Transformer. 23

GRE Graduate Record Examinations. 28

IG Integrated Gradients. 29, 49, 51, 55

inf. informal. 40–42

LIME Local Interpretable Model-agnostic Explanations. 30, 49–51, 55, 80, 81, 87

lit. literary. 40–42

LLM Large Language Model. 8, 23, 25, 26, 32, 38, 43, 58, 59, 61

LR Learning Rate. 12, 13, 53

MAP Maximum a Posteriori. 15

ML Machine Learning. 10, 14, 16, 26–28

MLM Masked Language Model. 20, 21

MLP Multilayer Perceptron. 16

MT Machine Translation. 17, 19, 22

NB Naïve Bayes. 14–16, 38–40, 42, 43, 45, 47, 50, 51, 65–68, 80, 82

NLL Negative Log-Likelihood. 12, 13, 20, 59

NLP Natural Language Processing. 10, 17, 19–21

NSP Next Sentence Prediction. 21

PMF Probability Mass Function. 11

PPO Proximal Policy Optimisation. 23, 24

RCz RobeCzech. 65, 67, 80

ReLU Rectified Linear Unit. 16

RLHF Reinforcement Learning From Human Feedback. 22, 23

RoPE Rotary Positional Embeddings. 18, 19

SD Standard Deviation. 61, 62, 82

SGD Stochastic Gradient Descent. 13

SVM Support Vector Machine. 26–29

TF-IDF Term Frequency-Inverse Document Frequency. 15, 16, 26, 38–41, 43, 45, 65, 68, 82

TOEFL Test of English as a Foreign Language. 28

TTR Type-Token Ratio. 27

w.r.t. with respect to. 28, 29, 49, 59, 64, 80, 81, 87

xAI Explainable AI. 29

A Attachments

A.1 Figures

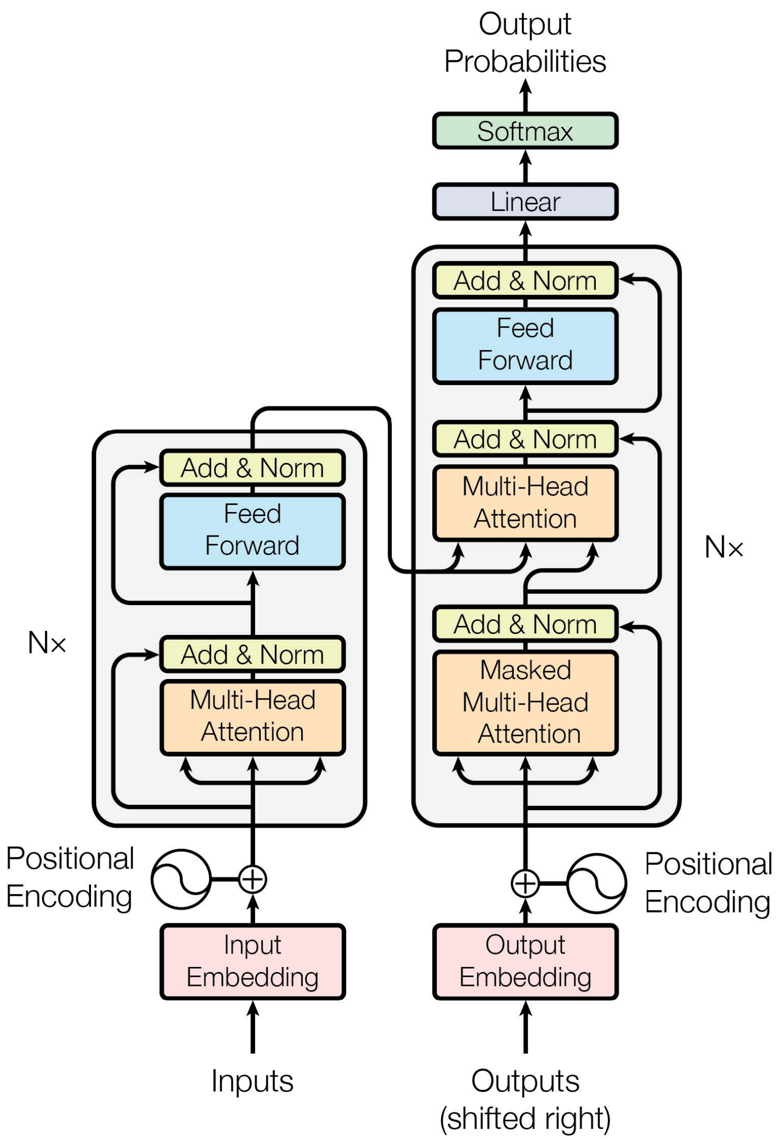


Figure A.1 Scheme of the Transformer architecture. Source: Vaswani et al. (2017).

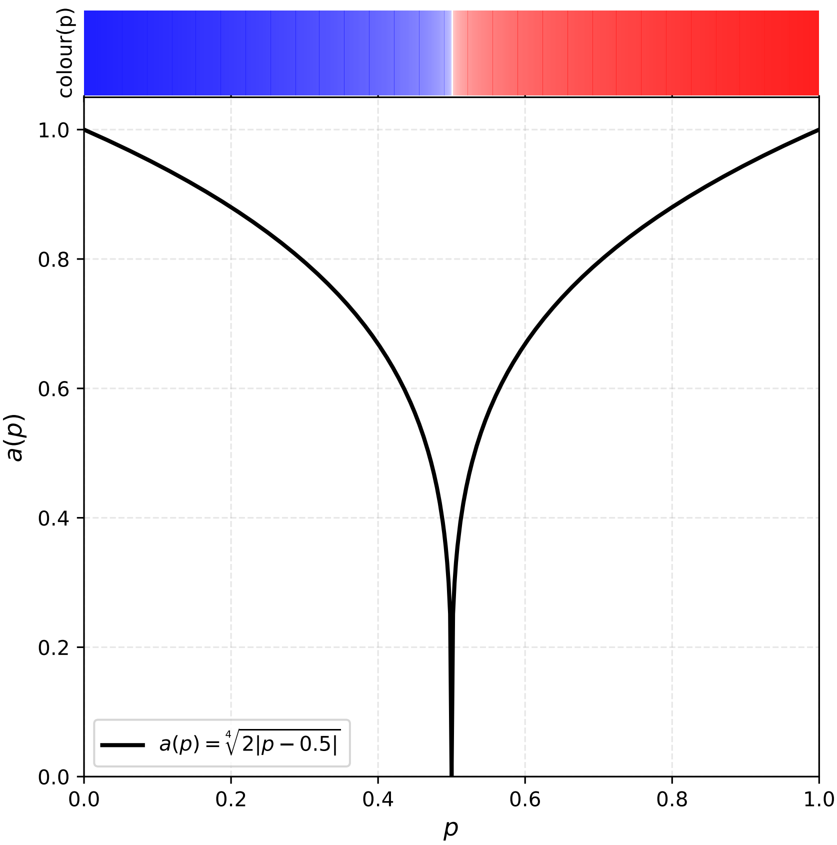


Figure A.2 Visualisation of the sample-level colouring function from Section 4.1.3.

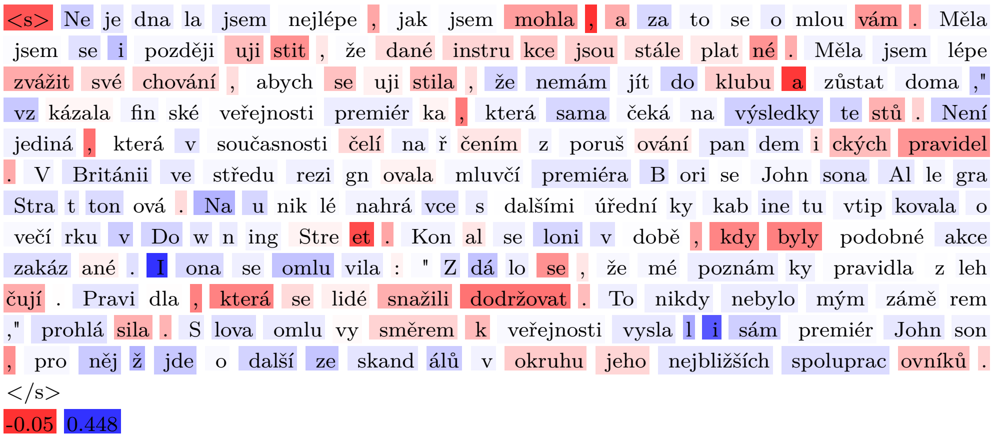


Figure A.3 Example result of selective token masking in the CTEC with masking window length of 1. The baseline probability was $b = 0.39$, and the true class was 0 (human-written). Red tokens contribute more to positive classification. The two boxes on the last line show the maximum negative/positive change w.r.t. b.

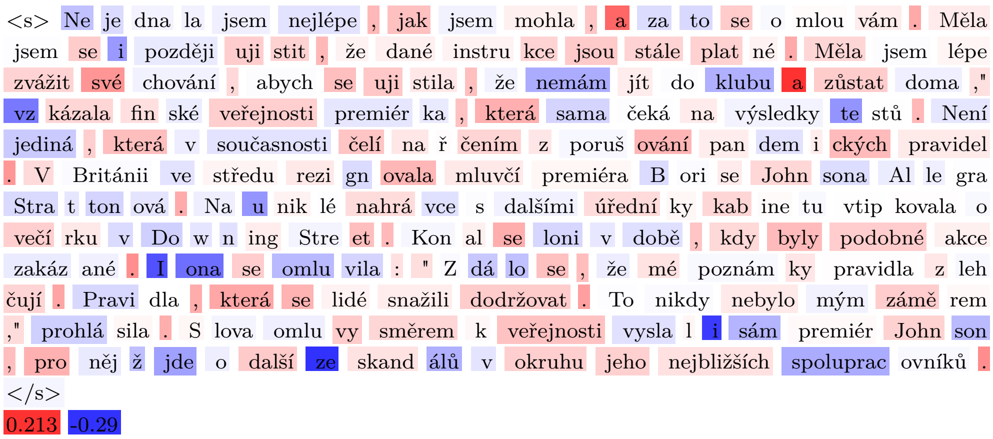


Figure A.4 Example result of the Integrated Gradients explanation method applied on the CTEC. The true label was 0 (human-written), and the model's output was 0.39. Red tokens contribute more to positive classification. The two boxes on the last line show the maximum negative/positive contributions.

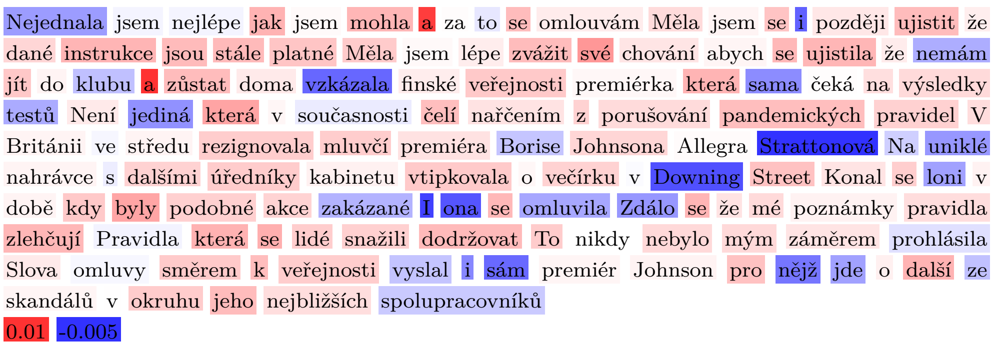


Figure A.5 Example result of the LIME explanation method applied on the CTEC. The true label was 0 (human-written), and the model's output was 0.39. Red tokens contribute more to positive classification. The two boxes on the last line show the maximum negative/positive contributions.

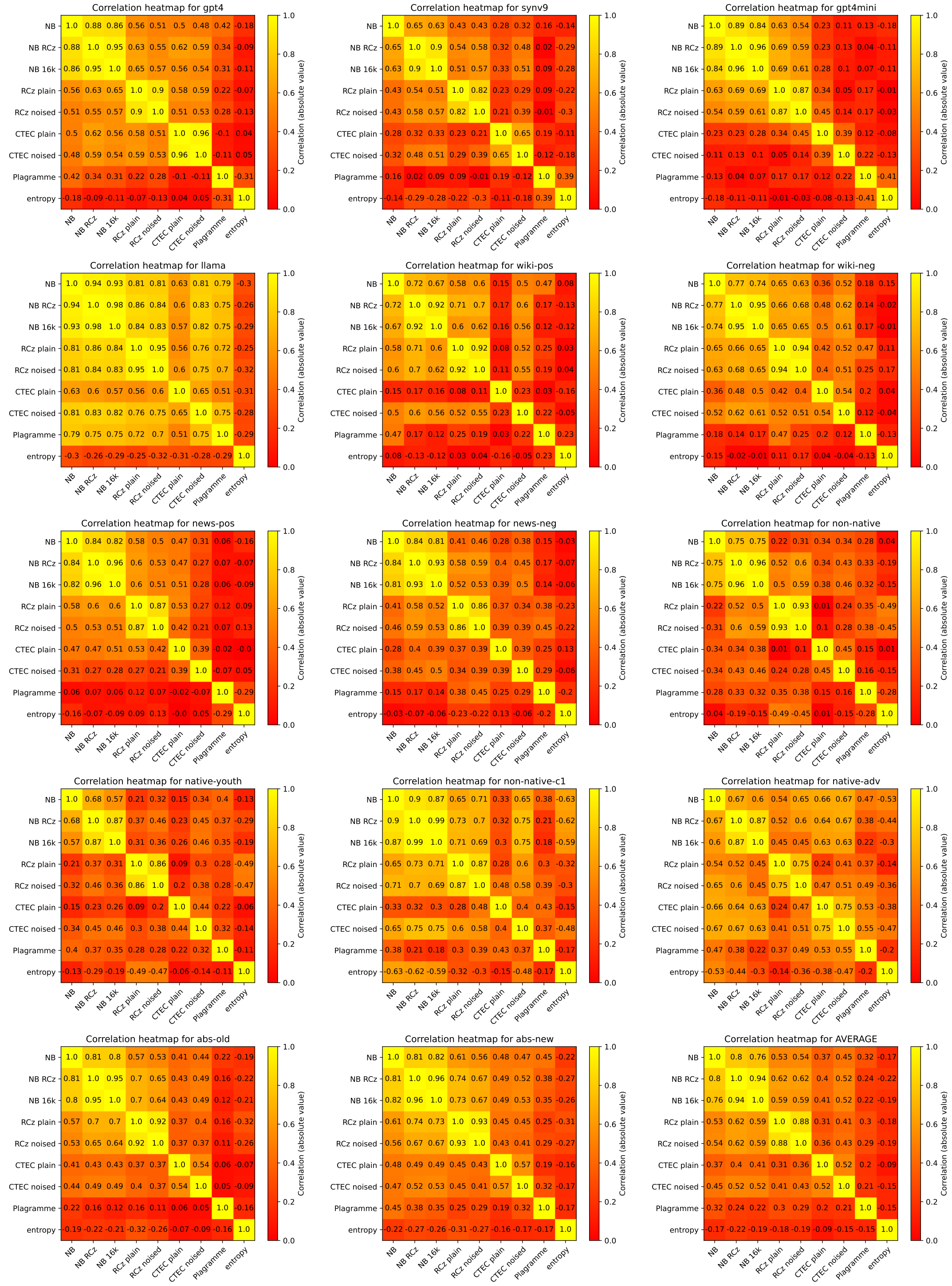


Figure A.6 Correlation heatmaps for each dataset for the compared models (see Section 4.6.1).