Fingerprinting Fine-tuned Language Models in the Wild*

Nirav Diwan[1], Tanmoy Chakravorty[1], and Zubair Shafiq[2]

[1]IIIT-Delhi, India
[2]University of California, Davis
[1]{nirav17072,tanmoy}@iiitd.ac.in
[2]zubair@ucdavis.edu

Abstract

There are concerns that the ability of language models (LMs) to generate high quality synthetic text can be misused to launch spam, disinformation, or propaganda. Therefore, the research community is actively working on developing approaches to detect whether a given text is organic or synthetic. While this is a useful first step, it is important to be able to further fingerprint the author LM to attribute its origin. Prior work on fingerprinting LMs is limited to attributing synthetic text generated by a handful (usually < 10) of pre-trained LMs. However, LMs such as GPT2 are commonly fine-tuned in a myriad of ways (e.g., on a domain-specific text corpus) before being used to generate synthetic text. It is challenging to fingerprinting fine-tuned LMs because the universe of fine-tuned LMs is much larger in realistic scenarios. To address this challenge, we study the problem of large-scale fingerprinting of fine-tuned LMs in the wild. Using a real-world dataset of synthetic text generated by 108 different fine-tuned LMs, we conduct comprehensive experiments to demonstrate the limitations of existing fingerprinting approaches. Our results show that fine-tuning itself is the most effective in attributing the synthetic text generated by fine-tuned LMs.

1 Introduction

Background & motivation. State-of-the-art language models (LMs) can now generate long, coherent, and grammatically valid synthetic text (Devlin et al., 2019; Radford et al., 2018, 2019; Brown et al., 2020). On one hand, the ability to generate high quality synthetic text offers a fast and inexpensive alternative to otherwise labor-intensive useful applications such as summarization and chat bots (Yoo and Jeong, 2020; Yu et al., 2020; Wang et al., 2019; Liu and Lapata, 2019). On the other hand, such high quality synthetic text can also be misused by bad actors to launch spam, disinformation, or propaganda. For example, LMs such as Grover (Zellers et al., 2019) are shown to be capable of generating full-blown news articles, from just brief headlines, which are more believable than equivalent human written news articles. In fact, prior work has shown that humans cannot distinguish between organic (i.e., human written) and synthetic (i.e., generated by LM) text (Ippolito et al., 2020; Jawahar et al., 2020; Munir et al., 2021). Thus, this ability to generate high quality synthetic text can further be misused for social impersonation and phishing attacks because users can be easily misled about the authorship of the text.

Problem statement. To mitigate the potential misuse of LMs, the research community has started developing new text attribution techniques. However, as shown in Figure 1, the attribution of synthetic text is a multistage problem. The first step is to distinguish between organic and synthetic text (P1). Prior work has used the LM's output word probability distribution to detect synthetic text (Ippolito et al., 2020; Gehrmann et al., 2019; Zellers et al., 2019). However, there are several publicly available pre-trained LMs that might be used to generate synthetic text. Thus, the second step is to detect the pre-trained LM used to generate synthetic text (P2). Prior approaches showed promising results by attempting to fingerprint the LM based on its distinct semantic embeddings (Pan et al., 2020; Uchendu et al., 2020). However, pre-trained LMs such as GPT2 (Radford et al., 2019) are commonly fine-tuned before being used to generate synthetic text. Thus, the third step is to detect the fine-tuned LM used to generate synthetic text (P3). To the best of our knowledge,

*Supplementary contains dataset, source code, and an appendix (including hyper-parameter setting and additional results). The code and dataset are available at https://github.com/LCS2-IIITD/ACL-FFLM.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 4652–4664

prior work lacks approaches to effectively fingerprint fine-tuned LMs.

Technical challenges. It is particularly challenging to fingerprint fine-tuned LMs simply because of the sheer number of possible fine-tuned variants. More specifically, a pre-trained LM can be fine-tuned in a myriad of ways (e.g., separately on different domain-specific text corpora), resulting in a large number of classes. Another challenge to fingerprint fine-tuned LMs is that we cannot make assumptions about the nature of fine-tuning (e.g., parameters, training data) or generation (e.g., prompts). Prior work on fingerprinting pre-trained LMs is limited to evaluation on a small number of classes (< 10 classes) and on synthetic text that is artificially generated using set prompts.

Proposed approach. To fingerprint synthetic text generated by fine-tuned LMs, we utilize the RoBERTa model (Liu et al., 2019) and attach a CNN-based classifier on top. We fine-tune the RoBERTa model for the downstream task of sentence classification using a synthetic text corpus. The fine-tuned model is used to extract embeddings as features that are then fed to the CNN classifier. We show that the fine-tuned RoBERTa model is able to capture the topic-specific distinguishing patterns of the synthetic text. Upon visualizing the generated features, the samples form closely-condensed distinguishable clusters based on the topic of the organic corpus the LMs have been fine-tuned upon. Therefore, we conclude that fine-tuning itself significantly helps fingerprinting a fine-tuned LM. Note that our fingerprinting approach does not assume access to the text corpus used for LM fine-tuning. We only assume access to arbitrary synthetic text generated by fine-tuned LMs.

Dataset. We gather a real-world dataset of synthetic text generated by fine-tuned LMs in the wild. More specifically, we extract synthetic text from the subreddit r/SubSimulatorGPT2. Each of the 108 users on r/SubSimulatorGPT2 is a GPT2 LM that is fine-tuned on 500k posts and comments from a particular subreddit (e.g., r/askmen, r/askreddit,r/askwomen). It is noteworthy that users on r/SubSimulatorGPT2 organically interact with each other using the synthetic text in the preceding comment/reply as their prompt.

Evaluation. We evaluate our models using a suite of evaluation metrics. We also adapt confidence-based heuristics, such as the gap statistic. Our best model is accurate for a large number of classes, across a variety of evaluation metrics, showing impressive results for the largest setting of 108 classes. While it obtains around 46% precision and 43% recall, its top-10 accuracy is about 70%. In other words, the correct class is one of the top-10 predictions in about 70% of the cases. If we give the model the option to not make a classification decision, via confidence estimation, it doubles the precision of the top classification from 46% to around 87% with a decrease of recall from 43% to 27%.

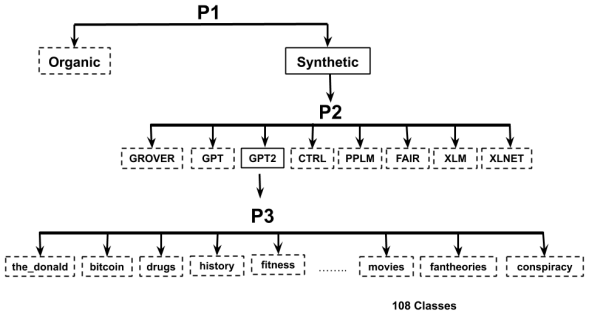


Figure 1: Attribution of text formulated as a series of three problems: P1, P2, and P3.

We summarize our key contributions as follows:

1. **Problem formulation.** To the best of our knowledge, we are the first to explore the problem of attribution of synthetic text generated by fine-tuned LMs (P3). We are also the first to investigate synthetic text attribution for a large number of classes on a real-world dataset. We also show that P3 is a much more challenging problem as compared to P1 and P2.
2. **Comprehensive model.** We design and implement a comprehensive set of different feature extraction techniques. We use them on a variety of machine learning and deep learning pipelines to build detection models.
3. **Rigorous evaluation.** We conduct rigorous evaluation on a real-world dataset of synthetic text generated by fine-tuned LMs. We use several evaluation metrics such as top-k accuracy and precision-recall tradeoff to compare different detection models. We also provide insights into the performance of different feature sets and classification algorithms.

Paper Organization: The rest of the paper is organized as follows. Section 2 contextuailzes our work with respect to prior literature. We analyze the real-world dataset of synthetic text generated

by fine-tuned LMs in Section 3. Section 4 describes different feature sets and classification algorithms for fingerprinting fine-tuned LMs. Section 5 presents the results of our experimental evaluation before concluding in Section 6.

2 Related Work

Figure 1 illustrates three different problem formulations for attribution of synthetic text generated by LMs. The first line of research (**P1**) aims to distinguish between organic (by humans) and synthetic (by LMs) text. Given synthetic text, the second line of research (**P2**) further aims to attribute the synthetic text generated by pre-trained LMs such as BERT and GPT. Finally, in this paper, we further aim to (**P3**) attribute the synthetic text generated by fine-tuned LMs. Here, we first discuss prior work on **P1** and **P2** and then highlight the importance and unique challenges of **P3**.

P1: As the quality of synthetic text generated by LMs has improved, the problem of distinguishing between organic and synthetic text has garnered a lot of attention. Gehrmann et al. (2019) aimed to distinguish between synthetic text generated by GPT2 and Heliograf versus organic text by books, magazines, and newspapers. They showed that humans had a hard time classifying between organic and synthetic. Their proposed GLTR model, which uses probability and ranks of words as predicted by pre-trained LMs as features, was able to achieve 87% AUC. Zellers et al. (2019) developed Grover, a LM to generate fake news. They also showed that humans had a hard time distinguishing between organic and synthetic text generated by Grover. A classifier based on Grover achieved near perfect accuracy and significantly outperformed other classifiers based on pre-trained LMs. Ippolito et al. (2020) presented an interesting trade-off between distinguishability of organic and synthetic text. They showed that synthetic text optimized to fool humans is actually much easily detected by automated classification approaches. They generated synthetic text from pre-trained GPT2 using different sampling strategies and parameters, and used different classifiers such as GLTR that use pre-trained LMs and a purpose-built fine-tuned BERT based classifier. They showed that their fine-tuned BERT based classifier was able to significantly outperform other approaches as well as humans.

P2: Given synthetic text, recent work has further attempted to attribute authorship of synthetic text generated by pre-trained LMs. Uchendu et al. (2020) fingerprinted pre-trained LMs by attributing synthetic text to its author LM. They conducted exhaustive experiments using conventional authorship attribution models (Kim, 2014; Zhang et al., 2015; Cho et al., 2014) for eight pre-trained LMs – GPT (Radford et al., 2018), GPT2 (Radford et al., 2019) , GROVER (Zellers et al., 2019), FAIR (Ng et al., 2019), CTRL (Keskar et al., 2019), XLM (Conneau and Lample, 2019), XLNET (Yang et al., 2019), and PPLM (Dathathri et al., 2020). They showed that derived linguistic features when used with simple classifiers (Random Forest, SVM) perform the best. Pan et al. (2020) prepared a corpus of organic text and queried each of the five LMs – BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019), GPT, GPT2 and XLNET, to generate pre-trained embeddings. Then, they trained a multilayer perceptron using the embeddings and obtained perfect accuracy in fingerprinting the pre-trained LM. Munir et al. (2021) used stylometric features as well as static and dynamic embeddings using ML classifiers to attribute synthetic text generated by four pre-trained LMs – GPT, GPT2, XLNet, and BART (Lewis et al., 2020). They obtained near perfect accuracy in fingerprinting pre-trained LMs on a purpose-built dataset of synthetic text.

P3: In this paper, we further attempt to attribute authorship of synthetic text generated by fine-tuned LMs. This problem is relevant in the real-world because malicious actors typically fine-tune a pre-trained LM for domain adaption (e.g., to generate fake news articles vs. food reviews) (Zellers et al., 2019). There are two main novel aspects of **P3** that make it more challenging than **P1** and **P2**. *First*, LMs can be fine-tuned in numerous different ways. More specifically, an attacker can use various datasets to fine-tune a pre-trained LM and further set the fine-tuning parameters and epochs in many different ways. Therefore, the size of the universe of fine-tuned LMs is expected to be quite large in **P3**. In **P1**, the problem is a simple binary classification problem (organic vs. synthetic). Similarly, in **P2**, the problem has still limited number classes because only a handful of pre-trained LMs are publicly available (e.g., GPT, GPT2, etc.). For instance, Munir et al. (2021), Pan et al. (2020), and Uchendu et al. (2020) respectively considered a universe of four, five, and

eight pre-trained LMs. In contrast, in this paper, we consider 108 fine-tuned LMs. *Second*, prior work mostly considered attributing authorship of synthetic text generated in a controlled setting. For example, synthetic text is often generated by providing a topical prompt (Uchendu et al., 2020). As another example, the attribution classifier often assumes some information about the training data (Zellers et al., 2019). In contrast, here we consider the problem of attributing synthetic text in an uncontrolled setting assuming no control or knowledge about training or generation phases of fine-tuning LM.

3 Dataset

Prior work on detection and attribution of synthetic text has relied on artificial purpose-built datasets in a controlled environment. We overcome this issue by gathering a real-world dataset of synthetic text by different GPT2 bots on the R/SUBSIMULATORGPT2. Note that R/SUBSIMULATORGPT2 is independently designed and operated by its moderators. Each user on the r/SUBSIMULATORGPT2 subreddit is a GPT2 small (345 MB) bot that is fine-tuned on 500k posts and comments from a particular subreddit (e.g., r/askmen, r/askreddit, r/askwomen). The bots generate posts on r/SUBSIMULATORGPT2, starting off with the main post followed by comments (and replies) from other bots. The bots also interact with each other by using the synthetic text in the preceding comment/reply as their prompt. In total, the subreddit contains 401,214 comments posted between June 2019 and January 2020 by 108 fine-tuned GPT2 LMs (or class). The complete details of various design choices are described here: `https://www.reddit.com/r/SubSimulatorGPT2Meta/`

Table 1 lists some representative examples of the synthetic text generated by three different fine-tuned GPT2 LMs on our dataset. We note that the synthetic text is fairly coherent and also captures the unique vocabulary and style of the subreddit used for fine-tuning. For example, the excerpt from r/conspiracy reads like a conspiracy discussion, the excerpt from r/confession mentions a suggestive reply to the main post, and the excerpt for r/wallstreetbets uses the specialised finance terms like "puts".

Next, we quantitatively compare and contrast synthetic and organic texts corresponding to different subreddits. To this end, we collect organic text from the corresponding subreddits. Specifically, we randomly sample 1,000 comments from each subreddit class of synthetic and organic text. We contrast basic lexical characteristics, vocabulary, and readability of synthetic and organic text.

Subreddit	Synthetic text excerpt
r/conspiracy	I'm sure the elite have been working to control what they do, but I don't think they have the resources or manpower to control something as massive as the global economy.
r/confession	You need to tell her how you feel! She needs to know how you feel so that you can work out how to make the right decision. You can tell her how much you hate being a coward and how you'd never be able to live up to your promise.
r/wallstreetbets	There's a huge amount of volatility right now that I don't know how well my eyes can handle. I'm not going to buy any puts for the next week.

Table 1: Excerpts of synthetic text generated by GPT2 LMs fine-tuned on different subreddits.

Lexical. First, we contrast the following basic lexical features: average/standard deviation of number of words/sentences per comment. We also measure the Pearson correlation coefficient (ρ) between pairs of synthetic and organic texts in Figures 2a to 2d. We note a high correlation ($\rho > 0.83$) across all lexical features. Thus, we conclude that there is a strong dependency between the lexical characteristics of the synthetic and organic text. This finding indicates that *synthetic text generated by fine-tuned GPT2 models indeed captures the lexical characteristics of the corresponding organic text used for fine-tuning*.

Vocabulary. Second, we compare the vocabularies of synthetic and organic text of each class. We do some basic pre-processing: lower-case, tokenize, and lemmatize all words, remove all punctuation and emojis, and replace hyperlinks and numbers with standard tags. Figure 2e compares the vocabulary size of synthetic and organic text. While organic text seems to have a large vocabulary than synthetic text, we note a strong correlation ($\rho = 0.76$) between their vocabulary sizes. We further compute Jaccard similarity to measure pair-wise vocabulary overlap between different synthetic and organic texts. Figure 3 visualizes the similarity matrices between all pairs classes – synthetic-synthetic, organic-organic, and synthetic-organic. It is noteworthy that the left

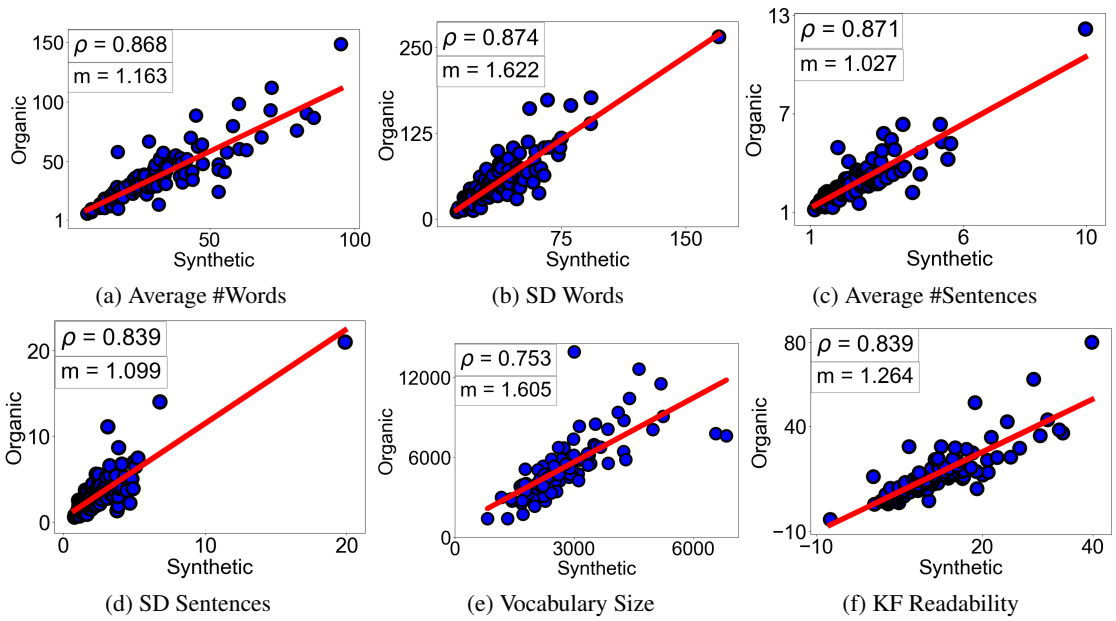


Figure 2: A class-wise comparison of organic and synthetic text indicating strong correspondence in terms of: (a) average number of words; (b) standard deviation of number of words; (c) average number of sentences; (d) standard deviation of number of sentences; (e) vocabulary size; and (f) Kincaid-Flescher readability. ρ is the Pearson correlation coefficient and m is the slope of the linear fit.

diagonal represents much higher overlap between corresponding pairs of synthetic and organic text classes, even across synthetic-organic text classes. This finding indicates that the *fine-tuned GPT2 models indeed pick up the vocabulary from the organic text in the corresponding subreddit*. It is also noteworthy that the average vocabulary overlap among synthetic text classes in Figure 3a is much higher than that among organic text classes as shown in Figure 3b. This indicates that *synthetic text vocabulary combines information from both pre-training and fine-tuning* – the text corpus used to pre-train the base GPT2 model as well as the text corpus from the particular subreddit used to fine-tune it.

Readability. Finally, we compare the readability of the synthetic and organic text. Figure 2f compares the Kincaid-Flescher readability score (Kincaid et al., 1975) of synthetic and organic text. We note that the average readability of synthetic text (16.8) is less than that of organic text (11.5). This observation corroborates the findings in recent prior work (Uchendu et al., 2020). Similar to lexical and vocabulary analysis, we note a strong correlation ($\rho = 0.84$) between the readability of synthetic and organic text.

Additionally, using the organic text as reference, we measure the quality of the synthetic text using well-known metrics – METEOR (Banerjee and Lavie, 2005) and BLEU (Papineni et al., 2002). We observe that the synthetic text achieves high average scores across all the metrics. We include the results in the appendix.

Overall, we have two key takeaways: (1) synthetic text is *coherent* although with lower overall readability than organic text; (2) synthetic text captures the *characteristics of the organic text used to fine-tune it*.

4 Methods

Fingerprinting fine-tuned LMs needs to operate under the following realistic assumptions. (i) Fingerprinting methods cannot assume any knowledge about the nature of the LM, fine-tuning (e.g., parameters, layers) or generation (e.g., prompt). (ii) They also cannot assume any access to the organic text used for LM fine-tuning. (iii) Since a LM can be fine-tuned in a myriad of ways, these methods need to consider a large number of classes. (iv) These methods are assumed to have access to a limited sample of synthetic text generated by each of the potential fine-tuned LMs.

4.1 Pre-processing

We lower cased the comments and replaced all hyperlinks with a standard tag [*LINK*]. Next, we to-

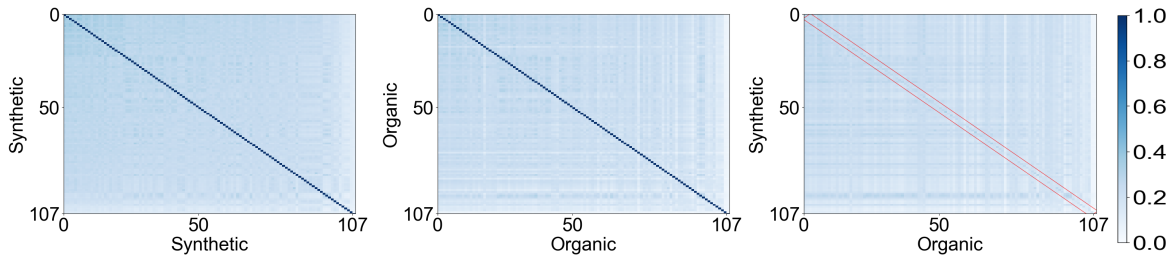


Figure 3: Pair-wise vocabulary overlap between classes of (a) synthetic text, (b) organic text, and (c) cross-comparison of organic and synthetic text. The higher intensity in (a) indicates more vocabulary overlap between classes of (a) synthetic text as compared to (b) organic text. The dark diagonal (highlighted in red box) in (c) indicates significant vocabulary overlap between the synthetic and organic text from the same subreddit.

kenized the comments. Any comment with 5 or fewer tokens was removed. The maximum number of tokens in a comment is limited to 75. In case a comment is larger, only the first 75 tokens in a comment are taken into consideration. This is consistent for all the models we experimented with. The limit was decided based on the limit of the size of the GPU (11GB) used for fine-tuning GPT2 and RoBERTa models.

4.2 Features

We consider several different feature extraction architectures to encode the synthetic text into representation vectors. As we discuss later, these representation vectors are then fed to classifiers.

Writeprints: Writeprints feature set has been widely used for authorship attribution (Iqbal et al., 2008; Pearl and Steyvers, 2012; Abbasi and Chen, 2006). We extract a total of 220 features that include *lexical* (e.g., total number of words, total number of characters, average number of characters per word, digits percentage, upper case characters percentage), *syntactic* (e.g., frequencies of function words, POS tags unigram, bigrams, trigrams), *content-based* (e.g., bag of words, bigrams/trigrams) and *idiosyncratic* (e.g., misspellings percentage) features.

GLTR: Gehrmann et al. (2019) used pre-trained LMs to extract word likelihood features – word ranks and probabilities. We follow the original approach to average the word probabilities of the text based on pre-trained BERT and GPT2. We also bin the word ranks into 10 unequal ranges. The bins are: [1], [2-5], [6-10], [11-25], [26-50], [51-100], [101-250], [251-500], [501-1000], [$>$ 1000].

Glove: Glove embeddings (Pennington et al., 2014) have been commonly used in large-scale authorship attribution (Ruder et al., 2016). We follow (Kim, 2014; Zhang et al., 2015) to create a representation vector of the size (max # tokens $\times$ 100, where max # tokens is set to 75) using Glove.

Pre-trained LMs (GPT2 and RoBERTa): We also extract the embeddings for each comment using the pre-trained GPT2/RoBERTa model. Similar to Nils and Gurevych (2019); Feng et al. (2020); Zhu and de Melo (2020), we take the [CLS] token representation from the last layer to extract the embeddings of size 1×768. The final embeddings are then scaled between the values of [-3, 3] using min-max normalization.

Fine-tuned (FT) LMs (GPT2 and RoBERTa): We add a softmax classification layer to the pre-trained GPT2/RoBERTa model. Then, we fine-tune the LM for the task of sentence classification using the synthetic text in the training dataset. We again extract the embeddings (size = 1×768) by taking the [CLS] token representation from the second last layer. The final embeddings are then scaled between the values of [-3, 3] using min-max normalization.

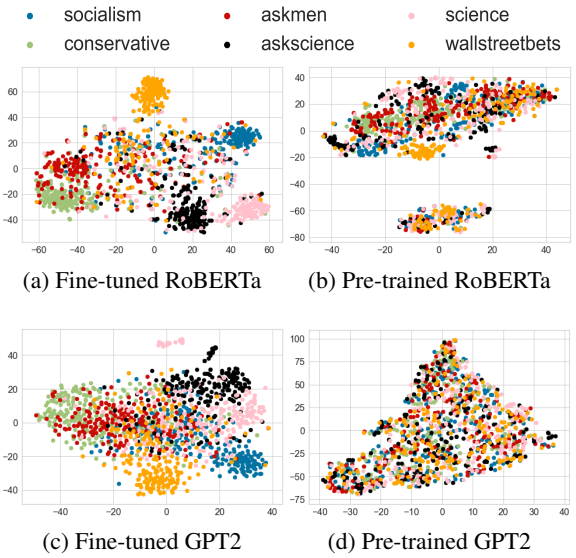
4.3 Classifiers

Shallow classifiers: We use a probabilistic classifier – Gaussian Naive Bayes (GNB), an ensemble decision classifier – Random Forest (RF), and a feed-forward multilayer perceptron (MLP) across all of the feature representations.

CNN classifier: We experiment with a *stacked* CNN model where convolution layers with different kernel sizes are stacked before the embedding layer (Kim, 2014). In this architecture, we attach two stacked convolution 1D layer, followed by a batch normalisation layer and a max pooling layer.

Architecture	Classifier	Macro		Top-k	
		Prec	Recall	5	10
GLTR	GNB	5.5	4.4	12.9	20.9
	RF	7.8	6.6	12.6	19.0
	MLP	3.6	6.3	15.6	23.7
Writeprints	GNB	8.2	5.8	14.1	21.4
	RF	10.2	8.4	14.9	21.8
	MLP	16.9	14.7	30.8	42.1
GloVE	GNB	19.2	9.3	21.9	31.2
	RF	20.5	16.9	27.1	36.2
	MLP	29.7	27.2	44.4	54.1
	CNN	31.1	26.7	44.2	53.5
GPT2	GNB	24.8	12.4	27.8	37.7
	RF	10.5	7.8	15.8	27.1
	MLP	44.9	29.0	47.5	56.9
	CNN	30.9	28.7	49.1	59.1
RoBERTa	GNB	39.2	15.8	30.8	41.0
	RF	11.1	8.4	16.6	25.8
	MLP	44.0	34.8	54.8	62.5
	CNN	33.5	32.0	53.1	63.0
FT-GPT2	GNB	40.1	37.0	56.9	66.0
	RF	27.6	22.8	34.8	45.2
	MLP	40.2	36.4	55.7	64.0
	CNN	44.6	42.1	60.9	68.9
FT-RoBERTa	GNB	**47.7**	41.5	57.9	64.9
	RF	42.0	36.8	46.9	53.2
	MLP	42.8	41.5	58.2	65.3
	CNN	46.0	**43.6**	**62.0**	**69.7**

Table 2: Performance of multi-class classifiers based on macro Precision (Prec), Recall and top-k accuracy ($k = 5, 10$) for the largest setting of 108 classes.



(a) Fine-tuned RoBERTa (b) Pre-trained RoBERTa

(c) Fine-tuned GPT2 (d) Pre-trained GPT2

Figure 4: Visualisation of fine-tuned (a,c) and pre-trained embeddings (b,d) of specific classes. Closely condensed clusters specific to the domain of the organic text form in the fine-tuned embeddings.

The output is then fed to two dense layers, the latter of which is a softmax layer.

In addition, we also experiment with other shallow classifiers (SVM, Decision Tree) and two more types of feature generators (fine-tuned GLTR, trainable word embeddings). We report additional results, observations and the hyperparameters for all the models in the appendix.

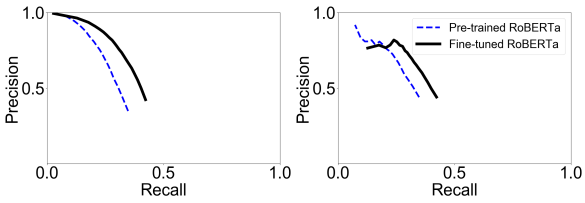
5 Evaluation

We conduct experiments to evaluate these methods using the real-world Reddit dataset as described in Section 3. Our training, validation, and test sets consist of 800, 100 and 200 synthetic comments respectively from each of the 108 subreddit classes. In total, our training, validation and test sets comprise 86k, 11k and 22k comments respectively. For evaluation, we use macro precision and recall. We also measure top-k accuracy based on the confidence score to assess the accuracy of the classifiers in k ($k = 5, 10$) guesses for 108 classes.

5.1 Results

Table 2 lists the results of different feature representations and classifiers. Overall, classifiers based on fine-tuned LM embeddings perform the best, with RoBERTa slightly outperforming GPT2. Fine-tuned embeddings are successfully able to capture the domain of the organic text the LM is fine-tuned on. To provide further insights into our best performing feature representations, we visualize the feature embeddings of pre-trained and fine-tuned RoBERTa and GPT2. Figure 4 plots the 2D projection of synthetic text (using t-SNE) generated by different LMs that are fine-tuned on various subreddits. Fine-tuned embeddings form more cohesive and separated clusters than pre-trained embeddings. Thus, we conclude that fine-tuning these embeddings is beneficial in attributing synthetic text generated by different fine-tuned LMs. Note that certain clusters are more cohesive and better separated than others. For example, the most distinct cluster is observed for r/wallstreetbets in Figures 4a and 4c for fine-tuned embeddings. However, it is not quite distinct in Figures 4b and 4d for pre-trained embeddings. We also note that some clusters with high topical similarity (e.g., r/science and r/askscience) are closer to each other. On the other hand, some clusters with likely lower topical similarity (e.g., r/socialism and r/conservative) are far apart.

Despite combining word probabilities from both BERT and GPT2, GLTR is ineffective. We find that synthetic texts generated from different fine-tuned models have similar word probabilities because perhaps they are more impacted by the pre-training process rather than the subsequent fine-tuning. This shows that the classifier that performs well for distinguishing between organic and synthetic text (P1) does not work well



(a) Micro (b) Macro

Figure 5: (a) Micro and (b) Macro *precision-recall trade-off* by varying the gap statistic threshold. The comparison with *all baselines* is included in the appendix.

for distinguishing between synthetic text by different fine-tuned LMs (P3). Writeprints feature set provides some improvement but is still ineffective. Our finding corroborates Manjavacas et al. (2017), who reported that linguistic and stylistic features as used in Writeprints are not effective in distinguishing between synthetic text. GloVE again offers some improvement over Writeprints but its performance remains significantly worse than that of our best performing method.

Overall, fine-tuned RoBERTa embeddings with CNN performs the best with 46.0% precision and 43.6% recall. In about 44% of the cases, this classifier can correctly fingerprint fine-tuned LM amongst 108 classes; in about 70% of cases the correct prediction is one of the top-10 guesses. It is noteworthy that Random Forest which performed exceedingly well in prior work on fingerprinting pre-trianed LMs (Uchendu et al., 2020), does not perform well for fingerprinting fine-tuned LMs. Surprisingly, a relatively simple classifier like GNB achieves comparable precision and recall for our top guess. However, CNN outperforms GNB by a small margin, achieving the best top-10 accuracy of 69.7% for FT-RoBERTa.

5.2 Discussion

Next, we analyze the performance of our best performing RoBERTa feature representation and classifier (CNN) under different conditions.[1]

Precision-Recall trade-off. We evaluate the precision-recall trade-off by imposing a threshold on the confidence of our prediction. To this end, we use the *gap statistic*, defined as the difference between the probability of the highest and second highest prediction (Narayanan et al., 2012). If the gap statistic is lower than our threshold, the classifier chooses to not make a prediction for the test sample. This invariably has an impact on precision and recall. Typically, the precision of the classifier increases, since it can more accurately predict the correct class for the samples it has a high confidence in. Due to certain samples not being predicted for, recall is expected to decrease. Note that since the classifier may make different number of predictions across classes, micro and macro precision/recall could be different.

Figures 9a and 9b respectively plot the micro and macro precision/recall as we vary the gap statistic. Overall, the classifier using FT-RoBERTa embeddings achieves a better precision-recall trade-off compared to using standard RoBERTa embeddings. As expected, precision improves at the expense of recall for larger values of gap statistic. Micro precision increases 46% to over 87% with a reduction in the micro recall from 43% to 27%. Similarly, despite potential class imbalance, macro precision increases 46% to over 81% with a reduction in the micro recall from 43% to 26%. Thus, we conclude that the confidence of our best performing fingerprinting method can be tuned to achieve very high precision with some compromise on recall.

Impact of number of training samples. Next, we evaluate the impact on the performance of our best models by varying training size from 50 to 800 samples per class. As we vary the training data, we keep the same test set, i.e., 200 samples from each class. Figures 8a and 8b, respectively show that precision and recall of FT-RoBERTa plateau at around 400 samples per class. Despite at twice the training data, using training 800 samples per class has similar precision/recall as compared to using 400 training samples per class. We conclude that having more training samples of synthetic text may not always lead to a significant improvement in fingerprinting performance.

Impact of number of classes. We further vary the number of classes from 10 to 108 and report the performance of our best models. Figures 8c and 8d show that for a 10-class problem, FT-RoBERTa + CNN achieves 63.0% precision and 61.7% recall. As expected, both precision and recall decrease as the number of classes increases. For 108 classes, the same classifier achieves 46.0% precision and 43.6% recall. This indicates that fingerprinting a fine-tuned LM is highly challenging when the universe of potential fine-tuned LMs is larger.

[1] Other baseline results are reported in the appendix.

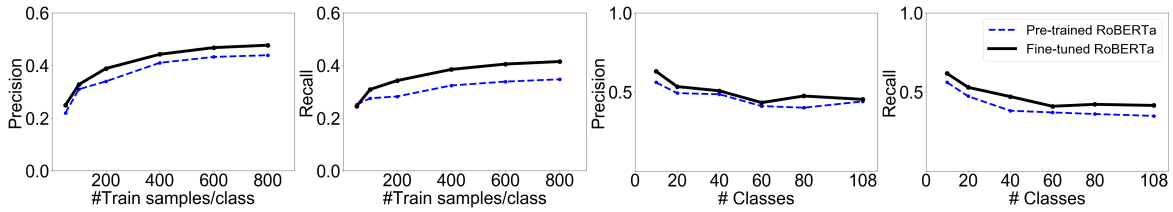


Figure 6: Comparison between the performances of pre-trained and fine-tuned RoBERTa by varying different parameters. (a) Precision and (b) Recall with the varying training size. (c) Precision and (c) Recall with the varying number of classes. Overall, fine-tuned RoBERTa outperforms pre-trained RoBERTa. The comparison with *all baselines* is included in the appendix.

6 Conclusion

In this paper, we studied the problem of attribution of synthetic text generated by fine-tuned LMs. We designed a comprehensive set of feature extraction techniques and applied them on a number of different machine learning and deep learning pipelines. The results showed that the best performing approach used fine-tuned RoBERTa embeddings with CNN classifier. Our findings present opportunities for future work on fingerprinting LMs in even more challenging open-world scenarios, where the list of potential LMs might be incomplete or synthetic text is not available to train attribution classifiers.

Ethics and Broader Impact

Any biases found in the gathered dataset are unintentional, and we do not intend to do harm anyone. We would also like to acknowledge that the use of large-scale transformer models requires non-trivial compute resources (GPUs/TPUs) that contributes to global warming (Strubell et al., 2019). However, we did not train the models from scratch and simply fine-tuned them for our work.

Acknowledgement

The work was partially supported by the Ramanujan fellowship and the Infosys Centre for AI, IIIT-Delhi.

References

Ahmed Abbasi and Hsinchun Chen. 2006. Visualizing authorship for identification. In *International Conference on Intelligence and Security Informatics*, pages 60–71. Springer.

Satanjeev Banerjee and Alon Lavie. 2005. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, pages 65–72.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. In *34th Conference on Neural Information Processing Systems (NeurIPS 2020)*, pages 1–15.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734.

Alexis Conneau and Guillaume Lample. 2019. Cross-lingual language model pretraining. In *Advances in Neural Information Processing Systems*, volume 32, pages 7059–7069.

Sumanth Dathathri, Andrea Madotto, Janice Lan, Jane Hung, Eric Frank, Piero Molino, Jason Yosinski, and Rosanne Liu. 2020. Plug and play language models: A simple approach to controlled text generation. In *ICLR*, pages 1–25.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *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)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Fangxiaoyu Feng, Yinfei Yang, Daniel Cer, Naveen Arivazhagan, and Wei Wang. 2020. Language-agnostic BERT sentence embedding. *arXiv preprint arXiv:2007.01852*.

Sebastian Gehrmann, Hendrik Strobelt, and Alexander M Rush. 2019. 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.

Daphne Ippolito, Daniel Duckworth, Chris Callison-Burch, and Douglas Eck. 2020. Automatic detection of generated text is easiest when humans are fooled. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1808–1822.

Farkhund Iqbal, Rachid Hadjidj, Benjamin CM Fung, and Mourad Debbabi. 2008. A novel approach of mining write-prints for authorship attribution in e-mail forensics. *digital investigation*, 5:S42–S51.

Ganesh Jawahar, Muhammad Abdul-Mageed, and Laks Lakshmanan, V.S. 2020. Automatic detection of machine generated text: A critical survey. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 2296–2309, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Nitish Shirish Keskar, Bryan McCann, Lav R Varshney, Caiming Xiong, and Richard Socher. 2019. Ctrl: A conditional transformer language model for controllable generation. *arXiv preprint arXiv:1909.05858*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar. Association for Computational Linguistics.

J Peter Kincaid, Robert P Fishburne Jr, Richard L Rogers, and Brad S Chissom. 1975. Derivation of new readability formulas (automated readability index, fog count and flesch reading ease formula) for navy enlisted personnel. Technical report, Naval Technical Training Command Millington TN Research Branch.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Yang Liu and Mirella Lapata. 2019. Text summarization with pretrained encoders. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3721–3731.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A robustly optimized BERT pretraining approach. *arXiv preprint arXiv:1907.11692*.

Enrique Manjavacas, Jeroen De Gussem, Walter Daelemans, and Mike Kestemont. 2017. Assessing the stylistic properties of neurally generated text in authorship attribution. In *Proceedings of the Workshop on Stylistic Variation*, pages 116–125.

Shaoor Munir, Brishna Batool, Zubair Shafiq, Padmini Srinivasan, and Fareed Zaffar. 2021. Through the Looking Glass: Learning to Attribute Synthetic Text Generated by Language Models. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*.

Arvind Narayanan, Hristo Paskov, Neil Zhenqiang Gong, John Bethencourt, Emil Stefanov, Eui Chul Richard Shin, and Dawn Song. 2012. On the feasibility of internet-scale author identification. In *2012 IEEE Symposium on Security and Privacy*, pages 300–314.

Nathan Ng, Kyra Yee, Alexei Baevski, Myle Ott, Michael Auli, and Sergey Edunov. 2019. Facebook fair's wmt19 news translation task submission. In *Proceedings of the Fourth Conference on Machine Translation (Volume 2: Shared Task Papers, Day 1)*, pages 314–319.

R Nils and I Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, Hong Kong, China*, pages 3–7.

Xudong Pan, Mi Zhang, Shouling Ji, and Min Yang. 2020. Privacy risks of general-purpose language models. In *2020 IEEE Symposium on Security and Privacy (SP)*, pages 1314–1331.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.

Lisa Pearl and Mark Steyvers. 2012. Detecting authorship deception: a supervised machine learning approach using author writeprints. *Literary and linguistic computing*, 27(2):183–196.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training (2018). *URL https://s3-us-west-2. amazonaws. com/openai-assets/research-covers/language-unsupervised/language_ understanding_paper. pdf*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Sebastian Ruder, Parsa Ghaffari, and John G Breslin. 2016. Character-level and multi-channel convolutional neural networks for large-scale authorship attribution. *arXiv preprint arXiv:1609.06686*.

Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2019. Energy and policy considerations for deep learning in NLP. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3645–3650, Florence, Italy. Association for Computational Linguistics.

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

Qicai Wang, Peiyu Liu, Zhenfang Zhu, Hongxia Yin, Qiuyue Zhang, and Lindong Zhang. 2019. A text abstraction summary model based on bert word embedding and reinforcement learning. *Applied Sciences*, 9(21):4701.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems*, volume 32, pages 5753–5763.

SoYeop Yoo and OkRan Jeong. 2020. An intelligent chatbot utilizing bert model and knowledge graph. *Journal of Society for e-Business Studies*, 24(3).

Shi Yu, Yuxin Chen, and Hussain Zaidi. 2020. A financial service chatbot based on deep bidirectional transformers. *arXiv preprint arXiv:2003.04987*.

Rowan Zellers, Ari Holtzman, Hannah Rashkin, Yonatan Bisk, Ali Farhadi, Franziska Roesner, and Yejin Choi. 2019. Defending against neural fake news. In *Advances in Neural Information Processing Systems*, pages 9054–9065.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. *Advances in neural information processing systems*, 28:649–657.

Xunjie Zhu and Gerard de Melo. 2020. Sentence analogies: Exploring linguistic relationships and regularities in sentence embeddings. *arXiv preprint arXiv:2003.04036*.

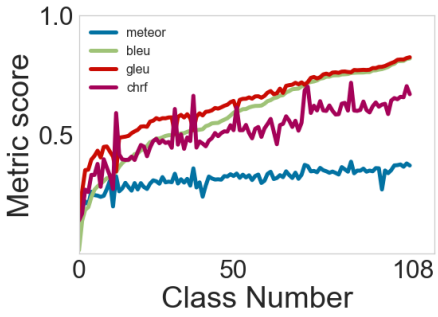


Figure 7: Comparison of different metrics on the synthetic corpus taking the reference of the organic corpus. The classes are sorted by increasing CHRF scores.

A Appendix

A.1 Implementation, Infrastructure, Software

We run all experiments on a 24-core machine with two Intel(R) Xeon(R) Silver 4116 CPU@2.10GHz CPU's and 512 GB RAM. Additionally, the server has a GeForce RTX 2080 Ti (11 GB) GPU card. We use huggingface, pytorch (1.4.0) and Tensorflow (v.1.23.0) to implement and evaluate all deep learning models in the paper. For classical ML, we utilize scikit-learn (v.0.23.1). All implementations are done using Python language (v.3.7).

A.2 Hyper-parameters

Fine-tuned RoBERTa + CNN: We attach 2 CNN convolutional 1D of kernel sizes 2 and 3 layers back to back. The stride for both the layers is 1. The number of output filters for each of the two convolutional layers is 16. We then attach a batch normalization layer of size 16. This is followed by a max pooling layer of size 2 with stride 1. We fine-tuned the model for 15 epochs and monitored the validation loss. If the validation loss did not improve for 5 consecutive epochs, we stopped the fine-tuning.

Fine-tuned RoBERTa + Dense: We used RoBERTaForSequenceClassification wrapper from the huggingface module, which attaches a softmax layer on top of pre-trained RoBERTa model. We extracted the embeddings of the second to last layer of size 1x768. For all of our models, we used a *AdamW* optimizer with a learning rate of 0.00005. We used batch size of 48 and only picked the first 75 tokens of each token. We did not use any padding.

For the soft classifiers, using the fine-tuned RoBERTa embeddings, we obtained the best results with the following parameters -

SVM: C = 0.1, cachesize=200, classweight = None, coef0 = 0.0, degree=3, gamma='scale', kernel='linear', tol=0.001

MLP: activation='relu', alpha=0.01,epsilon=1e-08, hiddenlayersizes = 64, learningrate='adaptive', learningrateinit=0.0001

RF - criterion='entropy', maxdepth=None, maxfeatures='auto', maxleafnodes=None, minsamplesleaf = 1

DT - criterion = 'entropy', maxdepth = None, maxfeatures = None, maxleafnodes=None, minsamplesleaf = 1

GNB - Default sklearn parameters performed the best

A.3 Running Time

All fine-tuned models took a maximum of 8 minutes per epoch. For the soft classifiers, MLP and SVM took the maximum time. Due to the large size of the dataset and the large embedding space, SVM took about 6 hours to train. MLP took an average of 3 hours. Rest of the classifiers took less than an hour for training.

A.4 Additional Results

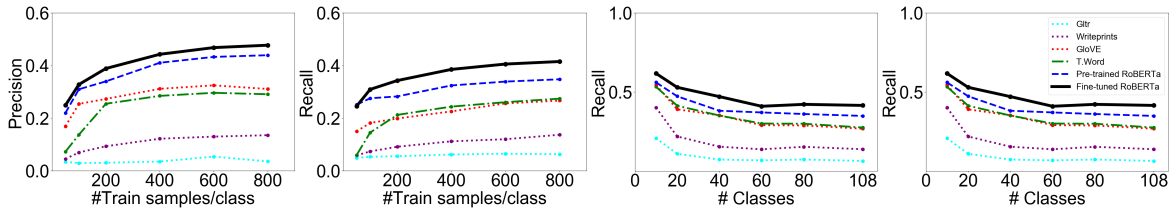
A.4.1 Additional Dataset Analysis

Besides the analaysis in the dataset section of the main text, we used 4 metrics - METEOR, BLEU, GLEU and CHRF for measuring the coherency of the synthetic text, using the organic text as reference. Using 1000 comments from each class of synthetic and organic text, we obtained high class-wise average scores on all metrics - METEOR (0.31), BLEU (0.58), GLEU (0.63) and CHRF (0.51). This provides further evidence that synthetic text is coherent and readable when compared to its organic counterpart. In figure 7 , we illustrate the scores for each individual class, sorted by increasing CHRF scores.

A.4.2 Feature extraction

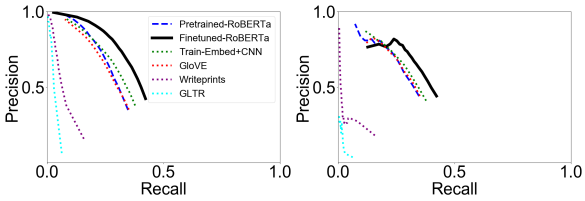
Besides the feature extraction methods we mentioned in the Methods section of the main text, we also experimented with two more methods:

1. **Fine-tuned GLTR:** Using the training set, we fine-tuned separate BERT and GPT2 models for each class for the task of mask completion. All models were then used for extracting GLTR word likelihood features for the complete training and test sets. Subsequently, the training representations were then fed to a sequential neural classi-



(a) (b) (c) (d)

Figure 8: A comparison between the performances of various embeddings by varying problem parameters. (a) Precision and (b) Recall with the varying training set size. (a) Precision and (b) Recall with the varying number of classes. Overall, fine-tuned RoBERTa outperforms all the baselines.



(a) Micro (b) Macro

Figure 9: (a) Micro and (b) Macro *precision-recall trade-off* by varying the gap statistic threshold.

Base Architecture	Classifier	Macro		Top - 5	Top - 10
		Precision	Recall		
Writeprints	DT	6.9	6.6	6.6	6.6
	SVM	19.3	16.3	33.2	44.8
GLTR	DT	5.7	5.5	5.57	5.58
	SVM	7.3	7.0	10	13.1
RoBERTa	DT	6.3	5.3	6.3	6.3
	SVM	8.3	6.9	7.8	8.9
Trainable-Word	MLP	20.4	18.9	34.3	44.3
	CNN	28.6	27.0	44.0	53.1
FT RoBERTa	Dense	44.0	42.3	60.8	68.9
	DT	29.5	28.7	28.7	28.7
	SVM	42.7	41.1	58.3	65.6

Table 3: Results of Decision Tree (DT) and SVM with the embeddings. We also include the results of the trainable word embeddings model.

fier like Bi-LSTM. The intuition was that word likelihoods extracted using the class's fine-tuned GLTR model would be high for synthetic text generated for the class's language model. For example, a r/wallstreetbets fine-tuned GLTR feature extractor would extract higher word likelihoods, as compared to other GLTR models, from a synthetic comment generated by the language model fine-tuned on r/wallstreetbets. However, for an extremely small setting of 5 classes, we only obtained a precision and recall of around 33% each. Due to the (a) expensive cost of fine-tuning 108 BERT and GPT2 models, (b) expensive cost of extracting GLTR features for a dataset of 100k examples, (c) extremely poor results of the approach on a small setting, we did not continue with experimenting the model in a larger setting.

2. **Trainable word embeddings** - We tokenized and represented each comment using an allocated set of integers based on the vocabulary of the training set. Then, similar to the GloVE feature extraction, we passed the representation into a trainable word embedding layer, followed by MLP or CNN. The results for these were slightly worse than that for the GloVE features. We report the results in Table 3.

Additionally, for all the feature representations mentioned in the main text, we tested SVM and decision tree. Overall, they were outperformed by the other classifiers with one notable exception. For Writeprints features, SVM showed the best results of precision and recall of 19.3% and 13.3% respectively. We report the results in Table 3.

A.5 Other baselines

Precision-recall trade-off. For the precision-recall trade-off in the results section of the main text, we presented a comparison with additional baselines in Figure 9. Fine-tuned RoBERTa performs the best among all methods for both micro and macro precision-recall trade-offs. They are followed in a decreasing order by the pre-trained RoBERTa embeddings, the trainable word embeddings, the GloVE word embeddings, Writeprints, and GLTR respectively.

Training set size and classes. We report the comparison with all other baselines for the varying training size and the number of classes in Figure 8. Similar to the precision-recall trade-off, the fine-tuned RoBERTa embeddings perform the best, followed by pre-trained RoBERTa embeddings, GloVE word embeddings, trainable word embeddings, Writeprints, and GLTR.