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 [14], GPT-2[15], GPT-3 [2], CTRL [7] and mT5 [20]. 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 [10]. Also perplexity [1] of the text, frequency of rare bigrams [5] or value of tf-idf [16] can be taken into account. Another approach is to use classifiers that try to label given texts [8].

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 [18] 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 [16]. The first method is using a language model that searches for artefacts from methods, which most models are using for generating concise texts [4]. 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 [6].

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 [8] on plagiarism detection. We measure the quality of our model using metrics from text segmentation and plagiarism detection problems[13]. 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 [6]. 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

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

be the space of our documents.

We have 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.

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

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

where J 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.

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

where $\mathbf{W} \in \mathbb{R}^{n_\text{v} \times n_\text{t}}$. 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[19]:

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

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[13] 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

Our task of text segmentation of subdivided by 2 task. The first one is binary classification and the second one is multiclass classification.

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, each of documents consists of 5-6 fragments of different origin, either human-authored or machine-authored. Human-written fragments were taken from Wikipedia Dataset [4].Machine-generated texts were generated by GPT-2 model[5].

For multiclass segmentation we separately generated datasets of 2000 documents with fixed number of language models, participating in creating those fragments. This number varies from 3 to 5. Machine-generated texts for multiclass classification problem were generated by 8 models as described in [17]. Statistics for each model and percent of its presence in texts is described in Appendix 1.

Further classification is done on the same datasets, that were created for multiclass segmentation, but processed with binary segmentation pipeline.

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

[5] https://github.com/openai/gpt-2-output-dataset

3.2 Configuration of algorithm of segmenting

Pipeline for extracting segments from the document consists of tokenization of sentences, basic feature extraction, feature transformation and clustering.

Parameters are randomly initialized with values from a truncated normal distribution with standard deviation 0.1. It is trained using RMSProp optimizer with learning rate 10^{-5}.

Feature extraction was conducted with a sliding window that includes context of each token. The size of sliding window is 120 tokens. Among features, that were extracted, were stop word counts, Bag of Words, character trigram counts. After that we transformed these features with a linear layer. We trained weight matrix for transformation, using RMSProp as optimizer with learning rate 10^{-5}. Clustering was done with AutoKMeans algorithm from `sklearn` library.

3.3 Configuration of algorithm of classifaction

For tokenization of sentences in text fragments we used pretrained embedding from `bert-base-multilingual-cased`[6]. 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[11].

4 Conclusion

On binary segmentation on 3000 documents we receive following metrics: precision equals 0.8017, recall is 0.4247, granularity is equal 1.4386. Our model is rather good at distinguising human fragments from machine-generated ones, but has a poor quality on finding machine-generating fragments. Granularity is close to 1, which shows, that model is not tend to detect one large segment by smaller parts of it, but rather detects it as a whole fragment.

For problem of multiclass segmentation various metrics, such as precision, recall and granularity can be viewed on Figure 3, Figure 4 and Figure 5. On the contrary, for all possible number of authors our model showed low precision, but high recall. We received the largest recall on 5 authors equal to 0.94, then recall drops to 0.89 for 4 authors and drops again to 0.86 for 3 authors. Granularity equals to 7.5 is extremely high for dataset with 5 authors, which means it tends to extract very small segments, For 4 and 3 authors it does not change very much between each other, but 4.5 is still a large value for given metric.

On task of classifying segments accuracy is equal to 0.20 We can see the decreasing loss for both train and validation at Figure 2, which means we can increase the number of epochs, because model is not saturated with training yet. However, adding new epoch only increases accuracy on 0.01, and with further epochs an overfit is starting to happen.

Below is table with figures on train and validation loss with respect to epoch.

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

Number of epoch	Train. loss	Valid.loss	Valid. accuracy
1	2.20	2.20	0.11
2	2.20	2.17	0.16
3	2.16	2.14	0.20

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

5 Discussion

Our model performs well at task of binary classification, but there are some problems, that need to be figured out and resolved. Also the accuracy on task of classifying the fragments is very low and reasons of it still needed to be researched. In RUATD competition the same pipeline received accuracy equal to 0.5, but on Russian texts.

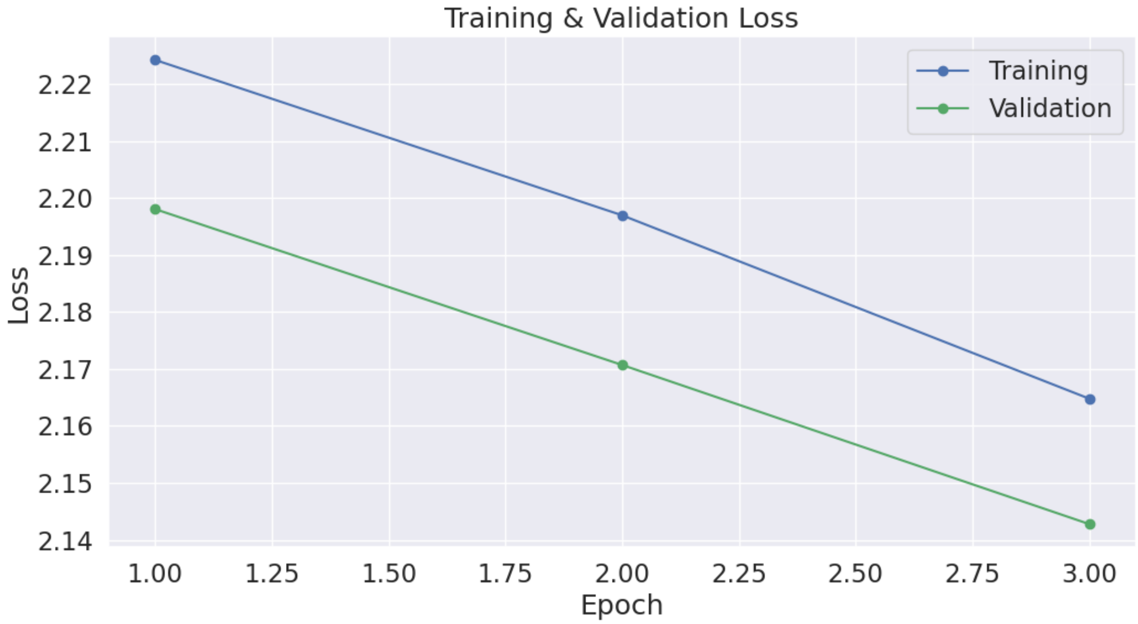

Figure 1: Graph of validation / train loss versus epoch

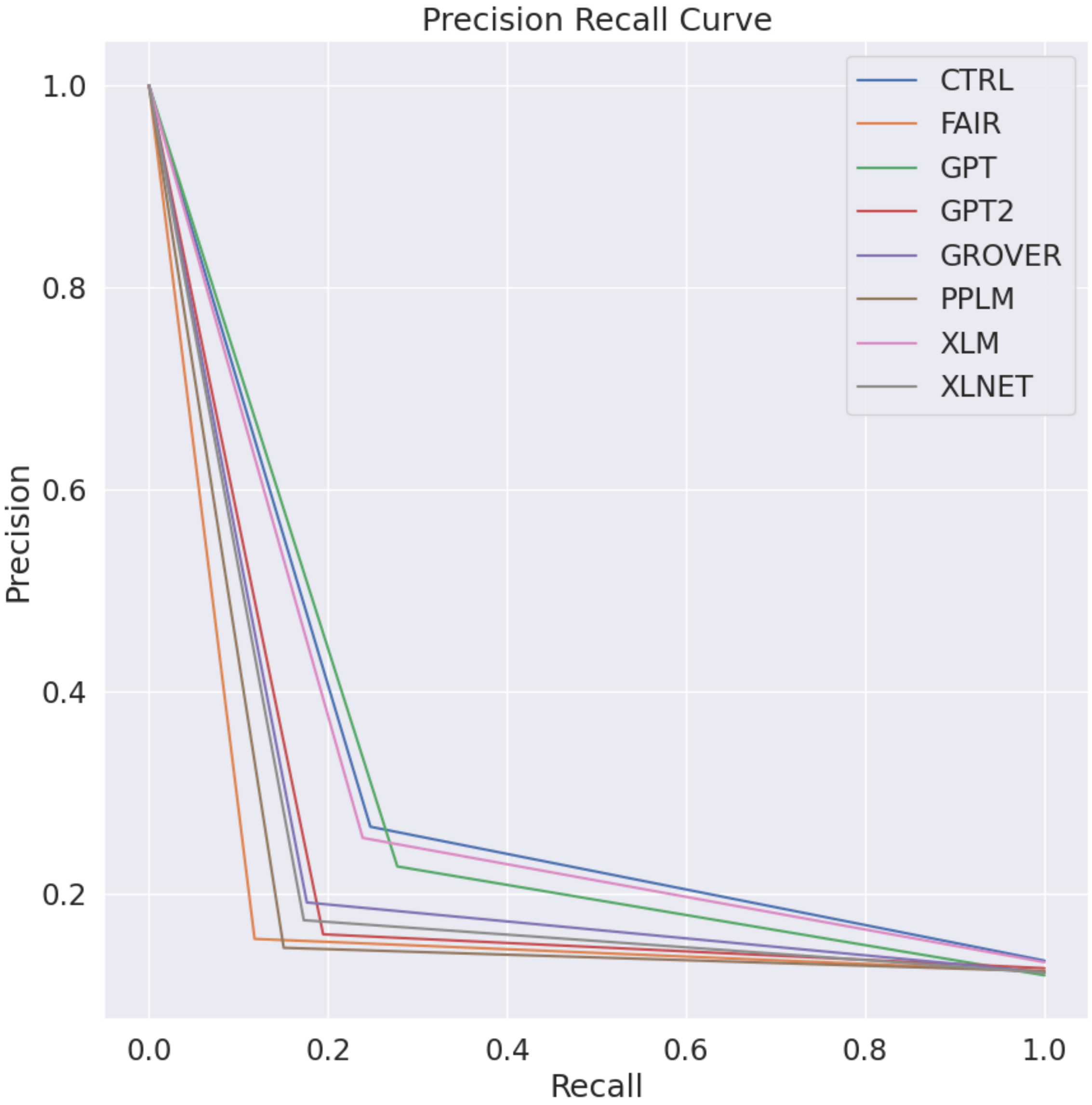

Figure 2: Graph of Precision-Recall Curve for every class

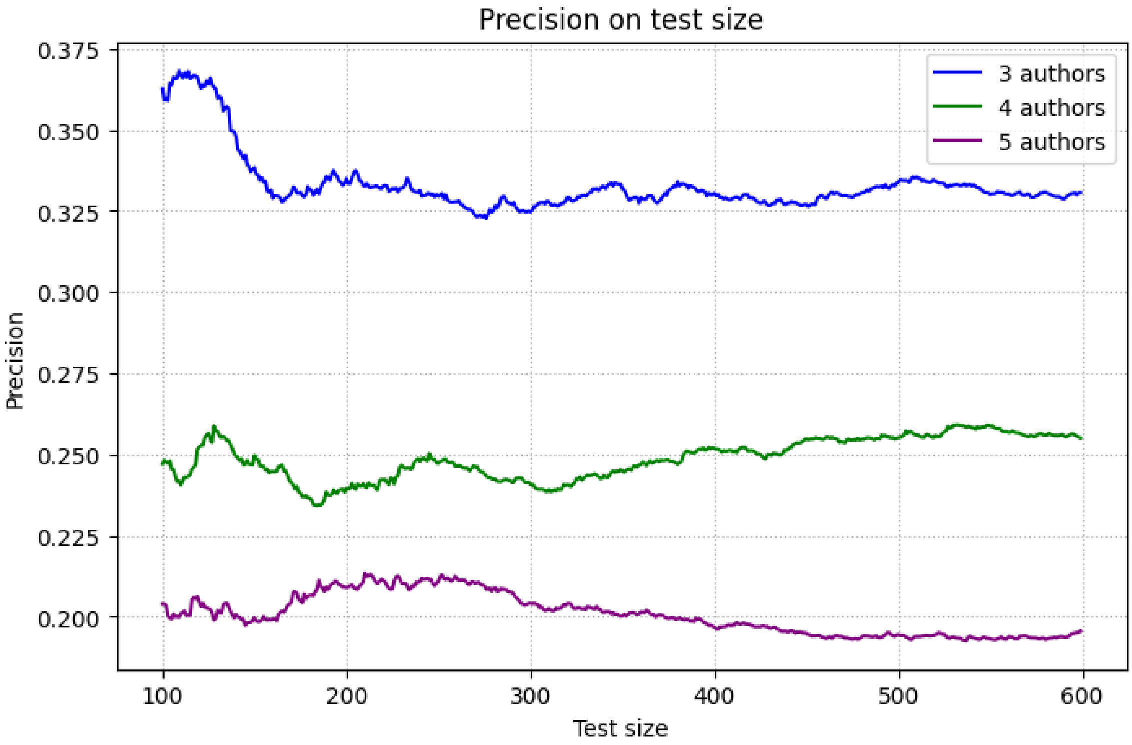

Figure 3: Graph of Precision of Multiclass Fragmenting

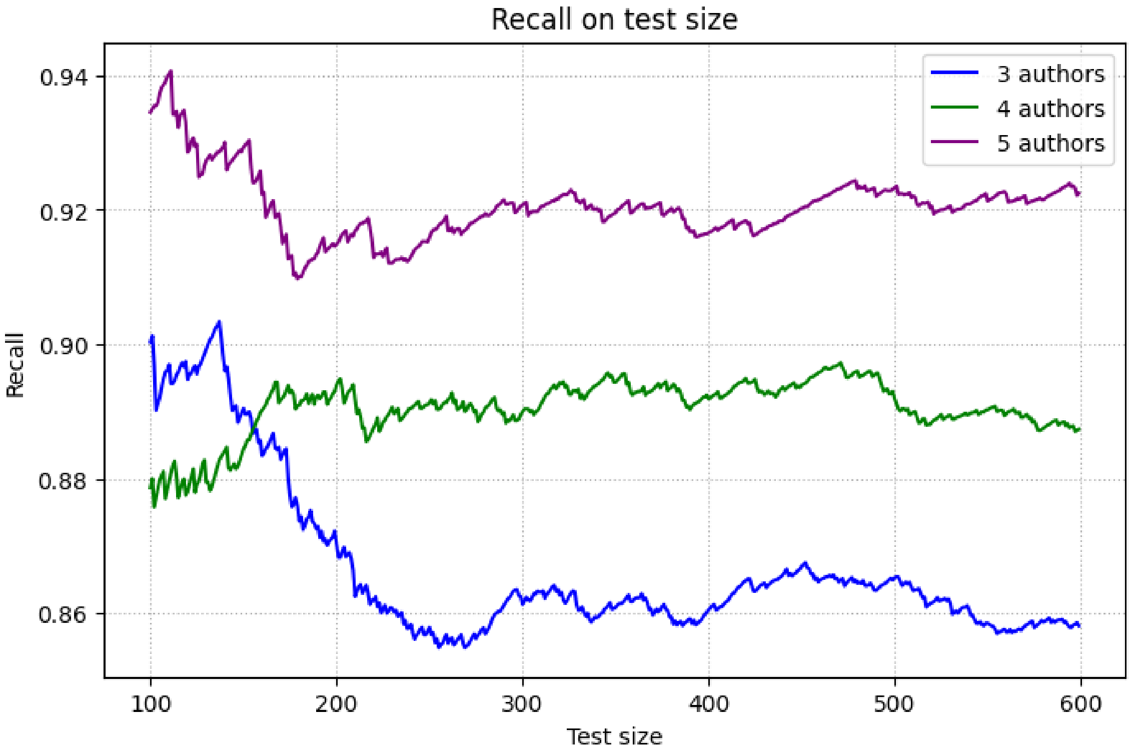

Figure 4: Graph of Recall of Multiclass Fragmenting

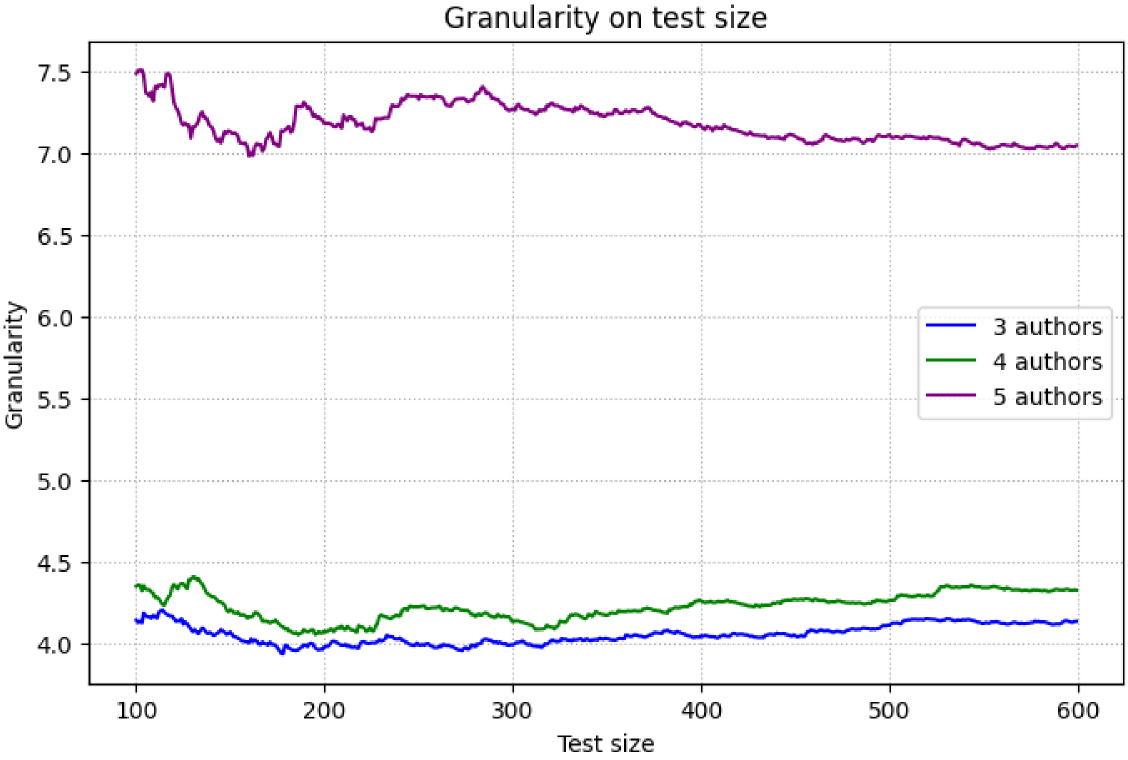

Figure 5: Graph of Granularity of Multiclass Fragmenting

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] Sumanth Dathathri, Andrea Madotto, Janice Lan, Jane Hung, Eric Frank, Piero Molino, Jason Yosinski, and Rosanne Liu. Plug and play language models: A simple approach to controlled text generation, 2020.

[4] 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.

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

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

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

[8] 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.

[9] Guillaume Lample and Alexis Conneau. Cross-lingual language model pretraining, 2019.

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

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

[12] Nathan Ng, Kyra Yee, Alexei Baevski, Myle Ott, Michael Auli, and Sergey Edunov. Facebook fair's wmt19 news translation task submission, 2019.

[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] Adaku Uchendu, Thai Le, Kai Shu, and Dongwon Lee. Authorship attribution for neural text generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8384–8395. Association for Computational Linguistics, November 2020.

[18] 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.

[19] 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.

[20] 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.

[21] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. Xlnet: Generalized autoregressive pretraining for language understanding, 2020.

[22] Rowan Zellers, Ari Holtzman, Hannah Rashkin, Yonatan Bisk, Ali Farhadi, Franziska Roesner, and Yejin Choi. Defending against neural fake news, 2020.

A Multiclass Segmentation

In the task of segmenting text with fragments generated by several language models we used 8 models: XLNet [21], GPT-2 [15], PPLM [3], GPT [14], GROVER [22], FAIR [12], XLM [9], CTRL [7].

We generate documents with 3, 4, 5 different authors, as we want each document to contain only 5-6 fragments. For every number of authors we create 2000 documents and for each document we sampled 3 participants and then 6 times sample author of next fragment. If on some step we sample the same author from previous step, we resample it until received different author.

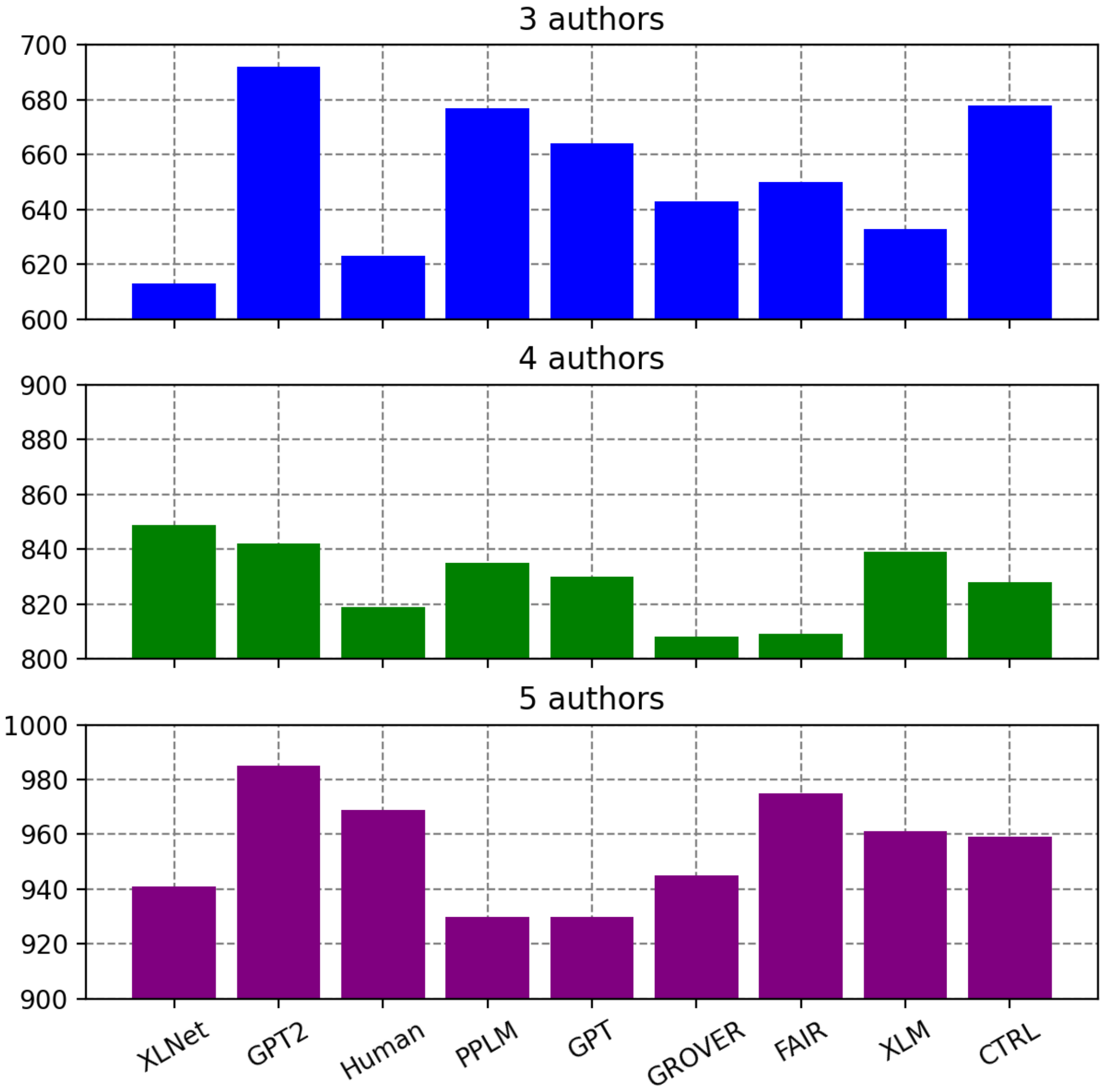

Figure 6: Statistics on language models appearing as authors in generated files

We can see the least differences among authors' occurrences is for 4 authors, while the largest is for 3 authors.

B List of figures and tables

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

2. Validation / train loss versus epoch - Figure 1
3. Precision-Recall Curve for every class - Figure 2
4. Precision of Multiclass Fragmenting - Figure 3
5. Recall of Multiclass Fragmenting - Figure 4
6. Granularity of Multiclass Fragmenting - Figure 5
7. Statistics on language models as authors occurrence in generated files - Figure 6