Proceedings of the 13th Conference on Language Resources and Evaluation (LREC 2022), pages 6889–6898
Marseille, 20-25 June 2022

Unraveling the Mystery of Artifacts in Machine Generated Text

Jiashu Pu, Ziyi Huang, Yadong Xi, Guandan Chen, Weijie Chen, Rongsheng Zhang*
Fuxi AI Lab, NetEase Inc., Hangzhou, China
{pujiashu,huangziyi03,xiyadong,chenguandan,chenweijie05,zhangrongsheng}@corp.netease.com

Abstract

As neural Text Generation Models (TGM) have become more and more capable of generating text indistinguishable from human-written ones, the misuse of text generation technologies can have serious ramifications. Although a neural classifier often achieves high detection accuracy, the reason for it is not well studied. Most previous work revolves around studying the impact of model structure and the decoding strategy on ease of detection, but little work has been done to analyze the forms of artifacts left by the TGM. We propose to systematically study the forms and scopes of artifacts by corrupting text, replacing them with linguistic or statistical features, and applying the interpretable method of Integrated Gradients. Comprehensive experiments show artifacts a) primarily relate to token co-occurrence, b) feature more heavily at the head of vocabulary, c) appear more in content word than stopwords, d) are sometimes detrimental in the form of number of token occurrences, e) are less likely to exist in high-level semantics or syntaxes, f) manifest in low concreteness values for higher-order n-grams.

Keywords: Natural Language Generation,Text categorisation,Corpus,Syntax,Semantics

1. Introduction

In recent years, neural **T**ext **G**eneration **M**odels (TGM) have grown by leaps and bounds with the support of big data and escalating computing power (Brown et al., 2020). State-of-the-art models can generate fluent and genuine-looking text (Gehrmann et al., 2019; Dou et al., 2021), thus we need to be especially wary of the abuse of text generation techniques, such as producing fake news, impersonating others in email, phishing, etc. (Jawahar et al., 2020).

Recently, Humans are worse at detecting synthetic text than Machines (Gehrmann et al., 2019; Solaiman et al., 2019) because Humans cannot properly perceive the difference in data distribution, but rely more on semantic errors or logical contradictions (Ippolito et al., 2020). This phenomenon has spawned much work investigating the use of neural classifiers to detect synthetic text, most of which analyze the impact of variables such as the structure of TGM, the decoding strategy, and the sentence length (Tay et al., 2020; Ippolito et al., 2020; Solaiman et al., 2019; Munir et al., 2021). Besides, a small amount of work studies TGM's property through its left behind ***artifacts*** (detectable signatures that originate from the TGM). (Tay et al., 2020) reveals that TGMs of different decoding configurations leave distinguishable artifacts when generating text, proving artifacts are not correlated with word order. Another work (Ippolito et al., 2020) claims artifacts exist in over-sampled high-likelihood words when TGM adopts decoding strategies of top-k sampling. Despite these sporadic findings, there is a lack of systematic analysis of the forms in which artifacts exist.

To fill this void, we conduct an empirical study on three datasets to reveal characteristics and possible manifestations of artifacts. Specifically, we corrupt text in different ways or replace them with high-level linguistic/statistical features, retaining or corrupting certain information of the text; we locate artifacts mainly based on the performance variation of the classifier against different corruptions. Besides, to track artifacts in the details, we compute the contribution of tokens by means of interpretable methods.

We summarize our contribution as follows:

- We open-sourced a **C**hi**n**ese Novel **D**ataset cr**A**wled f**R**om m**I**xed online s**O**urces (CnDARIO) for artifacts discovery in the Chinese Literature domain.
- We offer empirical findings for artifacts: **a)** token co-occurrence is the predominant form in which artifacts exist, **b)** artifacts exist mainly at the head of the vocabulary, **c)** content words contain more artifacts than stopwords, **d)** the number of occurrences of tokens offers limited or even detrimental artifacts, **e)** high-level semantic/syntactic features contain much fewer artifacts than shallow features, **f)** some artifacts are present in higher-order n-grams in the form of low concreteness values, **g)** pre-training is extremely helpful for effective use of artifacts[1].

2. Task Formulation

Though neural classifiers achieve extremely high accuracy on discerning synthetic text, the reason for it is not fully-studied. Our work attempts to explain this phenomenon by unveiling the forms in which artifacts exist. To this end, we adopt several text corruption/replacement operations, which may retain or corrupt certain linguistic or statistical information, as surrogates for unraveling artifacts. Formally, given a classification dataset $D = (E_1, \ldots, E_n)$ containing both Human and Machine excerpts[2], we corrupt and

* Corresponding author

[1] Code is available at `https://github.com/iamlxb3/UMAMGT`

[2] Each excerpt E_i contains m tokens, denoted as $E = (t_1, \ldots, t_m)$

replace the original text in D with various operations. For each operation CO, we train the neural classifier f_θ on the corrupted training set D^{co}_{train}, regarding the accuracy variation of f_θ on D^{co}_{test} as a sensitivity indicator for the presence or the absence of certain linguistic/statistical features associated with CO. The high absolute value or the drastic change of accuracy helps us to attribute artifacts to specific linguistic/statistical features. This process can be comprehend as sensitivity analysis (Saltelli, 2002; Saltelli et al., 2008) of corruption/replacement operations (input) on the generalisation performance of f_θ (output). As a practical example, suppose we apply the operation of 'token shuffle' to D, and f_θ still manages to achieve high accuracy, we can conclude token co-occurrences in the corrupted D^{co} contains enough distinguishable artifacts. Conversely, if there is a significant drop in accuracy, it is evident that the token order has detectable artifacts and there are relatively small differences in the distribution of word co-occurrence between genuine and synthetic text. Furthermore, we use interpretable methods based on Influence Functions (Koh and Liang, 2017) to directly analyze the connection between the classifier's decisions and the properties of the n-grams.

3. Text Corruption Operations

We define the corruption/replacement operation as $CO : E \rightarrow E^{co}$, where E^{co} is a corrupted excerpt; we keep the length variable fixed, adding `[MASK]` token to E^{co} when necessary. All text corruptions operate within the scope of E, with the token being the basic operating unit. Operations in the following sections are selected to cover as many forms in which artifacts may exist as possible.

3.1. Corruptions in the text form

Deduplicate tokens Given the original excerpt E, the deduplicate operation only keeps the first occurrence of token t, replacing all subsequent occurrences of t with the `[MASK]` token.

Shuffle tokens Given the original novel excerpt E, the shuffle operation produces a new excerpt E^{co} with random token orders, i.e., $\forall t \in E$, t has a different position in E^{co}.

Retain only (non)-stopwords Stopwords[3] are the most common words, and they are considered to retain abstract author signatures for Human authorship attribution (Rajkumar et al., 2009). We are curious whether the inclusion or exclusion of stopwords helps to attribute a Machine author—the TGM, too.

[3]For Chinese stopwords, we adopt the Baidu stopword list `https://github.com/goto456/stopwords`, while for English, we combine the Baidu stopword list with NLTK stopword list `https://gist.github.com/sebleier/554280`.

Retain tokens in high/low frequency regions To investigate the differences in token frequency between Human and Machine text, we design the frequency gap score as

$$score_g = \frac{Freq(t_i, D_m) - Freq(t_i, D_h)}{Freq(t_i, D_m)}$$

, where $Freq(t, D)$ is the frequency of a token t in the corpus D and $Rank(t, D)$ is the frequency rank of t in descending order. D_h and D_m are subsets of D, containing Human written and Machine generated text respectively. Figure 1 shows that the frequency distribution between Human text and Machine text differs in each of the three datasets. Human and Machine use of high-frequency tokens is broadly similar on CnDARIO but Humans use low-frequency tokens more frequently, while the Writing Prompt dataset shows an opposite trend. On the CNN/Daily Mail dataset, the difference of token usage between Humans and Machines is consistently small across all frequency ranks.

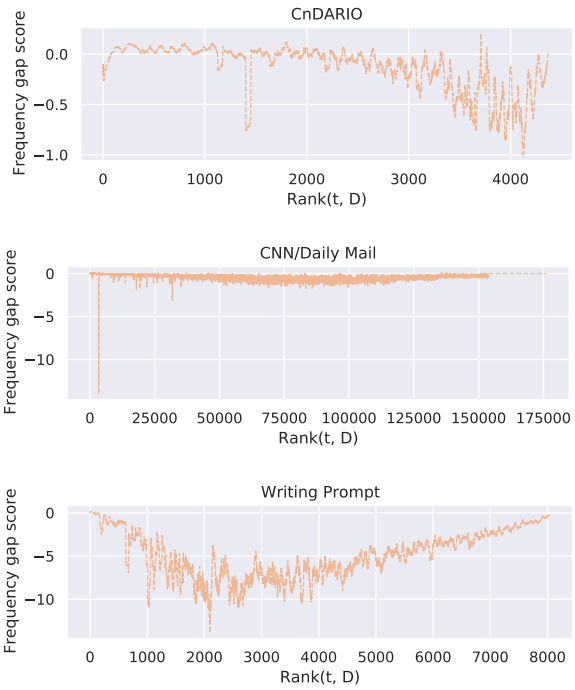


Figure 1: Differences in unigram token frequency distribution between Human and Machine text. Scores are smoothed over a window size of 50. The protruding dropping line around Rank 3000 of the Grover dataset corresponds to the token 'Opinion:'.

We introduce two corruption operations to further study the effect of retaining tokens within a certain frequency range. The operation to constrain tokens of E^{co} in the high-frequency region satisfies

$$t^{co}_i = \begin{cases} t_i & \text{if } Rank(t_i, D) < \rho \\ \texttt{[MASK]} & \text{if } Rank(t_i, D) \geqslant \rho \end{cases}$$

, where t_i and t^{co}_i are the i-th token in E and E^{co}, $\rho \in \mathbb{N}$ is the rank threshold to split the high and

Origin_1	It is believed Congress is going to finish its authorization of NASA's budget.
Replace with likehood rank_8	509 469 330 5299 4168 285 263 13 2537 695 23919 288 263 ...
Replace with POS_9	PRON AUX VERB PROPN AUX VERB PART VERB PRON NOUN ...
Replace with Dep. Tree_{10}	nsubjpass 0 6 auxpass 3 6 ROOT 6 6 nsubj 15 27 aux 24 27 ccomp 27 6 ...
Replace with Cons. Tree_{11}	(S(NP(PRP))(VP(VBZ)(VP(VBN)(SBAR(S(NP(NNP))(VP(VBZ)(VP(VBG) ...
Replace with NE_{12}	ORG 15 23 ORG 64 68

Table 1: An concrete example of how a piece of text is replaced. Some replaced text are shortened for the sake of presentation. We have sub-scripted each operation to be consistent with Section 4.3 and Table 4.

low frequency region. Constraining tokens in the low-frequency region is defined oppositely.

3.2. Operations by replacing text with different forms

(Gehrmann et al., 2019) shows that highlighting text passages according to tokens' likelihood rank greatly improves the Human detection rate of synthetic text. Moreover, (See et al., 2019) finds synthetic text can be syntactically repetitive even if they are not textually repetitive. These evidences motivate us to test how the classifier performs with mere high-level linguistic or statistical features. We purpose to replace the original text with other linguistic forms or statistical values. To simplify experiments, all replaced forms are regarded as strings, tokenized, and trained as ordinary text.

Replace text with likelihood ranks One of the characteristics of Human language is that it intermittently dips in and out of low probability zones (Holtzman et al., 2019). Inspired by this, we propose to replace text with likelihood ranks to verify whether the classifier can exploit this distinctiveness. The likelihood rank of an excerpt E can be obtained by querying a TGM. Concretely, at each position i, the TGM produces a probability distribution $P\left(t_i \mid E_{1\ldots i-1}\right)$ of token t_i, from which we obtain the likelihood rank r_i of t_i, constituting $E^{co} = (r_1, \ldots, r_i, \ldots)$. For Out-Of-Vocabulary tokens, we use the rank of `[UNK]` for substitution.

Replace text with specific linguistic features To test whether Machine-generated text differs significantly from Human text in certain linguistic features, we replace the original text form with part-of-speech, Dependency Trees and Constituent Trees. In experiments, Linguistic features of POS and Dependency Trees are obtained by *spaCy* [4], while the Constituent Tree of text is parsed by Berkeley Neural Parser (*benepar*) (Kitaev et al., 2019). We adopt specific models of `en_core_web_sm` and `zh_core_web_sm` in *spaCy*, and `benepar_en3` and `benepar_zh2` in *benepar*.

∘ **Part of Speech** A part-of-speech (POS) is a category of words that have similar grammatical properties. It has been shown the distribution of POS n-grams is a useful feature to represent the textual style (Roemmele et al., ; Ireland and Pennebaker, 2010), and the distribution of POS is different between real and fake text (Vijayaraghavan et al., 2020; Pérez-Rosas et al., 2018). These findings inspire us to test the effect of replacing the original text input T with a a list of POS, denoted as $E^{co} = (pos_1, \ldots, pos_i, \ldots), pos_i \in V_{pos}$, where pos_i is a POS and the size of POS Vocabulary V_{pos} is 36.

∘ **Dependency Trees** Dependency parsing is a syntactical parsing technique used to identify semantic relations between words in a sentence (Kübler et al., 2009). Such relations form a tree, with each word having exactly one head. We parse the original text input T into a list of dependency relations, with each followed by the position of its head word and tail word. The list of Dependency Trees is defined as $E^{co} = (dp_1, head_1, tail_1, \ldots, dp_i, head_i, tail_i \ldots), dp_i \in V_{dp}$, where dp_i is a dependency relation, $head_i$ and $tail_i$ are the positions of head and tail word of dp_i. The size of dependency relation Vocabulary V_{dp} is 45.

∘ **Constituent Trees** Constituency parsing is another type of syntactical parsing, based on the formalism of context-free grammars. In the constituency parse tree, a sentence is divided into constituents of a specific category in the grammar, such as Verb phrase (VP) and Noun phrase (NP) (Jurafsky, 2000). We transform the original text input E into a string-form constituency parse tree E^{co}, where only constituent categories and the tree structure are available. The vocabulary size of constituents is 136.

∘ **Named Entities** Named entities often include person names, organizations, locations, and etc. (Sekine and Ranchhod, 2009). We extract all named entities from the text with corresponding span locations. Given the original text input E, we extract named entities and form a NE list $E^{co} = (ne_1, sp_{l1}, sp_{r1}, \ldots, ne_i, sp_{li}, sp_{ri} \ldots), ne_i \in V_{ne}$, where ne_i is a named entity, sp_{li} and sp_{ri} are the left and right position index of the span of entity ne_i. The total number of named entities V_{ne} is 18.

4. Experiment

4.1. Dataset

We collect and generate datasets of paired samples of Human-written and Machine-generated text from three sources. The **first** source is the CNN/Daily Mail

[4] `https://spacy.io`

Dataset	Source (Parameters)	N. Sample	N. Unigram	Avg.Len.	Distinct-1/2/3
CnDARIO	Human	12661	4767	103	0.682 / 0.925 / 0.960
	GPT-2 (1.5 billion)	12661	5111	103	0.794 / 0.958 / 0.971
CNN/Daily Mail	Human	29610	356189	363	0.634 / 0.945 / 0.983
	Grover Mega (1.5 billion)	29610	379309	366	0.604 / 0.917 / 0.962
Writing Prompt	Human	8945	43499	156	0.594 / 0.908 / 0.957
	Fusion Model (255.4 million)	8945	8392	149	0.432 / 0.753 / 0.866

Table 2: Basic data analysis. Abbreviations of *N. Sample*, *N. Unigram*, and *Avg.Len.* stands for the number of samples, the number of unique unigram tokens and the average excerpt length in a dataset respectively. *Distinct-1/2/3* is the Distinct value (Li et al., 2015) for unigram, bigram and trigram.

dataset (Nallapati et al., 2016), from which we randomly select the headline and the three subsequent sentences as the prefix. With the prefix, we use Grover-Mega (Zellers et al., 2019) to generate an excerpt by nucleus sampling ($p = 0.95, t = 1.0$) (Holtzman et al., 2019), pairing it with the original subsequent text. Similarly, we collected the readily available synthetic and Human excerpts from the **second** source — the Writing Prompt dataset (Fan et al., 2018), with synthetic text generated by the Fusion Model (Fan et al., 2018). The **third** source is CnDARIO dataset, the size of which is 30 gigabytes. We first solidly train the GPT-2 (Radford et al., 2019) on this dataset, with a vocabulary size of 13,762 and a context length of 1,024. Afterward, we query the pre-trained GPT-2 to generate synthetic novel excerpts. The length of the prefix and continuation is randomly chosen between 3 and 5. For the construction of all datasets, we include a post-processing step to ensure the genuine and the synthetic excerpt of the same prefix contains roughly the same amount of tokens, with a length difference of 10 tokens or less. We also provide the likelihood rank of the TGM when decoding each token for the CNN/Daily Mail and CnDARIO dataset.

In summary, we provide carefully constructed datasets of paired samples from three different domains: Online Forum, News, and Literature[5]. We believe these readily available datasets are friendly for students and labs with fewer computing resources. It is also worth noting that we provide an alternative form of data—likelihood ranks—that corresponds to the token, which allows researchers to easily investigate the effect of using likelihood rank to distinguish synthetic text.

4.2. Experimental setup

We adopt Roberta-base (Liu et al., 2019) as the classifier [6] because Roberta has been proved to be highly effective in detecting synthetic text and author attribution (Uchendu et al., 2021; Uchendu et al., 2020). We choose subword (Byte-Pair-Encoding (Gage, 1994)) and character tokenization for English and Chinese text respectively. We split all datasets into three parts, 80% for training, 10% for validation, and the rest 10% for testing. We determine the best model based on its accuracy on the validation set, then report its performance on the test set. For every corruption or replacement operation, we repeat the training and evaluation process 15 times with different random seeds, with each seed controlling the data split, the shuffle order, the initialized parameters of the non-pre-trained model, etc.

4.3. Effect of corruptions and replacements

We define the abbreviations of corruptions and replacements as follows: (1) *origin*: the original text; (2) *dedup.*: token deduplication; (3) *shuf.*: token shuffling operation; (4) *shuf.dedup.*: combined operations of token shuffling and deduplication; (5) *in.stop.*: only include stopwords; (6) *ex.stop.*: exclude stopwords; (8) *like.rank*: replace tokens with likelihood ranks of TGM; (9) *POS.*: replace tokens with POS; (10) *Dep.*: replace text with Dependency trees; (11) *Cons.*: replace text with Constituent Trees; (12) *NE.*: replace text with extracted Named entities.

4.3.1. Effect of text corruption operations

From Table 3, we summarise the following key findings. **Erasing information on the number of occurrences of token hardly hurts the performance and even greatly improves the accuracy of the non-pre-trained classifier.** For all datasets, the difference in accuracy between classifiers trained on *Dedup.* and *origin* data is minimal. Interestingly, when the classifier is not pre-trained, the accuracy of *Dedup.* is much higher than the accuracy of *origin* on the CnDARIO and CNN/Daily Mail dataset. We speculate that *Dedup.* operation simplifies the text and reduces the interference of the number of token occurrences, making it easier for the classifier to concentrate on token co-occurrences.

The Boolean BOW features are sufficient for the classifier to detect synthetic text. The *shuffle operation* disrupts the semantic and syntactic structure of

[5]The dataset is available at `https://drive.google.com/file/d/1xA9TtDYJE9BEwecL8QJ5d0LTytn5hhBr`

[6]Chinese Roberta: `https://huggingface.co/hfl/chinese-roberta-wwm-ext`. English Roberta: `https://huggingface.co/roberta-base`

Model	Dataset	origin$_1$	dedup.$_2$	shuf.$_3$	shuf. dedup.$_4$	in.stop.$_5$	ex.stop.$_6$
Pre.	CnDARIO	$0.955^{\ddagger^{all}}$	$\mathbf{0.934}^{\ddagger^{\{4,5,6\}}\dagger^{\{3\}}}$	$\underline{0.931}^{\ddagger^{\{4,5\}}\dagger^{\{6\}}}$	$0.906^{\dagger^{\{6\}}}$	0.825	$0.917^{\dagger^{\{4,5\}}}$
N.Pre.	CnDARIO	$0.845^{\ddagger^{\{3,5,6\}}}\downarrow$	$\mathbf{0.892}^{\ddagger^{\{1,3,5,6\}}}$	$0.820^{\ddagger^{\{6\}}\dagger^{\{5\}}}$	$\underline{0.891}^{\ddagger^{\{1,3,5,6\}}}$	0.744	$0.811^{\ddagger^{\{5\}}}$
Pre.	CNN/D.	$0.929^{\ddagger^{all}}$	$\mathbf{0.878}^{\ddagger^{\{2,3,4,5,6\}}}$	$0.830^{\ddagger^{\{5\}}\dagger^{\{4\}}}$	$0.782^{\ddagger^{\{5\}}}$	0.584	$\underline{0.851}^{\ddagger^{\{3,5\}}\dagger^{\{4\}}}$
N.Pre.	CNN/D.	$0.501\downarrow$	$\mathbf{0.635}^{\ddagger^{\{1,3,5,6\}}}$	$0.512^{\ddagger^{\{1,5\}}}$	$\underline{0.630}^{\ddagger^{\{3,5\}}}$	0.500	0.500
Pre.	Writing P.	$0.997^{\ddagger^{\{5\}}\dagger^{\{2,3,4,6\}}}$	$\mathbf{0.995}^{\ddagger^{\{5\}}}$	$\mathbf{0.995}^{\ddagger^{\{5\}}\dagger^{\{4\}}}$	$0.992^{\ddagger^{\{5\}}}$	0.986	$\underline{0.994}^{\ddagger^{\{5\}}}$
N.Pre.	Writing P.	$0.988^{\ddagger^{\{6,4\}}}$	$\underline{0.978}^{\ddagger^{\{5\}}\dagger^{\{6\}}}$	$\mathbf{0.986}^{\ddagger^{\{5,6\}}}$	$0.975^{\ddagger^{\{5,6\}}}$	0.894	$0.947^{\ddagger^{\{5\}}}$

Table 3: Average accuracy of the classifier when text are applied with corruption operations. Standard deviation are omitted for brevity. *Pre.* denotes the pre-trained LM while *Not.Pre.* denotes the non-pre-trained LM. *CNN/D.* and *Writing P.* denote the dataset of CNN/Daily Mail and Writing Prompt. The best results of corruption operations (*origin* is excluded) are in **bold** and the second bests are underlined. The down arrow $\downarrow$ indicates the classifier trained on the original text perform even worse than that trained on text with corruption operations. We conduct paired t-test (Kim, 2015) for each pair of corruption operations, with $\ddagger$ and $\dagger$ indicating p-value < 0.001 and p-value < 0.05 respectively, for example, $0.917\dagger^{\{4,5\}}$ in the 1st row means $ex.stop_6$ outperforms $shuf.dedup._4$ and $in.stop_5$ at a significance level of 0.05.

Model	Dataset	origin$_1$	like.rank$_8$	POS.$_9$	Dep.$_{10}$	Cons.$_{11}$	NE.$_{12}$
Pre.	CnDARIO	$0.955^{\ddagger^{all}}$	$0.767^{\ddagger^{\{12\}}}$	0.799	$\mathbf{0.814}^{\ddagger^{\{8,9,11,12\}}}$	$\underline{0.804}^{\ddagger^{\{8,12\}}\dagger^{\{9\}}}$	0.556
Pre.	CNN/D.	$0.929^{\ddagger^{all}}$	$\mathbf{0.922}^{\ddagger^{\{9,10,11,12\}}}$	$0.606^{\dagger^{\{12\}}}$	0.571	$\underline{0.607}^{\dagger^{\{8\}}}$	0.552
Pre.	Writing P.	$0.997^{\ddagger^{all}}$	/	$\mathbf{0.982}^{\ddagger^{\{9,10,11,12\}}}$	$0.938^{\ddagger^{\{12\}}}$	$\underline{0.949}^{\ddagger^{\{12\}}\dagger^{\{10\}}}$	0.797

Table 4: Average accuracy of the classifier when text are replaced with other forms.

the text, but even so, the performance loss of the classifier on two datasets is negligible. Decreased value of accuracy on the CnDARIO and Writing prompt dataset is merely 0.003 and 0.002 respectively, while the accuracy loss of 0.1 is significant on the CNN/Daily Mail dataset. The insensitivity of token order has been mentioned in several works (Pham et al., 2020; Sinha et al., 2021; Wang et al., 2018). Moreover, when we apply *shuf.* and *Dedup.* operations simultaneously, this loss of accuracy is still minimal (CnDARIO, Writing prompt) or acceptable (CNN/Daily Mail). The results indicate that the current TGM-generated text can easily be recognized by co-occurrence of tokens without using much advanced semantic and syntactic knowledge.

The content word contains more artifacts that can be learned than the stopwords. As the CNN/Daily Mail data has a large vocabulary, stopwords only account for a small percentage. When only retaining stopwords, the effective length of the text is very short, due to this, the classifier's performance degrades greatly. On the other two datasets, training the classifier with merely stopword or content words both yield acceptable accuracy, but the accuracy of *in.stop* is always lower than that of *ex.stop*.

4.3.2. Effect of constraining tokens in high/low frequency region

The most important observations in Figure 2 is generalized as follows: **sufficiently accurate differentiation between Human and Machine-written text requires only a fraction of the vocabulary.** Furthermore, on the CnDARIO and Writing Prompt dataset, we find the classifier trained with E^{co} of high-frequency tokens is comparable with or better than that trained with E^{co} of low-frequency tokens, even when the effective length of the former is much shorter. In contrast, the classifier shows the opposite pattern on the CNN/Daily mail dataset: the accuracy of the high-frequency region is slightly lower than that of the low-frequency region when the effective length of the high-frequency region is longer ($\rho = 400$ and $\rho = 800$). Despite the inconsistencies, all classifiers achieve satisfactory accuracy (close to or significantly above 0.8) when only a small portion of the vocabulary—0.0014, 0.00135, 0.0012 for CnDARIO, CNN/Daily mail, and Writing Prompt respectively—is available.

4.3.3. Effect of text replacement operations

Based on the statistics in Table 2 and trends in Figure 1, we rank the quality and the complexity of text on Writing Prompt, CnDARIO, and CNN/Daily Mail from easy to difficult. Based on this quality ranking and the results in Figure 4, we conclude the following pattern: **retaining only high-level semantic or syntactic features reduces the classifier's ability to recognize synthetic text, with the decrease of accuracy proportional to the text quality generated by TGM.** Training with pure dependency trees and constituent trees obtain comparable accuracy while training with POS has mixed results on three datasets. Replacement with

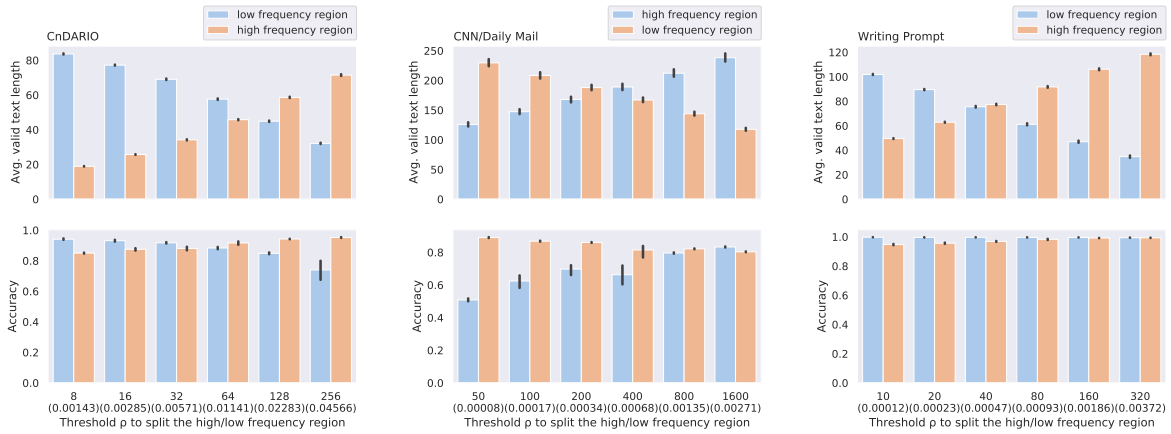


Figure 2: X-axis: Dividing threshold ρ between high and low frequency region; the numbers in parentheses represent the proportion of vocabulary size of the high-frequency region to the total vocabulary size. Y-axis (Top): Valid length of text (without `[MASK]`) when text are restricted to high or low frequency region. Y-axis (Bottom): Accuracy of the classifier when training with text from high and low frequency region.

Named Entities achieve the worst result probably because the effective length of E^{co} is too short, with only a few tokens in the text marked as Named Entities. Notably, the *like.rank.* operation has opposite results on CnDARIO and CNN/Daily Mail. When replacing text with likelihood ranks on CNN/Daily Mail data, the accuracy of the classifier is almost identical to that of the classifier trained from the original data, while replacing text with likelihood ranks on CnDARIO reduces the accuracy of the classifier by about 0.2. The low accuracy on CnDARIO is in better agreement with previous works (Ippolito et al., 2020; Diwan et al., 2021), where results show that likelihood rank is a poor feature for neural classifiers to identify Machine-generated text.

4.3.4. Pre-training matters a lot

Model pre-training has become the preceding step for almost all NLP tasks and this paradigm indeed proves effective (Devlin et al., 2019; Liu et al., 2019; Qiu et al., 2020). Our results in Table 2 once again confirm the validity of pre-training. It is extremely helpful in improving the accuracy of detecting synthetic text, and the classifier without pre-training is even worse than the pre-trained one by more than 0.2 on accuracy (e.g. applying *shuf.* on CNN/Daily Mail data). (Aghajanyan et al., 2020) provides a theoretical analysis of why pre-training is so effective — attributing the benefits of pre-training to minimized intrinsic dimension.

4.4. Interpretability analysis

In addition to exploring the existence of artifact forms in a holistic manner, we also seek to find artifacts by cutting through the more localized details. We employ the Integrated Gradients (IG)[7] (Sundararajan et al., 2017) approach to assign an interpretable attribution score for each token t in an excerpt E, rendering a list of attribution scores of $(a_1, \ldots, a_i, \ldots)$, where a_i is the sum of gradients of the classifier's prediction output to its embedding layer of t_i. The higher the value of a_i, the greater the contribution of t_i to the classifier's prediction of a particular class.

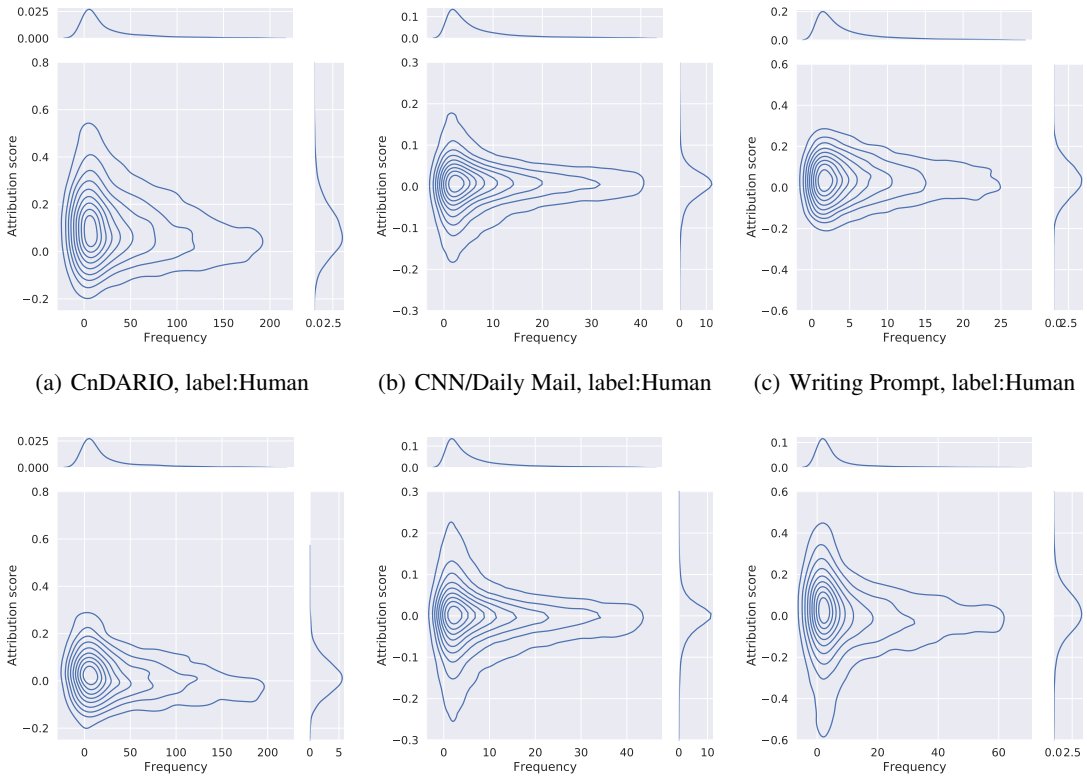
4.4.1. The relationship between term frequency and attribution score

From Figure 2 we find that the classifier achieves high performance with only a portion of high-frequency tokens, therefore we would like to further verify whether classifiers assign higher attribution scores to high-frequency tokens. For each dataset, we randomly sampled 10, 000 excerpts and apply IG towards them. From Figure 3, we can see that the classifier does not deliver higher attribution scores to high-frequency tokens, instead, some of the tokens in the low-frequency region receive larger absolute values of the attribution score, probably because the number of tokens in the low-frequency region is much larger, leading to high variance. In Figure 3(a), low-frequency tokens have a positive attribution mean (around 0.1) for the Human label; Human text of CnDARIO containing more low-frequency words (Figure 1) may contribute to this particular result. In summary, except for CnDARIO's Human label, attribution scores of unigram tokens basically maintain a normal distribution with zero as the mean regardless of the frequency variation. **The results also demonstrate that the classifier does not pay special attention to a specific frequency band or certain high frequency tokens, which further supports the view that token co-occurrence is the most critical artifact.**

4.4.2. Concreteness of highly attributed n-grams

We design experiments to explore whether n-grams tokens with high attribution values differ in *concreteness*. The *concreteness* of a word is defined as 'the degree

[7] We adopt the implementation of Captum (Kokhlikyan et al., 2020) and set the step of approximation to 500.



(a) CnDARIO, label:Human (b) CNN/Daily Mail, label:Human (c) Writing Prompt, label:Human

(d) CnDARIO, label: Machine (e) CNN/Daily Mail, label: Machine (f) Writing Prompt, label: Machine

Figure 3: Univariate joint distribution of unigram token frequency (X-axis) and IG attribution score (Y-axis). The probability density is estimated by kernel density (KDE) (Terrell and Scott, 1992).

to which the concept denoted by a word refers to a perceptible entity' (Brysbaert et al., 2014; See et al., 2019). Words such as 'soybean' and 'liquid' have a high concreteness of 4.82 and 4.72 while word such as 'acceptableness' has a low concreteness of 1.28. For English, we adopt the concreteness ratings of 40,000 common English lemmas by (Brysbaert et al., 2014), with words' rating scaling from 1 to 5.14. For Chinese, we adopt concreteness ratings of 9,877 two-character Chinese words[8] (Xu and Li, 2020). We use these ratings to measure the total concreteness of an n-grams by means of sliding windows (concreteness of tokens that has no rating is set to zero).

We employ IG to calculate the attribution scores for all n-grams in E, averaged over the entire training set, and calculated the concreteness values for top attributed n-grams in both classes (Human, Machine). In Figure 4, we present the relationship between n-gram attribution ranks and their concreteness values, finding little difference in the mean values of concreteness between Human and Machine text, especially for the CNN/Daily Mail dataset. Moreover, we find an intriguing pattern on the CnDARIO and CNN/Daily Mail datasets: **for top attributed n-grams, as n increases, the concreteness value of Machine text becomes lower and lower with respect to the mean value, and the classifier increasingly prefers lower concreteness values as cues (artifacts) for Machine-generated text.** However, on the Writing Prompt dataset, we find that the concreteness value is always around the mean, regardless of the variation in n. We hypothesize that Human and Machine text on this dataset are already very different in terms of shallow features, leading to the classifier not learning more implicit features like concreteness.

[8]For consistency with the English concreteness rating scale, we subtract the concreteness values of all Chinese two-character from 6.

5. Related work

The contrast of Humans' inability of detecting synthetic text and models' impressive performance has attracted many researchers. There are some detailed works examining TGMs directly, in the direction of repetition, rare words, decoding strategies, linguistic novelty, etc. (McCoy et al., 2021; See et al., 2019) Other works (Tay et al., 2020; Ippolito et al., 2020; Solaiman et al., 2019; Munir et al., 2021) analyze how neural classifiers perform the task of detecting synthetic text, investigating the impacts of various variables: classifiers' architecture, TGMs' decoding strategy, and excerpt length. (Jawahar et al., 2020) provides a survey on the automatic detection of Machine-generated text. MAUVE (Pillutla et al., 2021) directly compares the distribution of Machine and Human text

Figure 4: X-axis: rank of n-grams with the highest attribution value for Human (blue solid line) and Machine (orange dotted line) label, with attribution score descending from left to right. The concreteness score (Y-axis) is averaged over a window size of 50. Horizontal lines represent the mean of the concreteness value of all n-grams.

using divergence frontiers. Shifting the analysis target, (Dugan et al., 2020) shows that Human annotators mainly use logic inconsistency, token repetition, etc. as cues for identifying synthetic text. The most relevant work to us is the work of (Tay et al., 2020); it exclude its presence of artifacts in word order but leaves other presence forms of artifacts under-explored.

6. Limitations and Future Work

Most of the experiments we have designed revolve around uncovering the form in which the artifacts of synthetic text appear, but do not provide a good answer as to why TGM generates certain forms of artifacts. In addition, some of our findings are not agreed across all datasets (e.g. replace with *like.rank*) and more detailed experiments are needed for generalizable conclusions. We study the attribution scores of continuous n-grams, but there is still no practical solution for attributing skipped n-grams through existing interpretable methods of Influence Functions[9]. As the experimental results indicate the co-occurrences of tokens suffices to make a distinction, the explanation of how the classifier works can start from pinpointing the decisive token set. Besides, we call for new metrics for scoring synthetic text (e.g. the degree to which synthetic text can be easily detected by simple structure models and BOW features), new design of artifacts-inspired systems to aid Human judgment (e.g. prompt the user when a large number of n-grams of significantly lower concreteness appear) and greater use of simple classifiers that rely only on shallow features but are still effective—for detecting synthetic text—to minimize carbon emissions.

[9]The number of combinations grows exponentially with n, for instance, the total number of combinations of selecting 3 tokens from a sequence of length 256 is as high as $2,763,520$.

7. Conclusion

Currently, neural classifiers are far more capable than Humans at recognizing synthetic text. Most of the work addressing this phenomenon has been around the impact of factors such as TGMs' structure and decoding strategies on the ease of detection. Instead, we systematically investigate the presence of artifacts directly from the generated text, which helps gain insight into the TGMs. Detailed experiments on three datasets show that artifacts exist mostly in the form of surface-level semantics, and appear more in high-frequency tokens, but rarely present in the form of high-level semantics or syntax. In addition, we unexpectedly found artifacts in the form of concreteness in higher-order n-grams using an interpretable approach.

For the dataset, we have open-sourced a high-quality paired Chinese novel dataset and also supplemented the CNN/Daily dataset with a new form of data, likelihood ranks that stem from the Grover-Mega model.

8. Bibliographical References

Aghajanyan, A., Zettlemoyer, L., and Gupta, S. (2020). Intrinsic dimensionality explains the effectiveness of language model fine-tuning. *arXiv preprint arXiv:2012.13255*.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. (2020). Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Brysbaert, M., Warriner, A. B., and Kuperman, V. (2014). Concreteness ratings for 40 thousand generally known english word lemmas. *Behavior research methods*, 46(3):904–911.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*, pages 4171–4186.

Diwan, N., Chakravorty, T., and Shafiq, Z. (2021). Fingerprinting fine-tuned language models in the wild. *arXiv preprint arXiv:2106.01703*.

Dou, Y., Forbes, M., Koncel-Kedziorski, R., Smith, N. A., and Choi, Y. (2021). Scarecrow: A framework for scrutinizing machine text. *arXiv e-prints*, pages arXiv–2107.

Dugan, L., Ippolito, D., Kirubarajan, A., and Callison-Burch, C. (2020). Roft: A tool for evaluating human detection of machine-generated text. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 189–196.

Fan, A., Lewis, M., and Dauphin, Y. (2018). Hierarchical neural story generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 889–898.

Gage, P. (1994). A new algorithm for data compression. *C Users Journal*, 12(2):23–38.

Gehrmann, S., Strobelt, H., and Rush, A. M. (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.

Holtzman, A., Buys, J., Du, L., Forbes, M., and Choi, Y. (2019). The curious case of neural text degeneration. In *International Conference on Learning Representations*.

Ippolito, D., Duckworth, D., Callison-Burch, C., and Eck, D. (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.

Ireland, M. E. and Pennebaker, J. W. (2010). Language style matching in writing: Synchrony in essays, correspondence, and poetry. *Journal of personality and social psychology*, 99(3):549.

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

Jurafsky, D. (2000). *Speech & language processing*. Pearson Education India.

Kim, T. K. (2015). T test as a parametric statistic. *Korean journal of anesthesiology*, 68(6):540.

Kitaev, N., Cao, S., and Klein, D. (2019). Multilingual constituency parsing with self-attention and pre-training. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3499–3505, Florence, Italy, July. Association for Computational Linguistics.

Koh, P. W. and Liang, P. (2017). Understanding black-box predictions via influence functions. In *International Conference on Machine Learning*, pages 1885–1894. PMLR.

Kokhlikyan, N., Miglani, V., Martin, M., Wang, E., Alsallakh, B., Reynolds, J., Melnikov, A., Kliushkina, N., Araya, C., Yan, S., and Reblitz-Richardson, O. (2020). Captum: A unified and generic model interpretability library for pytorch.

Kübler, S., McDonald, R., and Nivre, J. (2009). Dependency parsing. *Synthesis lectures on human language technologies*, 1(1):1–127.

Li, J., Galley, M., Brockett, C., Gao, J., and Dolan, B. (2015). A diversity-promoting objective function for neural conversation models. *CoRR*, abs/1510.03055.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. (2019). Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

McCoy, R. T., Smolensky, P., Linzen, T., Gao, J., and Celikyilmaz, A. (2021). How much do language models copy from their training data? evaluating linguistic novelty in text generation using raven. *arXiv preprint arXiv:2111.09509*.

Munir, S., Batool, B., Shafiq, Z., Srinivasan, P., and Zaffar, F. (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: Main Volume*, pages 1811–1822.

Nallapati, R., Zhou, B., dos Santos, C., Gulçehre, Ç., and Xiang, B. (2016). Abstractive text summarization using sequence-to-sequence rnns and beyond. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 280–290.

Pérez-Rosas, V., Kleinberg, B., Lefevre, A., and Mihalcea, R. (2018). Automatic detection of fake news. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3391–3401.

Pham, T. M., Bui, T., Mai, L., and Nguyen, A. (2020). Out of order: How important is the sequential or-

der of words in a sentence in natural language understanding tasks? *arXiv preprint arXiv:2012.15180*.

Pillutla, K., Swayamdipta, S., Zellers, R., Thickstun, J., Welleck, S., Choi, Y., and Harchaoui, Z. (2021). Mauve: Measuring the gap between neural text and human text using divergence frontiers. *Advances in Neural Information Processing Systems*, 34.

Qiu, X., Sun, T., Xu, Y., Shao, Y., Dai, N., and Huang, X. (2020). Pre-trained models for natural language processing: A survey. *Science China Technological Sciences*, pages 1–26.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al. (2019). Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Rajkumar, A., Ravi, S., Suresh, V., Murthy, M. N., and Madhavan, C. V. (2009). Stopwords and stylometry: A latent dirichlet allocation approach. In *Proceedings of the NIPS 2009 Workshop on Applications for Topic Models: Text and Beyond (Poster Session), Whistler, BC, Canada*.

Roemmele, M., Gordon, A. S., and Swanson, R.). Evaluating story generation systems using automated linguistic analyses. In *SIGKDD 2017 Workshop on Machine Learning for Creativity*, pages 13–17.

Saltelli, A., Ratto, M., Andres, T., Campolongo, F., Cariboni, J., Gatelli, D., Saisana, M., and Tarantola, S. (2008). *Global sensitivity analysis: the primer*. John Wiley & Sons.

Saltelli, A. (2002). Sensitivity analysis for importance assessment. *Risk analysis*, 22(3):579–590.

See, A., Pappu, A., Saxena, R., Yerukola, A., and Manning, C. D. (2019). Do massively pretrained language models make better storytellers? *arXiv preprint arXiv:1909.10705*.

Sekine, S. and Ranchhod, E. (2009). *Named entities: recognition, classification and use*, volume 19. John Benjamins Publishing.

Sinha, K., Jia, R., Hupkes, D., Pineau, J., Williams, A., and Kiela, D. (2021). Masked language modeling and the distributional hypothesis: Order word matters pre-training for little. *arXiv preprint arXiv:2104.06644*.

Solaiman, I., Brundage, M., Clark, J., Askell, A., Herbert-Voss, A., Wu, J., Radford, A., Krueger, G., Kim, J. W., Kreps, S., et al. (2019). Release strategies and the social impacts of language models. *arXiv preprint arXiv:1908.09203*.

Sundararajan, M., Taly, A., and Yan, Q. (2017). Axiomatic attribution for deep networks. In *International Conference on Machine Learning*, pages 3319–3328. PMLR.

Tay, Y., Bahri, D., Zheng, C., Brunk, C., Metzler, D., and Tomkins, A. (2020). Reverse engineering configurations of neural text generation models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 275–279.

Terrell, G. R. and Scott, D. W. (1992). Variable kernel density estimation. *The Annals of Statistics*, pages 1236–1265.

Uchendu, A., Le, T., Shu, K., and Lee, D. (2020). Authorship attribution for neural text generation. In *Conf. on Empirical Methods in Natural Language Processing (EMNLP)*.

Uchendu, A., Ma, Z., Le, T., Zhang, R., and Lee, D. (2021). Turingbench: A benchmark environment for turing test in the age of neural text generation. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 2001–2016.

Vijayaraghavan, S., Wang, Y., Guo, Z., Voong, J., Xu, W., Nasseri, A., Cai, J., Li, L., Vuong, K., and Wadhwa, E. (2020). Fake news detection with different models. *arXiv preprint arXiv:2003.04978*.

Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., and Bowman, S. (2018). Glue: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355.

Xu, X. and Li, J. (2020). Concreteness/abstractness ratings for two-character chinese words in meld-sch. *PloS one*, 15(6):e0232133.

Zellers, R., Holtzman, A., Rashkin, H., Bisk, Y., Farhadi, A., Roesner, F., and Choi, Y. (2019). Defending against neural fake news. In *NeurIPS*.