Detection of machine-generated fragments in text

Draft

Anastasiya Voznyuk
Department of Applied Informatics and Mathematics
Moscow Institute of Physics and Technology
vozniuk.ae@phystech.edu

Andrey Grabovoy
Moscow Institute of Physics and Technology
Antiplagiat Company
grabovoy@ap-team.ru

Abstract

This paper considers the problem of detecting machine-generated parts of text in the document. We introduce a model, that detects such text fragments and distinguish their origin among several language models. To solve this problem we combine two models. The objective of the first one is to divide document into fragments of human-written text and machine-generated text. The second model aims to classify the generated fragments according to the language model from which they were obtained. In the computational experiment, we analyse the quality of such approach on dataset of documents with fragments generated by GPT-2 and GPT-3.

Keywords Text Generation, Detection of Machine-generated text, Neural Authorship Attribution, Machine Learning, Natural Language Processing

1 Introduction

In recent years there has been a rapid development of language models for text generation, transformers in particular, for example GPT[14], GPT-2[15], GPT-3[5], CTRL[9] and mT5[18]. Problem of detecting machine-generated text has become more relevant recently due to the released ChatGPT model by OpenAI. This model was learnt on massive amounts of data and now is able to provide texts that are hardly distinguishable from human texts. The opposite task of detecting machine-generated text becomes more important due to many possible malicious usages of this technology. One can analyse the whole text or fragments of text. We will focus on methods of subtracting the text-generated fragments in the set of documents.

With increased computational resources past statistical methods were applied to the novel detection problem. One of them was prediction entropy as an indicator of fake text, as described in [11]. Also perplexity[4] of the text or frequency of rare bigrams [7] or value of tf-idf [16] can be taken into account. Another approach is to use classifiers that try to label given texts [10].

For a long period recursive neural networks like LSTM were showing best results at solving the problem of detection. At 2018 the mechanism of self-attention[17] was introduced and transformers have become new state-of-the-art approach. Every new model has its own features and is usually learnt on larger amounts of data, but attention mechanism always stays. These model can be used in detection of machine-generated text in two ways[16]. The first method is using a language model that searches for artefacts from methods, which most models are using for generating concise texts [6]. Additional training on new data is not required. The second method is fine-tuning based detection: one fine-tunes a language model to "detect itself" with using some stochastic methods, for example top-k, top-p sampling etc.[8].

Our approach consists of two steps. The first step is to divide each document into non-overlapping blocks of sentences. We take the baseline from work[10] on plagiarism detection. We measure the quality of our model using metrics from text segmentation and clustering, like BCubed metric. We suppose, that within a fragment with artificial text there is a similar semantic structure. The next step is to detect the origin of all the fragments, whether they are machine-generated text and this can be done with usage of transformer-based models, similar to [8]. This papers presents computational experiments with these models to combine them in one model and analyse the parameters on which the best performance will be obtained. Dataset will be constructed from existing datasets, such as GPT-2 output dataset[1] and RUaTD competition dataset[2].

2 Problem statement

Let $\mathbf{W}$ be our alphabet, tokens consists from characters from that alphabet.

Let

$$\mathbb{D} = \left\{ \Big[t_j \Big]_{j=1}^{n} \ \mid \ t_j \in \mathbf{W}, n \in \mathbb{N} \right\}$$

be the space of our documents.

We have set of n documents

$$\mathbf{D} = \bigcup_{i=1}^{n} D^i, D^i \in \mathbb{D}$$

We have K classes of our text-generative models, that generated fragments of text in $\mathbf{D}$.

We have $K+1$ labels, that we will use for multiclass classification, where 0 is the label of human-written fragment, $1...K$ are the labels that represent corresponding language model, let $\mathbf{C}$ be our set of labels.

Our model ϕ is a superposition of two mappings, $\mathbf{f}$ and $\mathbf{g}$. Mapping $\mathbf{f}$ is responsible for text segmentation. Mapping $\mathbf{g}$ is responsible for classifying obtained fragments.

$$\phi : \mathbf{g} \circ \mathbf{f}$$

$$\phi : \mathbb{D} \to \mathbb{T} \qquad \mathbb{T} = \left\{ \Big[t_{s_j}, t_{f_j}, C_j \Big]_{j=1}^{J} \ \mid \ t_{s_j} = t_{f_{j-1}}, \ \ s_j \in \mathbb{N}_0, \ \ f_j \in \mathbb{N}, \ \ C_j \in \mathbf{C} \right\},$$

where J - number of fragments in segmentation of the document, t_{s_j} is starting index of the j-th fragment, t_{f_j} is finishing index of the j-th fragment, C_j is class of the j-th fragment.

2.1 Text fragmentation

The first step is to divide our text into sequential non-overlapping fragments of different origin.

$$\mathbf{f} : \mathbb{D} \to \mathbf{T}^* \qquad \mathbf{T}^* = \left\{ \Big[t_{s_j}, t_{f_j} \Big]_{j=1}^{J} \quad \mid \quad t_{s_j} = t_{f_{j-1}}, s_j \in \mathbb{N}_0, f_j \in \mathbb{N} \right\},$$

where s_j - start of the j-th fragment and f_j - end of the j-th fragment.

Thus, $\mathbf{T}^*$ is a set of all possible sequences of non-overlapping blocks of texts, that covers complete document.

We assume that in each document blocks of text with the same origin is big enough. Thus, we search for style changes on sentence levels. Style change indicates that several next sentences staring from the beginning of the style change have different origin. Each sentence receive a label 0 or 1, where 0 means it is a human-written sentence and 1 is a machine-generated sentence. A fragment is a sequence of neighbouring sentences of maximum possible length, that have the same label. We repeat this process for every document in given set of document.

We will cluster neighbouring fragments, that have got the same label and consider these fragments as one larger fragment.

2.2 Fragment classification

The second step is to classify each fragment with label from $\mathbf{C}$

$$\mathbf{g} : \mathbf{T}^* \to \mathbf{C}.$$

We apply that function to every text fragment, obtained from the previous step. This is a classical multi-classification task. We will use cross-entropy as a loss function:

$$Loss_{cls}(x) = \sum_{i=0}^{K} -p(x) \log q(x)$$

where $p(x)$ is probability of object x to be actually labelled with i, $q(x)$ is probability of object x to be labelled with i in prediction.

2.3 Metrics

To measure how good our model is capable to divide our into fragments we will use BCubed[3] measurements. Let S be our true segmentation and $\hat{S}$ is the predicted segmentation. We divide the segments from the true segmentation S and the predicted segmentation $\hat{S}$ each into sets of segments S_i, $i \in \{1..c\}$, and $\hat{S}_j$, $j \in \{1..\hat{c}\}$, where c is the true number of authors, and $\hat{c}$ the predicted number of authors. Let l be a function that maps a segment to its length. If S is a set of segment, $l(S) = \sum_{s \in S} l(s)$. BCubed precision of an item is the proportion of items in its cluster which have the item's label, including itself. In our tasks item is a segment from a document, extracted by model. The overall BCubed precision is the averaged precision of all items in the distribution.

$$P_{\mathrm{B}^3} = \sum_{i=1}^{c} \frac{1}{l(S_i)} \sum_{j=1}^{\hat{c}} \sum_{s \in S_i} \sum_{\hat{s} \in \hat{S}_j} l(s \cap \hat{s})^2,$$

$$R_{\mathrm{B}^3} = \sum_{j=1}^{\hat{c}} \frac{1}{l(\hat{S}_j)} \sum_{i=1}^{c} \sum_{s \in S_i} \sum_{\hat{s} \in \hat{S}_j} l(s \cap \hat{s})^2,$$

Also, we would like to measure whether our model detects a segment of different origin as a whole or in several segments. Let it be a granularity[13] of model:

$$gran(S, R) = \frac{1}{|S_R|} \sum_{s \in S_R} |R_S|,$$

where $S_R \subseteq S$ - cases of proved chmage of author in R, and $R_S \subseteq R$ are detections -alleged changes of author in s.

3 Computational Experiment

The goal of computational experiment is to analyse how good baseline solution in classifying fragments of text. RUATD competition, which provided the dataset, also provided accuracy score for their baseline solutions. One solutuion uses tf-idf + logistic regression approach, the other solution uses fine-tuned BERT. We will use the second approach, but our own solution and compare our results with the ones, suggested by the authours of competition.

3.1 Dataset

To test how good our model in text segmentation we generate a dataset of 10000 documents, each of documents consists of 5-6 fragments of different origin. Human-written fragments were taken from Wikipedia Dataset [1]. Machine-generated texts were generated by Sberbank-AI model[2].
For our experiment with classification we took the data, that was collected for RUATD 2022 competition[2] for multitask classification. The test dataset contains 215105 short fragments, that were generated by different 13 models. Authors claim that various language models fine-tuned on different tasks: machine translation, paraphrasing, summarization, simplification and unconditional text generation - were used to generate texts. Moreover, the part of the set was annotated automatically by different generative models. Among models there are several versions of ruGPT, mT5 and ruT5 and M-BART.

3.2 Configuration of algorithm run

We used only a part of dataset for the sake of decreasing the time of training.

Our pipeline for extracting segments from the document consists of tokenization of sentences, basic feature extraction, feature transofrmation and clustering.

Feature extraction was conducted with a sliding window, that includes context of each token. The size of sliding window was 120 tokens. Among features, that were extracted, were stop words counts, Bag of Words, chracter tri-gram counts. After that we tranformed these features with linear layer. For optimization of this step we used RMSProp. On Figure 3 one can see the loss over sample size during training. Clustering was done with k-means.

For tokenization of sentences in text fragments we used pretrained embedding from `bert-base-multilingual-cased`[3]. The same model was fine-tuned for our sequence classification task. We trained our model for 3 epochs. For optimization we used AdamW optimization algorithm[12].

3.3 List of figures and tables

1. Table of train accuracy and validation accuracy on each epoch
2. Graph of validation / train loss versus epoch - Figure 1
3. Graph of Precision-Recall Curve for every class - Figure 2
4. Graph of Loss on Feature Tranformation during text segmentation - Figure 3

[1] https://huggingface.co/datasets/wikipedia

[2] https://huggingface.co/sberbank-ai/rugpt3small_based_on_gpt2

[3] https://github.com/google-research/bert/blob/master/multilingual.md

4 Preliminary report

We received 56% of accuracy with 3 epoch of training only using the part of the dataset. It is les than the score, provided by the authors of competition, which is 59%, but this difference is explained by smaller size of the dataset. Also we may suggest that due to increasing loss on validation set, our model started to overfit. Another issue with the quality of classification on some classes: for some classes there's a small recall, e.g class 6. For some classes there's both small recall and small precision, e.g class 3, class 12 and class 13. For some of class it is happened because of small amount of these classes in dataset.

Table 1: Table of train accuracy and validation accuracy on each epoch

Number of epoch	Train. loss	Valid.loss	Valid. accuracy
1	1.30	1.26	0.54
2	1.07	1.27	0.56
3	0.86	1.35	0.56

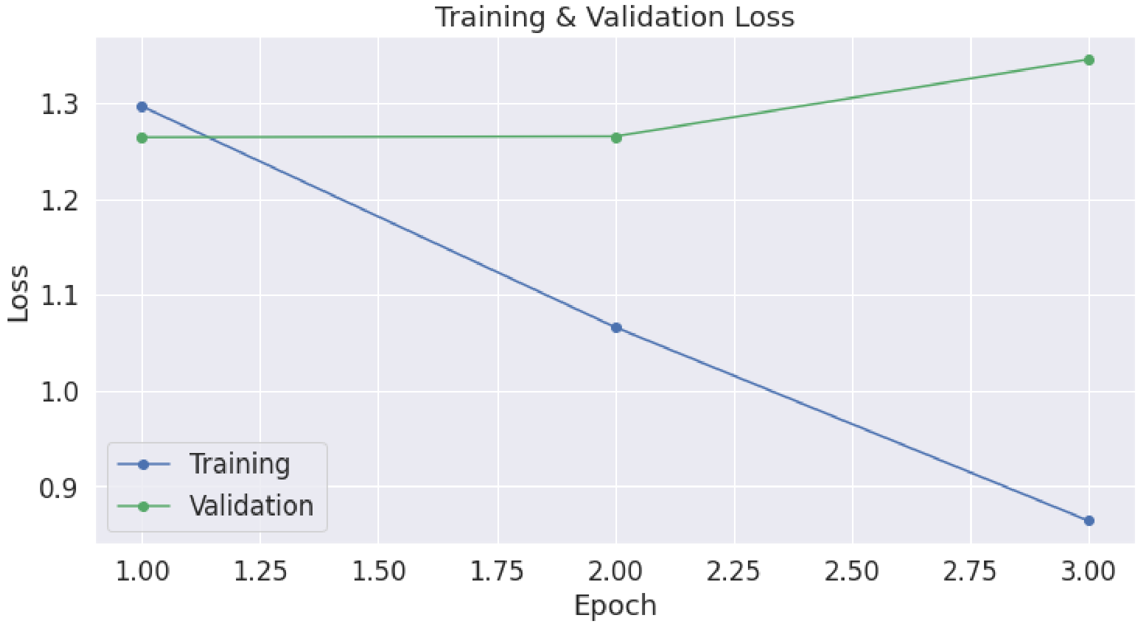

Figure 1: Graph of validation / train loss versus epoch

References

[1] Gpt-2 ouput dataset.

[2] Ruatd multitask dataset.

[3] Enrique Amigó, Julio Gonzalo, Javier Artiles, and M. Verdejo. Amigó e, gonzalo j, artiles j et ala comparison of extrinsic clustering evaluation metrics based on formal constraints. inform retriev 12:461-486. *Information Retrieval*, 12:461–486, 2009.

[4] Daria Beresneva. Computer-generated text detection using machine learning: A systematic review. volume 9612, pages 421–426, 2016.

[5] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher

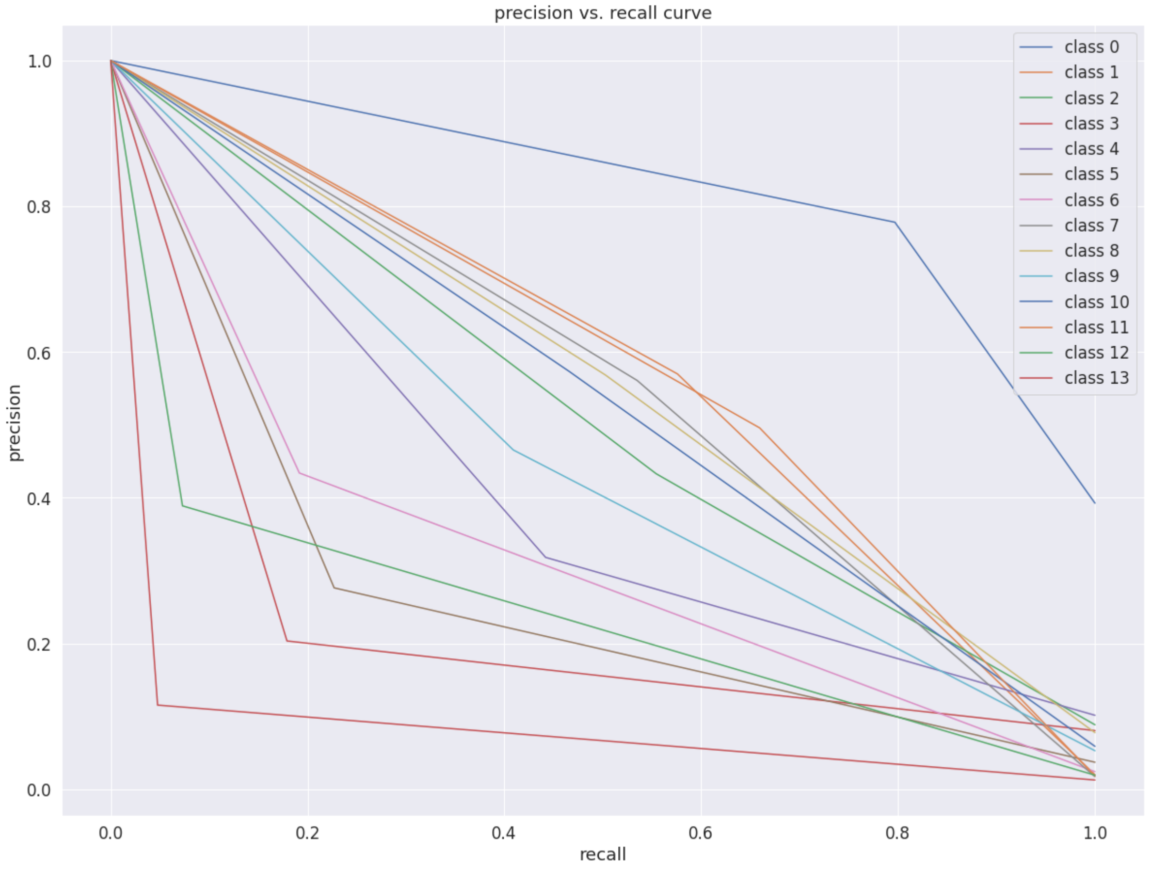

Figure 2: Graph of Precision-Recall Curve for every class

Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. 2020.

[6] Sebastian Gehrmann, Hendrik Strobelt, and Alexander Rush. Gltr: Statistical detection and visualization of generated text. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 111–116, 2019.

[7] Kustarev A.A. Raigorodsky A.M Grechnikov E.A., Gusev G.G. Detection of artificial texts. 20089.

[8] German Gritsay, Andrey Grabovoy, and Yury Chekhovich. Automatic detection of machine generated texts: Need more tokens. pages 20–26, 2022.

[9] Nitish Shirish Keskar, Bryan McCann, Lav R. Varshney, Caiming Xiong, and Richard Socher. Ctrl: A conditional transformer language model for controllable generation. 2019.

[10] Mikhail P. Kuznetsov, Anastasia Motrenko, Rita Kuznetsova, and Vadim V. Strijov. Methods for intrinsic plagiarism detection and author diarization. In *Conference and Labs of the Evaluation Forum*, 2016.

[11] Thomas Lavergne, Tanguy Urvoy, and François Yvon. Detecting fake content with relative entropy scoring. In *Pan*, 2008.

[12] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2017.

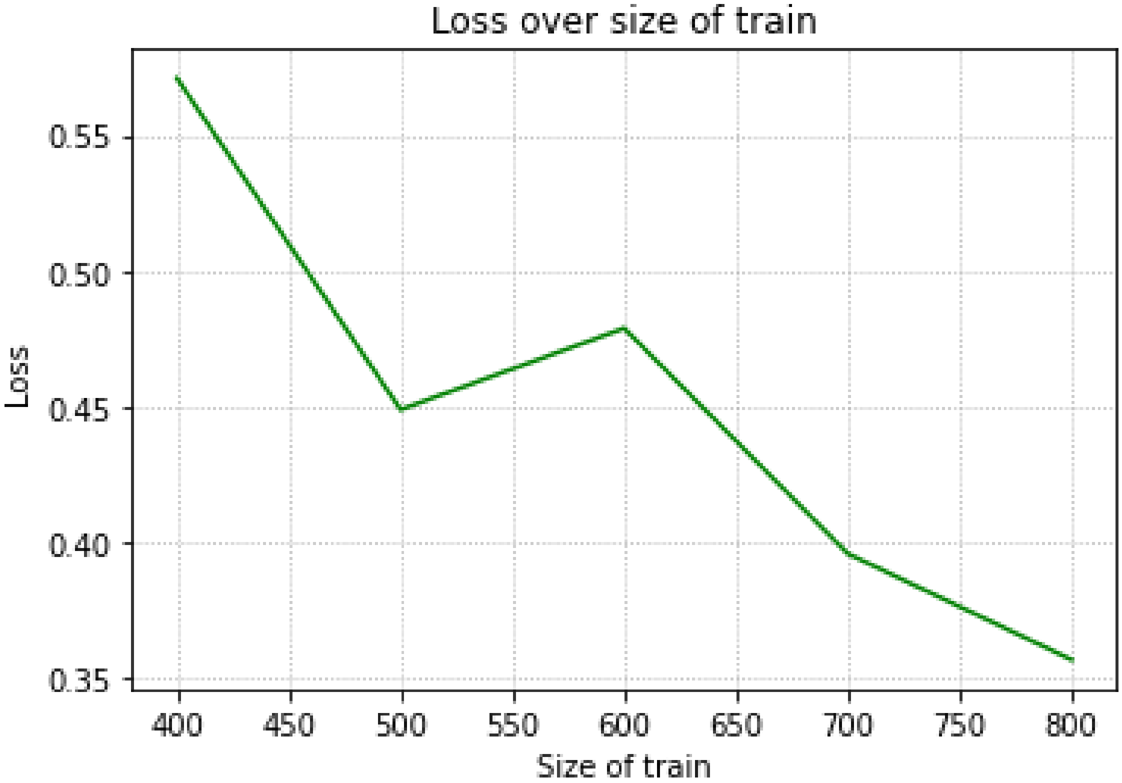

Figure 3: Graph on Loss Function

[13] Martin Potthast, Benno Stein, Alberto Barrón-Cedeño, and Paolo Rosso. An evaluation framework for plagiarism detection. In *Coling 2010: Posters*, pages 997–1005. Coling 2010 Organizing Committee, 2010.

[14] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. 2018.

[15] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

[16] Irene Solaiman, Miles Brundage, Jack Clark, Amanda Askell, Ariel Herbert-Voss, Jeff Wu, Alec Radford, Gretchen Krueger, Jong Wook Kim, Sarah Kreps, Miles McCain, Alex Newhouse, Jason Blazakis, Kris McGuffie, and Jasmine Wang. Release strategies and the social impacts of language models. 2019.

[17] Ashish Vaswani, Noam M. Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. 2017.

[18] Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. mt5: A massively multilingual pre-trained text-to-text transformer. 2020.