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 distinguishes 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, GPT, XLNet ans etc.

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 [10], GPT-2[11], GPT-3 [2], CTRL [6] and mT5 [15]. The problem of detecting machine-generated text has become more relevant recently due to the released ChatGPT[1] model by OpenAI. This model was trained 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 [8]. Also perplexity [1] of the text, frequency of rare bigrams [4] or value of tf-idf [12] can be taken into account. Another approach is to use classifiers that try to label given texts [7].

For a long period recursive neural networks like LSTM were showing best results at solving the detection problem. In 2018 the mechanism of self-attention [13] was introduced and transformers have become new state-of-the-art approach. Every new model has its own

[1] https://openai.com/blog/chatgpt

features and is usually trained on larger amounts of data, but attention mechanism always stays. These model can be used in detection of machine-generated text in two ways [12]. The first method is using a language model that searches for artefacts from methods, which most models are using for generating concise texts [3]. Additional training on new data is not required. The second method is based on fine-tuning on detection task. We fine-tune a language model to "detect itself", using some stochastic methods, for example top-k, top-p sampling [5].

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 [7] on plagiarism detection. We measure the quality of our model using metrics from text segmentation and plagiarism detection problems[9]. 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, if they are machine-generated text. It can be done with transformer-based models, similar to [5]. 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[2] and RUaTD competition dataset[3].

2 Problem statement

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

Let

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

be the space of our documents.

We have a 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 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.

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

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

where J is a 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] https://github.com/openai/gpt-2-output-dataset

[3] https://www.kaggle.com/competitions/ruatd-2022-multi-task/data

2.1 Text fragmentation

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

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

where s_j is start of the j-th fragment and f_j is 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 the blocks of text with the same origin are big enough. Thus, we search for style changes on sentence level. Style change indicates that several next sentences starting from the beginning of the style change have different origin. Each sentence receives label **0** or **1**, where **0** means it is a human-written sentence and **1** is a machine-generated sentence in case of binary classification, and corresponding K labels in case of multiclass classification. 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 documents.

We extract features from sentences in the document and get feature vectors. Our goal is to cluster groups of fragments of the same author, thus we need to maximise the clusterability of the feature vectors. Let $(v_i)_{i=1}^n$ be a sequence of basic feature vectors with elements from $\mathbb{R}^{n_v}$. We apply a transformation to these vectors to ease the process of following clustering of these vectors.

$$T : \mathbb{R}^{n_\mathrm{v}} \to \mathbb{R}^{n_\mathrm{t}},$$
$$T(x) = \mathbf{W}x,$$

where $\mathbf{W} \in \mathbb{R}^{n_\mathrm{v} \times n_\mathrm{t}}$. Let $(l_i)_{i=1}^n$ be the sequence of true author labels with elements from $\mathbf{C}$.

To optimise the transformation T, we use the center loss function[14]:

$$L_\mathrm{c} = \sum_{c=0}^{K+1} \frac{1}{N_c} \sum_{i=1}^{n} \|T(v_i) - \mu_c\|^2 [l_i = c], \tag{1}$$

where $N_c = \sum_{i=1}^{n} [l_i = c]$ is the number of sentences labeled with author a and μ_c represent the current centroid of the transformed feature vectors associated with the author label $c \in \mathbf{C}$:

$$\mu_c = \frac{1}{N_c} \sum_{i=1}^{n} T(v_i)[l_i = c]. \tag{2}$$

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 Performance criteria

To measure how good our model is capable to divide our documents into fragments we will use metric, used in PAN Competition[9] 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. 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.

$$prec(S, R) = \frac{1}{|R|} \sum_{r \in R} \frac{|\bigcup_{s \in S}(s \cap r)|}{|r|},$$

$$rec(S, R) = \frac{1}{|S|} \sum_{s \in S} \frac{|\bigcup_{r \in R}(s \cap r)|}{|s|},$$

Another metric is granularity, which measures whether our model detects a segment of different origin as a whole or in several segments.

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

where $S_R \subseteq S$ denotes proved change of author in R, and $R_S \subseteq R$ are alleged changes of author in s.

3 Computational Experiment

The goal of computational experiment is to analyse how good our solution is at segmenting fragments of text and classifying them.

3.1 Dataset

For binary segmentation we generate a dataset of 10000 documents. There were a following process: first, we took the Medium dataset with articles from Medium.com. This dataset was chosen as the text in it has different styles and if we train a model on them, it will be aware of wide range of styling human writing. Then, we cropped the articles to the length of 4000 tokens. For every document, we randomly picked up to 3 paragraphs to replace by machine-generated fragments. For machine-generation we took LLaMa-7b model, as it produced the most coherent and on-topic generation. For every chosen paragraph we would give the previous to it paragraph to model as prompt. After generating an artificial paragraph we would crop it, so its length was no longer than 700 tokens and put it in the place of original paragraph in the document. Our dataset consists of 4000 documents with 3 replaced paragraphs, 2500 documents of 4 replaced paragraphs and 3500 documents with 2 replaced paragraphs.

3.2 Baseline Experiment

As a baseline experiment we've decided to work on paragraph levels. It is a quite strong prior belief, that in the document each paragraph has only one author and we will weaken this condition further. We created a dataset with all the paragraphes from the documents from our dataset and assigned a label to each paragraph, where 0 is machine-generated and 1 is human-written. After that we've fine-tuned RoBERTa-XLM model on our data. In

order to do the clustering we took the hidden states of [CLS] token for each paragraph and clustered them. We used the cosine distance metric. As it was expected, we recieved two clusters with high negative cosine values in one cluster and high positive cosine values in another clusters. So, the model was very convinced in the classification.

Then, we've done the same procedure, but on the sentence-level. And here, as expected, the model wasn't so convinced, and a lot of sentences received cosine values as 0.3 or -0.10, which hardened the process of clustering.

4 Further Work

References

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

[2] 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 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.

[3] 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.

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

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

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

[7] 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.

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

[9] 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.

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

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

[12] 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.

[13] 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.

[14] Yandong Wen, Kaipeng Zhang, Zhifeng Li, and Yu Qiao. A discriminative feature learning approach for deep face recognition. volume 9911, pages 499–515, 10 2016.

[15] 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.