BitAbuse: A Dataset of Visually Perturbed Texts for Defending Phishing Attacks

Hanyong Lee[1], **Chaelyn Lee**[2], **Yongjae Lee**[3], **Jaesung Lee**[1,*]
[1] Department of Artificial Intelligence, Chung-Ang University, Seoul, South Korea
[2] Korea Electronics Technology Institute, Seongnam, South Korea
[3] Retrvr Inc., Seongnam, South Korea
glhy0718@gmail.com mylynchae@keti.re.kr ylee@retrvr.com curseor@cau.ac.kr

Abstract

Social engineering attacks, such as phishing, often target victims through visually perturbed texts to bypass security systems. The noise contained in these texts functions as an adversarial attack, designed to deceive language models and hinder their ability to accurately interpret the content. However, since it is difficult to obtain sufficient phishing cases, previous studies have used synthetic datasets that do not contain real-world cases. In this study, we propose the BitAbuse dataset, which includes real-world phishing cases, to address the limitations of previous research. Our dataset comprises a total of 325,580 visually perturbed texts. The dataset inputs are drawn from the raw corpus, consisting of visually perturbed sentences and sentences generated through an artificial perturbation process. Each input sentence is labeled with its corresponding ground truth, representing the restored, non-perturbed version. Language models trained on our proposed dataset demonstrated significantly better performance compared to previous methods, achieving an accuracy of approximately 96%. Our analysis revealed a significant gap between real-world and synthetic examples, underscoring the value of our dataset for building reliable pre-trained models for restoration tasks. We release the BitAbuse dataset, which includes real-world phishing cases annotated with visual perturbations, to support future research in adversarial attack defense. Our code and datasets are available at https://github.com/CAU-AutoML/Bitabuse.

1 Introduction

Social engineering attacks, including phishing, spam, pretexting, baiting, and tailgating, aim to leak confidential information by exploiting the psychological vulnerabilities of victims (Salahdine and Kaabouch, 2019). Among them, phishing often attacks victims through texts of email, SMS, and URLs. Specifically, these phishing techniques bypass security systems such as spam filtering using visually perturbed (VP) text (Deng et al., 2020; Julis and Alagesan, 2020; Boucher et al., 2022; Unicode Consortium, 2022), in which other characters, typically homoglyphs, replace a part of the characters in the text if the source language is English, that are nearly identical in appearance to the original characters[1]. For example, modifying 'Bitcoin' to 'ßitcöĭn' is an example of this technique.

Because phishing attacks based on VP texts can be prevented by restoring them to the original texts, most studies (Suzuki et al., 2019; Sawabe et al., 2019; Pruthi et al., 2019; Imam et al., 2022; Keller et al., 2021) focused on devising an restoration method. Specifically, they modified a non-VP text dataset into a VP text dataset based on their own heuristic rules and then evaluated the performance of their restoration methods based on the synthesized dataset. These approaches are effective for identifying the weaknesses of the restoration methods, but their analysis may be biased toward their own rules because the dataset is created without regard to real-world VP texts. For example, Viper (Eger et al., 2019) always perturbs a fixed portion of characters in a sentence, which is unrealistic. Furthermore, LEGIT (Seth et al., 2023) annotates the legibility of synthetic VP text and introduces a dataset by generating VP text that is applicable to real-world scenarios through a model that ranks transformations according to their readability. Nevertheless, research on VP text in real-world settings remains unexplored.

Although the data synthesizing strategy is helpful in circumventing the difficulty due to the lack of publicized real-world VP texts regarding phishing attacks, building a language model (LM)-based

[*] Corresponding author
WARNING: This paper contains offensive examples.

[1] We will indicate such character as VP character subsequently. Similarly, VP words, VP sentences, and VP texts mean words containing VP characters, sentences containing VP words, and texts containing VP sentences, respectively.

Findings of the Association for Computational Linguistics: NAACL 2025, pages 4367–4384
April 29 - May 4, 2025 ©2025 Association for Computational Linguistics

system for defending against phishing attacks only based on the synthesized dataset may be risky because there can be a gap between real-world and simulation. We argue that one way to achieve this limitation is to mix the real VP texts with the synthesized VP texts. In this case, it may be preferable that the original texts of the synthesized ones come from the same source for domain consistency. To achieve this, we propose a new dataset, namely `BitAbuse`, for defending phishing attacks.

Our contribution can be summarized as follows. First, based on 262,258 phishing-related emails identified from bitcoinabuse[.]com (Bitcoin Abuse, 2023), we created a raw corpus containing 325,580 sentences comprising 26,591 VP sentences and 298,989 non-VP English sentences. Second, based on the corpus, we created three datasets: `BitCore`, `BitViper`, and `BitAbuse`. Third, to depict the characteristics of our dataset, we conducted pilot studies using popular methods in this field and then compared their efficacy. We made the datasets of phishing attacks publicly available[2].

2 Related Work

In the studies involving VP texts, obtaining sufficient data is often difficult because VP texts, usually delivered as spam emails, are not widely shared on the web. In particular, there is a lack of datasets that reflect actual phishing attack situations, and existing datasets are only valid under specific conditions or environments (Elsayed and Shosha, 2018; Suzuki et al., 2019; Yazdani et al., 2020; Almuhaideb et al., 2022), such as internationalized domain names (IDNs.) As a result, conventional studies typically included a data synthesizing procedure with the method for restoring VP texts. Specifically, the dataset for testing the efficacy of their VP text restoration methods is synthesized by heuristic rules set in their own way.

Two notable studies regarding VP text data synthesizing are TextBugger (Li et al., 2019) and Viper. TextBugger is devised to generate VP texts using predefined homoglyph pairs and perturbation methods. Its goal is to degrade the performance of LMs by selecting characters in a text and replacing them with VP characters. This is useful for exposing vulnerabilities in security-sensitive tasks such as sentiment analysis (Pang and Lee, 2008) or malicious content detection (Hou et al., 2010). Viper searches for homoglyphs and generates VP texts based on embedding techniques. This method modifies the dataset by replacing characters in the text with VP characters and induces visual disturbance based on the replacement probability.

Regarding the restoration of VP texts, conventional methods first restore malicious text using SimChar DB-based (Suzuki et al., 2019), OCR-based (Sawabe et al., 2019), Spell Checker-based (Imam et al., 2022), or LM-based methods (Keller et al., 2021) and then detect malicious texts. The SimChar DB-based method automatically collects homoglyphs from the Unicode character set to detect VP characters in IDNs and restores them using a predefined restoration table. OCR-based methods were investigated to detect phishing attacks that deceive users by putting VP characters in IDNs. This method recognizes VP characters as images and converts them into the original characters. The Spell Checker-based method aims to detect images containing malicious text distributed on social networks by considering deformed characters in the text as typos and restoring them using a spell checker. The restoration strategy that combines two LMs, BERT (Devlin et al., 2019) and GPT (Radford et al., 2018), was also considered (Keller et al., 2021).

A common drawback of conventional studies is that the datasets used for evaluating the restoration performance of phishing attacks contain no real VP texts. As a result, the restoration performance in real-world situations may be over/underestimated, and unstable pre-trained LM models can be obtained. In this study, we create a new dataset that can contribute to phishing attack studies by collecting VP texts used in bitcoinabuse[.]com.

3 `BitAbuse`

We collect VP texts used in phishing attacks from the bitcoinabuse[.]com (Bitcoin Abuse, 2023) website. The website bitcoinabuse[.]com is a platform where worldwide users can share content related to Bitcoin fraud, such as emails. The site provides data collected through user participation, making it easy to find phishing email bodies containing VP texts. Additionally, because users directly upload emails after masking personal information, it can be ensured that the data can be collected safely without privacy concerns.

We used 262,258 phishing-related emails collected from bitcoinabuse[.]com between May 16,

[2] https://huggingface.co/datasets/AutoML/bitaubse

Original VP sentence	Ground truth (VP characters are restored)
i am going to sęnd out your vidęo recording to evęry bit of yoũr contacts and you cān easily imāgine cőncerning the disgracę you will sęe.	i am going to send out your video recording to every bit of your contacts and you can easily imagine concerning the disgrace you will see.
ìf you want to prevent thìs, transfer 0.019 btc to my bìtcoìn wallet (ìn case you do not know how to do ìt, then wrìte to google: "buy a bìtcoìn").	if you want to prevent this, transfer 0.019 btc to my bitcoin wallet (in case you do not know how to do it, then write to google: "buy a bitcoin").

Table 1: Examples of VP sentences (Left) and manually restored sentences (Right) in the raw corpus (note that vowels such as ’a’, ’e’, ’i’, ’o’, and ’u’ are mainly used as VP, which are highlighted in gray).

2017, and January 15, 2022. Detailed statistics of the raw dataset are discussed in Appendix A.

Although our primary goal is to create a VP text dataset related to English texts, the emails collected from English-speaking countries also included non-English texts, so removing irrelevant emails was followed. However, using existing language detection models to classify emails written in English is challenging due to the presence of English VP text, and fully manual filtering is also impractical. Therefore, we utilized the BERT model (Devlin et al., 2019) with a fully connected classification layer trained to automatically classify English text. The BERT model used in this classifier has a hidden state size of 768, 12 hidden layers, and 12 attention heads. Among the 262,258 email texts, 16,598 were randomly chosen and manually labeled as English (10,024 texts) or non-English (6,574 texts), and these labeled email texts were used to train the classification model. The labeled dataset was exclusively divided into train, validation, and test sets with 13,444, 1,494, and 1,660 texts, respectively. We provide the detailed hardware specification and hyperparameter settings in Appendix B.

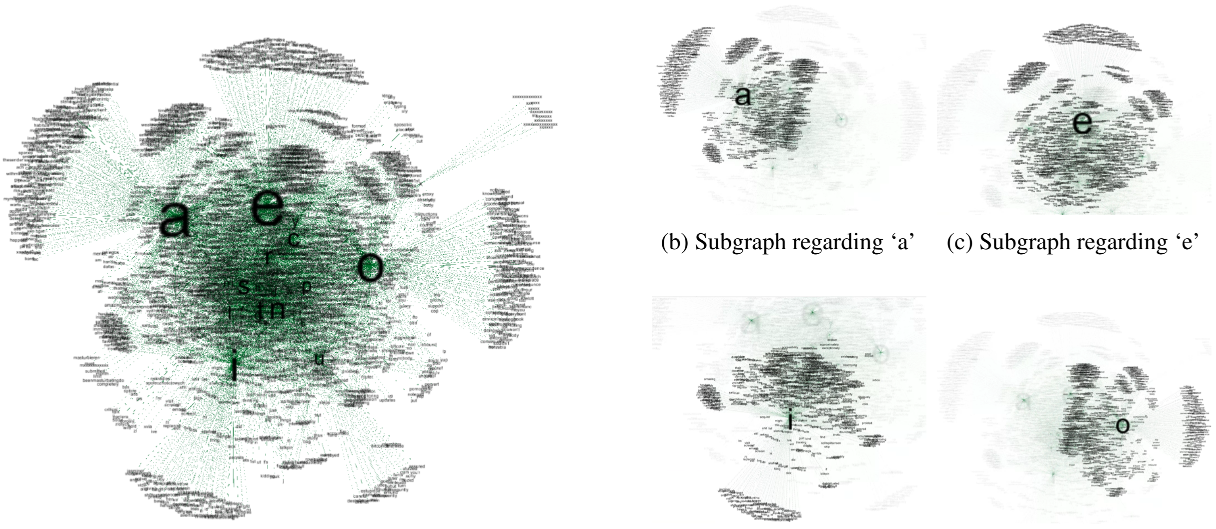
The classifier achieved an accuracy of approximately 99.28% on 1,660 uninvolved email texts in the training phase. The trained classifier removed 84,204 non-English email texts from the 262,258 ones, resulting in 178,054 email texts for further processing. Although this process significantly accelerates the preprocessing, non-English emails may still remain because the classification is imperfect. Such non-English sentences from those email texts are removed manually during a subsequent process that will be explained later.

After rough filtering 178,054 non-English email texts, we obtained 326,732 sentences by splitting the original texts with a maximum length of 512. Because those sentences include unnecessary components, such as random character sequences, we used a series of regular expressions to remove them efficiently. The list of regular expressions we used for further preprocessing and downloadable URL links are presented in Appendix C. We found that the raw dataset contains a wide range of VP characters not addressed in previous studies, such as control characters from U+0001 to U+0005, that will remain in BitCore and BitAbuse datasets.

To validate the restoration performance, we manually annotated the label for each character in the 326,732 sentences. Since manually annotating VP text is highly labor-intensive and inefficient, we extracted VP words from the VP text and manually created non-VP word labels for each VP word. These labels were then applied to the VP text to generate non-VP text labels. In cases where it was difficult to determine the label by looking only at the VP word, we referred to the original text to accurately annotate the corresponding non-VP word. While annotating, 1,152 irrelevant sentences, such as repetitive identical characters, random character sequences, or non-English sentences missed from the previous classification process, were removed. Table 1 shows examples of VP text with the corresponding ground truth sentences, and brief statistics of the raw corpus are presented in Appendix A. Also, We created BitCore, BitViper, and BitAbuse datasets based on the raw corpus. Brief statistics of the three datasets are presented in Table 10 of Appendix D.

4 Experimental Settings

We tested the restoration performance using Simchar DB (Suzuki et al., 2019), OCR (Sawabe et al., 2019), Spell Checker (Imam et al., 2022), Character BERT-based(El Boukkouri et al., 2020), and GPT-4o mini-based methods (OpenAI, 2023) in the viewpoint of three well-known evaluation mea-



(a) Complete graph of VP character-word association

(b) Subgraph regarding ‘a’ (c) Subgraph regarding ‘e’

(d) Subgraph regarding ‘i’ (e) Subgraph regarding ‘o’

Figure 1: Visualization of clustering based on VP character-word association of `BitCore` dataset. (a) overview of the obtained graph, (b) subgraph regarding VP characters of ‘a’, (c) subgraph regarding VP characters of ‘e’, (d) subgraph regarding VP characters of ‘i’, and (e) subgraph regarding VP characters of ‘o’

Measure	Dataset	Restoration Method				
		SimChar DB	OCR	Spell Checker	Character BERT	GPT-4o mini
Word Level Accuracy	`BitCore`	0.5515 ± 0.0036	0.6531 ± 0.0036	0.8909 ± 0.0016	**0.9984** ± 0.0004	0.7168 ± 0.0040
	`BitViper`	0.3373 ± 0.0006	0.3177 ± 0.0006	0.7133 ± 0.0011	**0.9534** ± 0.0008	0.5034 ± 0.0009
	`BitAbuse`	0.3547 ± 0.0010	0.3446 ± 0.0008	0.7275 ± 0.0012	**0.9568** ± 0.0006	0.5196 ± 0.0010
Word Level Jaccard	`BitCore`	0.6581 ± 0.0026	0.7255 ± 0.0022	0.8734 ± 0.0016	**0.9992** ± 0.0001	0.8966 ± 0.0023
	`BitViper`	0.4708 ± 0.0005	0.4617 ± 0.0005	0.6942 ± 0.0010	**0.9294** ± 0.0010	0.7963 ± 0.0007
	`BitAbuse`	0.4860 ± 0.0007	0.4830 ± 0.0007	0.7083 ± 0.0009	**0.9347** ± 0.0008	0.8037 ± 0.0005
BLEU	`BitCore`	0.8199 ± 0.0011	0.8860 ± 0.0011	0.9476 ± 0.0008	**0.9997** ± 0.0000	0.9328 ± 0.0025
	`BitViper`	0.7808 ± 0.0003	0.7748 ± 0.0002	0.8753 ± 0.0005	**0.9765** ± 0.0004	0.8919 ± 0.0004
	`BitAbuse`	0.7838 ± 0.0004	0.7836 ± 0.0004	0.8809 ± 0.0005	**0.9782** ± 0.0003	0.8947 ± 0.0004

Table 2: Comparison results of the five restoration methods in terms of three evaluation measures. Bold text indicates the best performance, and underlining indicates the second-best performance.

sures, such as Word Level Accuracy (Imam et al., 2022), Word Level Jaccard, and BLEU (Zeng et al., 2021). In addition, detailed information regarding the experiments, such as the model’s hyperparameters, is described in Appendix F.

4.1 Methods

We tested the restoration performance using five different methods. The SimChar DB-based method checks if there is an alphabetic homoglyph for each character in the Simchar Database and uses it to restore the homoglyph. The OCR-based method was implemented by applying OCR to each character and selecting the character with the highest probability. Spell Checker-based method entailed the segmentation of sentences into individual word units through a rule-based approach, followed by the restoration of each word using a spell checker based on Levenshtein Distance, as documented in the corresponding references (Norvig, 2016; Lison and Tiedemann, 2016).

Character BERT The Character BERT-based method employs a BERT model that processes token sequences at the character level to restore VP characters, inferring them as the original characters through the context of individual characters. In this approach, instead of relying on a standard subword tokenizer-based BERT—which is less effective when tokens contain perturbed characters—a character-level sequence approach is adopted for both input and output. This method is particularly important because attackers often modify characters within tokens to deceive victims, leading to widespread perturbations across most tokens. Stan-

Method	Example 1	Example 2
Original Text	afteɾ tʜ ɑt, i hąvɇ stąɾtɇd trɑcƙing yσυɾ înt ɇ ɾ п ɇt ɑctîvítíɇs.	I W┃ʟι ժeʟeTe eVeɦʏτhιɴɢ L've ɢoT aʙ0∪T ʏo∪.
SimChar DB	after tʜ ɑt, i havɇ startɇd trɑcking yσur intɇrnɇt ɑctivitiɇs.	I W┃ʟι ժeʟeTe eVeɦʏτhιɴo L've ooT ae0∪T ʏo∪.
OCR	after that, i hqva stqrtad tracking your tntarnat acttvítías.	i w┃ll ժelete evehythlng l've got ab0ut you.
Spell Checker	after that, i have started tracking your <None> <None>.	i will delete everything love got about you.
Character BERT	after that, i have started tracking your internet activities.	i will delete everything i've got about you.
GPT-4o mini	after that, i have started taking about internet activities.	i will let you know about what you.

Table 3: Restoration examples of the five methods where "<None>" indicates that the method failed to restore the word (VP and incorrectly restored characters are highlighted in gray color and underlined, respectively.)

dard BERT's Masked Language Model (MLM) mechanism, which relies on contextual information from surrounding tokens, struggles in such cases because the context tokens themselves may also be perturbed. Experiments with the Character BERT-based model involve training to restore and output the input VP sentence from three datasets into the corresponding restored sentence. In this process, both the input and output are sequences of character-level tokens. The training process of Character BERT was configured with a learning rate of 5×10^{-5}, a batch size of 32, and ten training epochs. Additionally, the AdamW optimizer was used with settings of $\beta_1 = 0.9$, $\beta_2 = 0.999$, and a weight_decay $= 0$, along with a linear learning rate scheduler. The experiment shown in Table 4 uses the same hyperparameters as the previously mentioned experiment, except the number of training epochs is set to 20.

We also employed the GPT-4o mini model to assess the performance of the latest large language model on the BitAbuse. GPT-4o mini is a closed-source generative language model, and the experiment was conducted via OpenAI's inference API. To leverage the model, we designed a prompt, as detailed in Table 15 of Appendix F.

4.2 Evaluations

We evaluated each method using the three measures that were used in previous VP text restoration studies: Word Level Accuracy, Word Level Jaccard, and BLEU. Word Level Accuracy is a measure that evaluates whether the restored word matches at each word position. When N_c represents the number of correctly restored words and N represents the total number of words in each sentence, Word Level Accuracy is calculated as

$$\text{Word Level Accuracy} = \frac{N_c}{N}.$$

The Word Level Jaccard score is calculated by forming the word set W_p from the predicted sentence and the word set W_l from the labeled sentence and then computing the ratio of the size of their intersection to the size of their union. Specifically, the Word Level Jaccard score is defined as

$$\text{Word Level Jaccard} = \frac{|W_p \cap W_l|}{|W_p \cup W_l|}.$$

The BLEU score is calculated by constructing the character sequences C_p of the predicted sentence and the character sequences C_l of the labeled sentence and then calculating the precision of the n-grams of the two sequences by

$$\text{BLEU} = \text{B} \times \exp\left(\sum_{n=1}^{N} w_n \log p_n\right).$$

where N and w_n are the maximum length and the weight of the n-grams, respectively. p_n represents the precision of the n-grams in C_l and C_p, and B is the brevity penalty used in the BLEU score calculation. In this paper, $N = 4$ is used to calculate the BLEU score, and $w_n = 1/N$ is set. The brevity penalty follows the standard BLEU score calculation method.

Without specific mentions, among sentences of BitAbuse dataset, 60%, 20%, and 20% of them were used for training, validation, and testing, respectively. The performance of each method was evaluated on the test set, and the average performance was measured by repeating the experiments with ten random training and test set splits.

5 Experimental Results

We conducted exploratory data analysis on VP words, VP characters, ratios, and so on that may help devise an effective methodology for defending phishing attacks.

Next, the number of VP sentences according to the occurrence ratio of VP characters to sentence length in the BitCore, BitViper, and BitAbuse datasets is presented as a histogram as in Figure 3 in Appendix D. Figure 1 shows the VP character-word association graph using the Yifan Hu algorithm (Hu, 2005) in Gephi (Bastian et al., 2009). This graph represents the clustering of the VP character-word association of BitCore dataset, where nodes correspond to characters and words subjected to perturbation attacks. The distance between nodes indicates the degree of their relatedness[3]. Figure 1(a) illustrates the overall graph, and Figures 1(b)–(e) each represents the graph regarding key characters. Notably, the core of the major clusters is occupied by vowels such as 'a', 'e', 'i', and 'o'. This likely occurs because vowels are frequently used across various words, resulting in strong associations within the graph and positioning them at the center. Specifically, the character 'e' appears to play a more global role within the graph, whereas other characters show stronger relations with words belonging to different clusters.

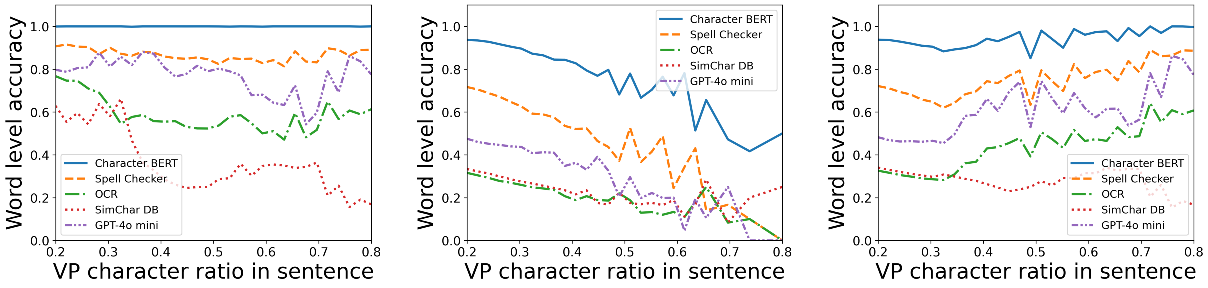
Table 2 shows the restoration performance of SimChar DB, OCR, Spell Checker, Character BERT, and GPT-4o mini-based methods on three datasets. Experimental results indicate that the Character BERT-based method significantly outperforms the other three methods. Regarding each dataset, all five methods achieved the best and worst performance for BitCore and BitViper datasets, respectively. Table 3 represents two examples of restoration results regarding five methods. Although the Character BERT-based method restores two VP sentence examples perfectly, Table 2 indicates that the restoration performance of Character BERT is imperfect. Table 5 lists VP words in three datasets that are incorrectly restored by the Character BERT-based method. The table shows that it often fails to restore if two or more VP characters are continued in the corresponding VP word.

We evaluated the Word Level Accuracy regarding the proportion of VP characters in sentences to validate the robustness of each method, as shown in Figure 2. In this experiment, the Character BERT-based method showed robust performance on both BitCore and BitAbuse datasets. It is interesting to note that it loses its robustness on BitViper dataset that does not include BitCore dataset, indicating that BitCore significantly contributes to the robust performance of the Character BERT-based method. In summary, the Character BERT-based method showed the most robust performance for VP sentences with a high VP character ratio. To see experimental results regarding Word Level Jaccard and BLEU, please refer to Appendix G.

Test performance with VP characters unseen during the training phase can be critical for the Character BERT-based method. Additional experiments were conducted using the Character BERT-based method with varying amounts of training VP sentences to validate this aspect. Specifically, when the amount of training VP sentences is extremely small, the Character BERT-based method encounters many unseen VP characters. Table 4 presents the performance of the Character BERT-based method when the proportion of training VP sentences is set to 1%, 5%, 10%, and 20%, respectively. In these experiments, we recognized that differences in the size of the test dataset could impact the fairness of performance comparisons. To address this, we sampled the remaining data, which were not used for training or validation, to standardize the size of the test dataset. Performance evaluation was conducted by measuring the performance of each pattern in the test dataset and calculating the mean and variance. This process was grounded in the Law of Large Numbers to include as many samples as possible, aiming to approximate the population mean.

The experimental results revealed that when the amount of training VP sentences was as low as 1% or 5%, significant performance degradation was observed for both the BitViper and BitAbuse

[3] The Yifan Hu algorithm uses a multiscale approach to position highly related nodes close to each other while placing less related nodes further apart. This algorithm is fundamentally based on a force-directed layout, where nodes are arranged according to the forces of attraction and repulsion between them based on the frequency of association.



(a) BitCore dataset (b) BitViper dataset (c) BitAbuse dataset

Figure 2: Word Level Accuracy performance of each method regarding VP character ratio in each sentence

Measure	Dataset	Training / Validation / Test ratio (%)			
		1 / 20 / 79	5 / 20 / 75	10 / 20 / 70	20 / 20 / 60
Word Level Accuracy	BitCore	0.9704 ± 0.0047	0.9917 ± 0.0016	0.9957 ± 0.0003	**0.9975** ± 0.0007
	BitViper	0.5318 ± 0.0013	0.7085 ± 0.0518	0.8786 ± 0.0112	**0.9236** ± 0.0015
	BitAbuse	0.5632 ± 0.0035	0.7778 ± 0.0690	0.8963 ± 0.0035	**0.9315** ± 0.0020
Word Level Jaccard	BitCore	0.9759 ± 0.0032	0.9951 ± 0.0007	0.9976 ± 0.0003	**0.9986** ± 0.0002
	BitViper	0.4446 ± 0.0003	0.6192 ± 0.0576	0.8242 ± 0.0150	**0.8862** ± 0.0021
	BitAbuse	0.4861 ± 0.0018	0.7041 ± 0.0768	0.8491 ± 0.0048	**0.8979** ± 0.0024
BLEU	BitCore	0.9923 ± 0.0013	0.9984 ± 0.0003	0.9992 ± 0.0001	**0.9996** ± 0.0001
	BitViper	0.7624 ± 0.0003	0.8563 ± 0.0259	0.9399 ± 0.0053	**0.9618** ± 0.0007
	BitAbuse	0.7803 ± 0.0009	0.8907 ± 0.0344	0.9485 ± 0.0017	**0.9656** ± 0.0009

Table 4: Comparison results of Character BERT-based restoration in terms of three evaluation measures with different amounts of training set

Dataset	Original VP word	Ground Truth	Restored Word
BitCore	Ɓîtcoin	bitcoin	rktcoin
	ƁîţÇoin	bitcoin	akomoin
	Àďďresś	address	uddress
	gòog	good	goog
BitViper	becɵme	became	become
	brụle	brute	broke
	ḥḛaṣt	beast	eeast
	selleı	seller	sellet
BitAbuse	trṷly	truly	trull
	⋈all	mail	mall
	seⱶÿioe	service	senside
	breakoùʟ	breakout	breakous

Table 5: List of VP words in three datasets that are incorrectly restored by the Character BERT-based methods from the experiments of Table 2 (VP characters are highlighted in gray color.)

datasets. This finding suggests that sufficient VP sentences are necessary to build a stable Character BERT model for VP text restoration. On the other hand, despite the performance drop with lower proportions of training data, the Word Level Accuracy still exceeded 0.5. This indicates that the model can restore relatively well from unseen attacks, even when it is exposed to many new VP attacks during the test phase. Additionally, using a smaller amount of training data allows the model to complete training more quickly, which is a desirable attribute in practical applications.

6 Discussion

Figure 2 demonstrates the performance of various restoration methods based on the proportion of VP characters. Compared to other methods, the Character BERT-based method performed more effectively as the proportion of VP characters increased. This indicates that the Character BERT-based method can accurately restore VP characters by leveraging contextual information. Conversely, the Spell Checker-based method exhibited a sharp decline in performance as the proportion of VP characters increased, highlighting the limitations of simply correcting typographical errors when dealing with text containing a high proportion of VP characters. The GPT-4o mini-based method underperformed compared to the spell checker-based method, likely because the GPT-4o mini is a generative model. Consequently, the word order and indexing between the input and output sentences are not maintained. This trait of generative language models seems to result in reduced performance in word accuracy evaluations, where word positioning is critical. Furthermore, the performance decline caused by the refusal responses triggered by the safety features of the language model, which will be discussed later, is also thought to have contributed to these results. Figures 4 and 5, like Figure 2, evaluate the Word Level Jaccard and BLEU performance for the ratio of VP characters in sentences. The Jaccard performance closely mirrored the Word Level Accuracy results, but the BLEU performance exhibited a slightly different pattern. In both Figures 2 and 4, the Character BERT-based method consistently demonstrated the most effective performance as the VP character ratio increased, with significant performance gaps between it and the other methods. However, in Figure 5, the performance of all methods, except for the Character BERT-based method, generally improved, reducing the performance gap. This suggests that the BLEU score is more sensitive to contextual accuracy, meaning that even if the exact words do not match, simpler methods can achieve higher scores as long as the sentence structure and meaning are somewhat preserved.

Table 2 presents the comparison results of five restoration methods across three datasets using three evaluation measures. The results show that the Character BERT-based method clearly outperformed the others, with all approaches achieving the highest performance on the BitCore dataset and the lowest performance on the BitViper dataset. Examples of the restoration for VP sentences through each method are shown in Table 3. Although the Character BERT, GPT-4o mini, and Spell Checker share the commonality of leveraging contextual information, the character BERT-based method was more accurate in the restoration. The SimChar DB-based method could only restore VP characters included in SimChar DB, and many of the VP characters that appeared did not exist in the DB, resulting in poor restoration performance. Additionally, a fundamental limitation of simple mapping-based methods like SimChar DB is their inability to handle one-to-many mappings for VP characters. Since these methods are rule-based, they can only output a single non-VP character for each VP character. We will demonstrate how frequently one-to-many corresponding VP characters appear in the dataset in Appendix H. The OCR-based method also had poor restoration capability for each VP character, and it was observed that character recognition was more difficult in the case of VP characters containing diacritics. The Spell Checker-based method showed high performance in restoring words containing VP characters, but it occasionally failed to find suitable words when the VP character ratio in the sentence was high. The GPT-4o mini-based method showed limited restoration capabilities. While it was able to successfully restore most VP characters in cases like Example 1, it failed when VP characters dominated the sentence, as in Example 2, producing outputs that differed significantly from the input. Additionally, in certain cases, due to the language model's safety features, responses such as "I'm sorry, but I can't assist with that," "I'm sorry, I can't assist with that," or "I'm sorry, I can't help with that" were generated in response to unethical content. These instances made up about 13.22% of the BitAbuse, which is a notable proportion. The Character BERT-based method excels by directly learning the context and succeeded in almost perfectly restoring VP words. This implies that models like BERT, which are significantly smaller than generative large language models such as GPT-4, can be more efficient for restoring VP text, as they still achieve high performance despite their smaller size. Table 5 provides examples of VP words that were incorrectly restored using the Character BERT-based method. The results indicate that restoration failures are more likely when VP characters appear consecutively or when there is a high density of attacked VP characters nearby.

As demonstrated by the comparison results in Table 2, the Character BERT-based method achieved nearly 100% accuracy on BitCore dataset, highlighting its robustness and reliability. In addition, with sufficient training VP sentences, it achieved almost perfect performance on BitAbuse dataset as shown in Table 4.

Given the high performance of LM pre-trained using BitAbuse, it may be employed in highly specialized, high-performance restoration tasks. For example, the pre-trained model could be applied in digital forensics to decode and reconstruct documents, emails, or logs that have been intentionally manipulated to obscure evidence. In addition, the model can be further trained to effectively handle even subtle and complex text modifications, which could improve forensic analysis. We believe that this model could also be used in secure messaging systems, where it would restore the original content of messages that have been deliberately obfuscated to ensure the secure transmission of sensitive information. These studies may highlight the potential of our datasets and pre-trained models to address critical challenges in secure communications.

7 Conclusion

In this study, we created three VP text datasets: `BitCore`, `BitViper`, and `BitAbuse`. Our analysis results show that `BitCore` and `BitViper` have significantly different characteristics, and the LM-based reconstruction method demonstrates strong robustness and potential on all three datasets. `BitAbuse`, a pre-trained model using 325,580 VP sentences, can be downloaded from `BitAbuse`.[4]. In future studies, a hybrid approach, such as combining OCR and Character BERT, can be explored to achieve robust performance with insufficient training samples. Internalizing them into LMs may be beneficial for remedying the greedy data consumption nature of LMs and in scenarios where collecting sufficient samples is challenging. In addition, lightweight yet accurate LMs for restoration tasks may be obtained if the bias to the words attacked frequently and vowel characters in real-world phishing attacks is exploited effectively. Lastly, validating the zero-shot performance of `BitAbuse` model should also be performed.

[4] https://huggingface.co/datasets/AutoML/bitaubse

Limitations

The VP text restoration experiments conducted in this study did not include additional restoration methods to avoid exceeding the scope of the study. Specifically, a performance comparison between the Character BERT-based and other LM-based restoration methods was not performed. Thus, it is difficult to evaluate the superiority of Character BERT over other modern LMs. Character BERT showed sufficiently good performance, but it will be possible to compare effectiveness and efficiency with methods applying other LMs in the future.

The BitAbuse dataset used in this study only includes data related to Bitcoin scams, which limits its ability to reflect a variety of phishing attack scenarios. In addition, phishing attacks may appear in more diverse or complex forms over time, and failure to reflect this diversity may reduce the generalizability of our study. Thus, future studies should aim to construct an extended dataset that includes various phishing attack scenarios and conduct studies comparing different restoration methods.

Also, our datasets were created for study purposes to defend against phishing attacks based on VP texts. However, there is a risk that this dataset could be used by non-experts in phishing to learn and execute attacks. For example, WormGPT, recently created on the dark web to generate criminal text, and PoisonGPT, released by Mithril Security, spread contaminated results. These models might use our datasets to develop malicious tools. Consequently, this could lead to the sophistication of phishing attacks, resulting in more victims. In addition, the damage caused by the misuse of such datasets is difficult to hold accountable legally. Currently, many countries lack clear regulations regarding the technological misuse of such datasets, necessitating careful considerations and observations. The datasets and models used in this paper are publicly available, but they should not be used for purposes other than research.

Ethics Statement

Our datasets were created for study purposes to defend against phishing attacks based on VP texts. However, there is a risk that this dataset could be used by non-experts in phishing to learn and execute attacks. For example, WormGPT, recently created on the dark web to generate criminal text, and PoisonGPT, released by Mithril Security, spread contaminated results. These models might use our datasets to develop malicious tools. Consequently, this could lead to the sophistication of phishing attacks, resulting in more victims. In addition, the damage caused by the misuse of such datasets is difficult to hold accountable legally. Currently, many countries lack clear regulations regarding the technological misuse of such datasets, necessitating careful considerations and observations. The datasets and models used in this paper are publicly available, but they should not be used for purposes other than research.

Acknowledgements

This research was supported by the Chung-Ang University Graduate Research Scholarship in 2024 and by the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (2021-0-01341, Artificial Intelligence Graduate School Program (Chung-Ang University)).

This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. RS-2024-00402898, Simulation-based High-speed/High-Accuracy Data Center Workload/System Analysis Platform)

References

Alan Akbik, Tanja Bergmann, Duncan Blythe, Kashif Rasul, Stefan Schweter, and Roland Vollgraf. 2019. FLAIR: An easy-to-use framework for state-of-the-art NLP. In *NAACL 2019, 2019 Annual Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 54–59.

Abdullah M. Almuhaideb, Nida Aslam, Almaha Alabdullatif, Sarah Altamimi, Shooq Alothman, Amnah Alhussain, Waad Aldosari, Shikah J. Alsunaidi, and Khalid A. Alissa. 2022. Homoglyph attack detection model using machine learning and hash function. *Journal of Sensor and Actuator Networks*, 11(3).

Satish B. 2024. satbyy/go-noto-universal. Original-date: 2021-12-10T17:48:27Z.

Tyler Barrus. 2024. barrust/pyspellchecker. Original-date: 2018-02-24T01:21:50Z.

Mathieu Bastian, Sebastien Heymann, and Mathieu Jacomy. 2009. Gephi: an open source software for exploring and manipulating networks. *ICWSM*, 3:361–362.

Bitcoin Abuse. 2023. Bitcoin Abuse Database. https://www.bitcoinabuse.com. [Online; accessed 30-April-2023].

Nicholas Boucher, Ilia Shumailov, Ross Anderson, and Nicolas Papernot. 2022. Bad characters: Imperceptible nlp attacks. In *2022 IEEE Symposium on Security and Privacy (SP)*, pages 1987–2004.

Simon Cozens, The Noto Project Authors, and Google Font Contributors. 2024. notofonts/runic. Original-date: 2022-06-20T22:10:47Z.

Perry Deng, Cooper Linsky, and Matthew Wright. 2020. Weaponizing unicodes with deep learning - identifying homoglyphs with weakly labeled data. In *2020 IEEE International Conference on Intelligence and Security Informatics (ISI)*, pages 1–6.

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.

Steffen Eger, Gözde Gül Şahin, Andreas Rücklé, Ji-Ung Lee, Claudia Schulz, Mohsen Mesgar, Krishnkant Swarnkar, Edwin Simpson, and Iryna Gurevych. 2019. Text processing like humans do: Visually attacking and shielding NLP systems. 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 1634–1647, Minneapolis, Minnesota. Association for Computational Linguistics.

Hicham El Boukkouri, Olivier Ferret, Thomas Lavergne, Hiroshi Noji, Pierre Zweigenbaum, and Jun'ichi Tsujii. 2020. Characterbert: Reconciling elmo and bert for word-level open-vocabulary representations from characters. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6903–6915, Barcelona, Spain (Online).

Yahia Elsayed and Ahmed Shosha. 2018. Large scale detection of idn domain name masquerading. In *2018 APWG Symposium on Electronic Crime Research (eCrime)*, pages 1–11.

Yung-Tsung Hou, Yimeng Chang, Tsuhan Chen, Chi-Sung Laih, and Chia-Mei Chen. 2010. Malicious web content detection by machine learning. *Expert Systems with Applications*, 37(1):55–60.

Yifan Hu. 2005. Efficient, high-quality force-directed graph drawing. In *10th International Mathematica Symposium*, pages 37–71, Banff, Canada.

Niddal H. Imam, Vassilios G. Vassilakis, and Dimitris Kolovos. 2022. Ocr post-correction for detecting adversarial text images. *Journal of Information Security and Applications*, 66:103170.

M Rubin Julis and S Alagesan. 2020. Spam detection in sms using machine learning through textmining. *Internatıonal Journal Of Scıentıfic & Technology Research*, 9(02).

Yannik Keller, Jan Mackensen, and Steffen Eger. 2021. BERT-defense: A probabilistic model based on BERT to combat cognitively inspired orthographic adversarial attacks. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 1616–1629, Online. Association for Computational Linguistics.

Matthias A. Lee. 2024. madmaze/pytesseract. Original-date: 2010-10-27T23:02:49Z.

Jinfeng Li, Shouling Ji, Tianyu Du, Bo Li, and Ting Wang. 2019. Textbugger: Generating adversarial text against real-world applications. In *26th Annual Network and Distributed System Security Symposium, NDSS 2019*, 26th Annual Network and Distributed System Security Symposium, NDSS 2019. The Internet Society. Publisher Copyright: © NDSS 2019.All rights reserved.; 26th Annual Network and Distributed System Security Symposium, NDSS 2019 ; Conference date: 24-02-2019 Through 27-02-2019.

Pierre Lison and Jörg Tiedemann. 2016. OpenSubtitles2016: Extracting large parallel corpora from movie and TV subtitles. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 923–929, Portorož, Slovenia. European Language Resources Association (ELRA).

P. Norvig. 2016. How to write a spelling corrector. `https://norvig.com/spell-correct.html`. [Online; accessed 4-May-2023].

OpenAI. 2023. Gpt-4 technical report.

Bo Pang and Lillian Lee. 2008. Opinion mining and sentiment analysis. *Foundations and Trends® in Information Retrieval*, 2(1–2):1–135.

Danish Pruthi, Bhuwan Dhingra, and Zachary C. Lipton. 2019. Combating adversarial misspellings with robust word recognition. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5582–5591, Florence, Italy. Association for Computational Linguistics.

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

Fatima Salahdine and Naima Kaabouch. 2019. Social engineering attacks: A survey. *Future Internet*, 11(4).

Yuta Sawabe, Daiki Chiba, Mitsuaki Akiyama, and Shigeki Goto. 2019. Detection method of homograph internationalized domain names with ocr. *Journal of Information Processing*, 27:536–544.

Dev Seth, Rickard Stureborg, Danish Pruthi, and Bhuwan Dhingra. 2023. Learning the legibility of visual text perturbations. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 3260–3273, Dubrovnik, Croatia. Association for Computational Linguistics.

Hiroaki Suzuki, Daiki Chiba, Yoshiro Yoneya, Tatsuya Mori, and Shigeki Goto. 2019. Shamfinder: An automated framework for detecting idn homographs. In *Proceedings of the Internet Measurement Conference*, IMC '19, page 449–462, New York, NY, USA. Association for Computing Machinery.

Unicode Consortium. 2022. Unicode Utilities: Confusables. `https://www.unicode.org/Public/security/15.0.0/confusables.txt`. [Online; accessed 30-April-2023].

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Ramin Yazdani, Olivier van der Toorn, and Anna Sperotto. 2020. A case of identity: Detection of suspicious idn homograph domains using active dns measurements. In *2020 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, pages 559–564.

Guoyang Zeng, Fanchao Qi, Qianrui Zhou, Tingji Zhang, Zixian Ma, Bairu Hou, Yuan Zang, Zhiyuan Liu, and Maosong Sun. 2021. OpenAttack: An open-source textual adversarial attack toolkit. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: System Demonstrations*, pages 363–371, Online. Association for Computational Linguistics.

Category	Value
Number of email texts	262,258
Min. length of email text	10
Max. length of email text	2,000
Average length of email text	417

Table 6: Brief statistics of phishing emails collected from bitcoinabuse[.]com

Category	Value
Number of sentences	325,580
Number of VP sentences	26,591
Number of non-VP sentences	298,989
Average length of sentences	91

Table 7: Brief statistics of the raw corpus

Description	Regular Expression		Description	Regular Expression	
Miscellaneous symbols	`[\u 260e\u 2610\u 2611\u 261e\u 2620\u 2639\u 2640\u 2642\u 2661\u 2665\u 267b\u 26a0\u 26d4]`	R	Dingbats	`[\u 2705\u 270a\u 270c\u 270d\u 2714\u 2757\u 2764\u 2795\u 2797\u 27a1]`	R
General punctuations and formatting characters	`[\u 200b-\u 200d\u 2022\u 202a\u 2028\u 2039\u 203a\u 2060-\u 2069]`	R	Emoticons, HTML tag patterns, and special character sequences	¯\_\(ツ\)_\/¯ | & #8203; | </?sp.n>| \?\ u200d [♀♂] | * †† * *	R
Latin supplements	`[\u 00a7\u 00a9\u 00ab-\u 00ae\u 00b0\u 00b7\u 00bb\u 00bf]`	R	Control characters	`[\u 0000\u 0006-\u 0008\u 000b-\u 001f\u 0080-\u 009f]`	R
Bitcoin wallet address	`[13][a-km-zA-HJ-NP-Z1-9]{25,34}`	R	Email address	`[\w -\.]+@([\w -]+\.)+[\w -]{2,4}`	R
CJK characters	`[\u 3040-\u 9fff\u ac00-\u d7ff]`	R	Box elements / geometric shapes	`[\u 2592\u 25a0\u 25cb\u 25cf]`	R
Emoji etc.	`[\u 1f000-\u 1ffff]`	R	Private use area	`[\u e000-\u f8ff]`	R
Variation selectors	`[\u fe00-\u fe0f]`	R	Combining diacritical marks	`[\u 032a\u 034f]`	R
Arabic characters	`[\u 061c\u 0640]`	R	Sinhala characters	`[\u 0d9a\u 0dd4]`	R
Letter-like symbols	`[\u 2116\u 2122]`	R	Mathematical operators	`[\u 2211\u 22ef]`	R
Miscellaneous symbols and arrows	`[\u 2b07\u 2b55]`	R	Halfwidth and fullwidth forms	`[\u ff0a\u ff5e]`	R
Modifier letter up arrowhead	`[\u 02c4]`	R	Superscript six	`[\u 2076]`	R
Combining enclosing keycap	`[\u 20e3]`	R	Upwards arrow	`[\u 2191]`	R
Top half integral symbol	`[\u 2320]`	R	Zero width no-break space	`[\u feff]`	R
Special space characters	`[\u 00a0\u 2002-\u 200a\u 3000]`	S	Small quotation mark, accent mark, or prime symbol	`[\u 00b4\u 02bb\u 02cb\u 2018\u 2019\u 2032]`	'
Diaeresis, double quotation mark, or double prime symbol	`[\u 00a8\u 201c\u 201d\u 2033\u 275d\u 275e]`	"	Various types of hyphens, dashes, or the minus sign	`[\u 2010\u 2011\u 2013\u 2014\u 2015\u 2212]`	-
Low quotation mark or a fullwidth comma	`[\u 201a\u 201e\u ff0c]`	,	Double exclamation mark or a fullwidth exclamation mark	`[\u 203c\u ff01]`	!
Various types of left brackets	`[\u 300a\u 3010\u ff08]`	(	Various types of right brackets	`[\u 300b\u 3011\u ff09]`	)
Various types of equals sign	`[\u 2248\u ff1d]`	=	Horizontal ellipsis	`[\u 2026]`	...
Fullwidth colon	`[\u ff1a]`	:	Text decoding errors	€™	'
Fullwidth semicolon	`[\u ff1b]`	;		â €[œ] ?	"
Multiplication sign	`[\u 00d7]`	x			

Table 8: Preprocessed characters represented in their Unicode based on corresponding regular expressions

Original Text	Preprocessed Text
【Reminder】 Your system devices has been Hacked 【National Security Agency】 Authority-11622272	(Reminder) Your system devices has been Hacked (National Security Agency) Authority-11622272
After receiving the payment, I will delete the video ,	After receiving the payment, I will delete the video,
You may not know me a&#8203;nd y&#8203;ou are pro&#8203;ba&#8203;bly&#8203;	You may not know me and you are probably

Table 9: Example of text preprocessed using regular expressions. The red box with the number in it indicates the unprintable Unicode character of the hex value written inside it (Please see color PDF.)

Dataset	Number of VP Sentences	Average Length	Number of VP Words (%)	Unique VP Words	Number of VP Characters (%)	Unique VP Characters
BitCore	26,591	92	261,460 (58%)	37,726	503,239 (26%)	317
BitViper	298,989	91	2,861,434 (58%)	1,126,986	4,347,988 (20%)	525
BitAbuse	325,580	91	3,122,894 (58%)	1,160,211	4,851,227 (21%)	706

Table 10: Brief statistics of BitCore, BitViper, and BitAbuse datasets

Word	Number of Variants	Word	Number of VP attacked
your	369	you	15,103
access	361	your	10,725
email	293	to	7,745
software	286	and	7,626
Bitcoin	268	the	6,906
videos	266	a	5,277
video	265	I	4,005
have	254	have	3,781
transfer	230	this	3,685
bitcoin	227	video	3,576
internet	225	that	3,103
browsing	220	of	2,881
you	207	know	2,713
which	205	will	2,436
about	200	is	2,407
will	198	all	2,391
contacts	195	on	2,196
activities	191	what	2,163
relatives	187	contacts	1,850
with	185	as	1,794
social	173	with	1,779
devices	171	it	1,773
account	158	i	1,758
antivirus	156	software	1,674
tracking	155	email	1,654
watching	154	after	1,625
after	143	from	1,614
managed	142	access	1,510
know	137	part	1,430
from	135	site	1,365
also	134	videos	1,345
considering	134	in	1,247
virus	134	me	1,234
microphone,	131	are	1,233
deactivate	130	not	1,222
information	130	bitcoin	1,197
this	129	which	1,186
accounts	125	do	1,170
according	124	watching	1,159
received,	123	visited	1,148
away	122	payment	1,131
websites	122	can	1,119
masturbating	120	for	1,086
purchased	119	malware	1,056
gained	118	porn	1,029
signatures	118	don't	985
happen	117	account	976
installed	117	right	923
months	117	screen	848
simple	117	about	843

Table 11: The top 50 list of VP word variants and VP attacks for each word appearing in the `BitAbuse` dataset.

A Statistics of Raw Dataset

Table 6 shows brief statistics of the collected email texts. We identified 262,258 phishing-related emails from bitcoinabuse[.]com between May 16, 2017, and January 15, 2022, and extracted the text bodies of these emails. The length of the email bodies averages about 417 characters, ranging from a minimum of 10 characters to a maximum of 2,000 because the platform limits the maximum number of characters to 2,000. The content of phishing-related emails was uploaded from approximately 224 countries, and the country of upload and the language of the collected text may differ.

Word	VP words
your	yøur, γoυг, γoυɾ, ɣoυг, γόυг, γoμг, yŏur, yőũr, γoυг, Y0Uh, . . .
access	αcceʂs, ἀcceṡṡ, accêss, access, accėss, access, άcceśś, . . .
email	eмaιl, éмaỉI, émãïl, ëmaïl, eмaíl, êṁaὶl, eḿαίI, ëmaíI, éḿáὶl, . . .
software	softwαгe, softwαre, softwäre, sσftwαгe, sόḟtẃἀгé, sòftwarè, . . .
Bitcoin	Bỉtcoỉп, Bítcoíп, Bίtcσỉп, Bιtcoιη, Bίtcoίп, βιtcoιη, . . .

Table 12: Examples of VP variants regarding five words of Table 11 with the highest number of variants (VP characters are highlighted in gray color.)

Table 7 presents the statistics of the raw corpus after splitting the collected texts into individual sentences and removing meaningless texts, as mentioned in the Data Collection section. The sentence-splitting process was performed using the NLTK library, resulting in a total of 325,580 sentences. In the next step, sentences containing non-ASCII characters were classified as VP sentences, and the classification was manually reviewed to ensure accuracy. After the review, 26,591 sentences were identified as VP sentences, while 298,989 were categorized as non-VP sentences.

B Filtering Non-English Texts

In our study, we exploited the BERT model with a fully connected classification layer trained to classify English texts from non-English texts. To train our model, we use the Flair library (Akbik et al., 2019). In addition, the learning rate was set to $1e-6$, with 1 learning epoch (the library early stopped training due to the very small learning rate), a batch size of eight, an AdamW optimizer ($\beta_1 = 0.9$, $\beta_2 = 0.999$, weight_decay $= 0$), and the AnnealOnPlateau scheduler implemented in the Flair library. Additionally, a single NVIDIA GeForce RTX 3080 GPU was used.

Char.	Number of Variants	Char.	Number of VP attacked
i	34	o	179,792
a	28	i	145,832
o	27	c	95,863
e	22	a	85,905
u	22	e	57,009
r	20	n	38,135
m	14	t	38,085
n	14	d	37,178
p	13	l	34,865
s	13	u	14,840
b	13	r	14,381
t	12	s	14,202
c	12	p	13,717
w	12	v	9,788
k	12	y	9,073
y	10	h	7,313
h	9	k	5,966
l	8	m	5,349
d	8	g	3,452
j	7	f	3,273
v	6	w	2,855
g	6	b	2,413
x	5	x	480
q	5	q	220
f	3	j	107
z	3	z	10
0	2	0	6

Table 13: The full list of VP character variants and VP attacks for each character appearing in the BitAbuse dataset.

C Regular Expressions

Table 8 shows the list of regular expressions we used for further preprocessing. The first and fourth, the second and fifth, and the third and sixth columns mean the description of characters, regular expressions, and replaced characters, respectively. R and S in the third and sixth columns mean “Removed” and “Space”. For example, No-break Space (U+00A0), En Space (U+2002), Hair Space (U+200A), and Ideographic Space (U+3000) are special space characters and would commonly be replaced with regular space characters. The space after \u in the regular expression is included intentionally for clarity but is excluded in the actual regular expression. We also release a downloadable list of regular expressions and preprocessing code from https://huggingface.co/datasets/AutoML/bitaubse/blob/main/preprocessing.py.

Table 9 shows example sentences after the preprocessing based on the regular expressions. In three examples of the table, emojis and special characters in the sentence are removed, and unusual characters are replaced with ASCII characters with the same meaning. For example, in the first example in the table, “Left Black Lenticular Bracket (U+3010)” and “Right Black Lenticular Bracket (U+3011)” were replaced with regular parentheses (U+0028, U+0029). In the second example, unprintable Unicode characters that are presented as a hex value in the red box are removed.

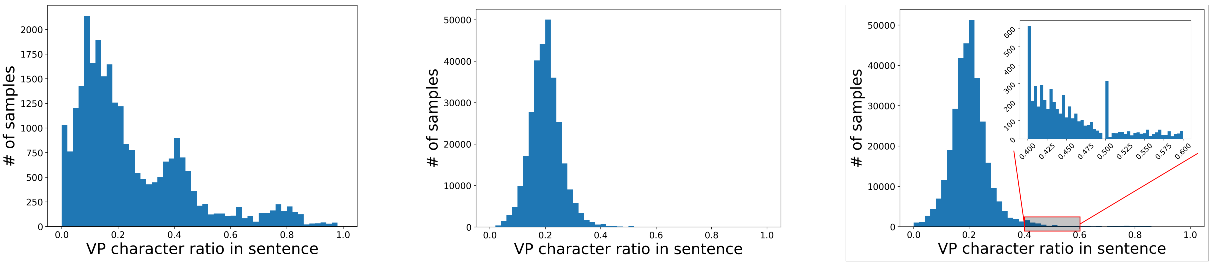
D Statistics of BitAbuse

We created BitCore, BitViper, and BitAbuse datasets based on the raw corpus. Brief statistics of the three datasets are presented in Table 10. Specifically, BitCore was created by simply selecting 26,591 VP sentences from the raw corpus. Next, BitViper was created by applying the character perturbation procedure of Viper that uses the ICEs method with a probability of 0.2 to 298,989 non-VP sentences of the raw corpus, following the same settings used in the original study for the restoration task[5]. Lastly, BitAbuse was created by merging BitCore and BitViper, resulting in the largest dataset of our study that contains both real-world and synthetic VP sentences.

E Statistics and Examples of VP Words and Characters

We summarized the number of VP word variants and that of attacks on each corresponding word appearing in VP texts in Table 11. VP word variants were frequently found in terms related to Bitcoin scam domains, such as “email”, “software”, “Bitcoin”, and “video”. In contrast, the words most often attacked were common words like “you”, “to”, “and”, “the”, and “a”. This indicates that these commonly used words are more likely to be targeted due to their frequent everyday use. Although domain-specific words exhibit a significant number of variants, their attack frequency is relatively low. This suggests that attackers are cautious

[5] TextBugger is not considered here because it attacks by altering keywords in sentences for semantic classification. Thus, applying TextBugger to non-VP texts of the raw corpus requires additional work, such as labeling whether a sentence is spam, which is out of the scope of this study.



(a) BitCore dataset (Real)

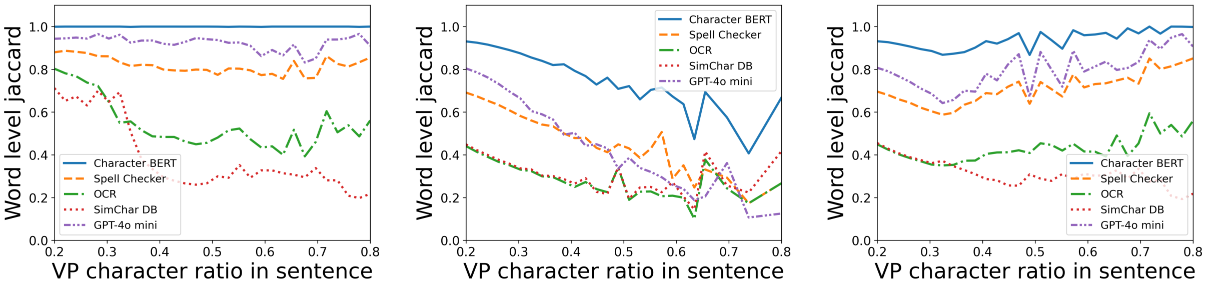
(b) BitViper dataset (Synthesized)

(c) BitAbuse dataset (Mixed)

Figure 3: The histogram of the number of VP sentences BitCore (26,591 sentences), BitViper (298,989 sentences), and BitAbuse datasets (325,580 sentences) according to the occurrence ratio against the sentence length.

Peak	VP sentence examples
0.07 - 0.09	if you do nōt fund thÍs bitcōin address with $1000 withÍn nĕxt 2 days, i will contact yoũr relativĕs ánd ĕverybōdy on yōũr contact lists ánd show them your recordÍngs.
0.32 - 0.34	rιghτ ɑfτer τhɑτ, my sofτwɑre obτɑιηed your comρleτe cοηταcτs from your messeηger, fɑcebοοκ, ɑs well ɑs emɑιlɑccouητ.
0.66 - 0.68	ỉ ȧIśȯ promíśè tȯ dëáctívàte ȧṅd dëIėte ȧIl thë hȧrṁful śoḟtwȧre ḟrȯṁ your dëvìceś ȧṅd thê ṗrịce ịś rêIatìveIɣ Iow, coṅsìdėring thát í hȧvè bėèń ćhêcκìńġ ȯut ɣour trȧḟḟìc ḟor sȯṁê tìḿê Ƅy ńoш.

Table 14: VP sentence examples of the three peaks in the histogram of BitCore dataset shown in Figure 3(a) (VP characters are highlighted in gray color.)

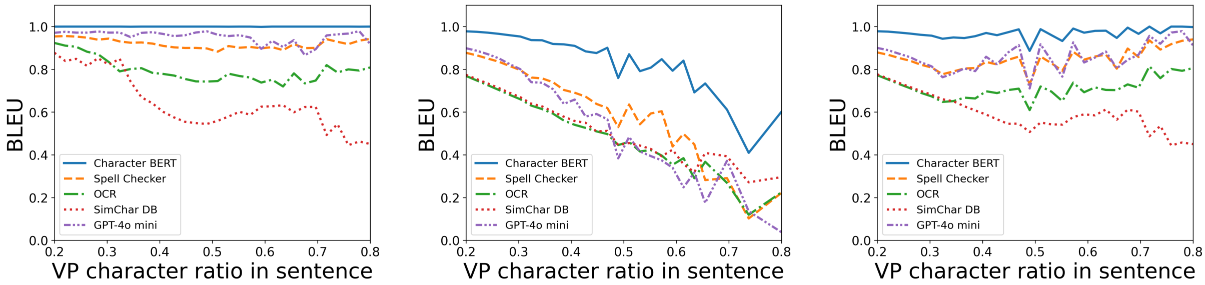


(a) BitCore dataset

(b) BitViper dataset

(c) BitAbuse dataset

Figure 4: Word Level Jaccard performance of each method regarding VP character ratio in each sentence



(a) BitCore dataset

(b) BitViper dataset

(c) BitAbuse dataset

Figure 5: BLEU performance of each method regarding VP character ratio in each sentence

about excessively altering key semantic words to avoid disrupting the overall context. In addition, our analysis indicates that the restoration model may put more weight on training common words to be an effective VP text restoration. Moreover, the presence of a wide range of variants necessitates that the restoration model is capable of handling these diverse perturbation attacks. Thus, domain-specific knowledge of source language may also be incorporated to achieve accurate training.

Table 12 shows examples of VP variants for five words from Table 11 with the highest number of variants. The left column lists the original words, while the right column displays the VP characters used to perturb the original words, including unusual cases with control characters like U+0001 to U+0005. These characters can be rendered as graphic symbols depending on the environment, suggesting their use in VP attacks. The examples highlight the variability of VP characters applied to the same word, resulting in multiple non-duplicated variations, such as “access” being perturbed into forms like “αcceșs” or “accêss”. Additionally, within the same word, certain characters (e.g., “c” and “s” in “access”) may appear as both VP and non-VP characters, indicating that attackers likely apply perturbations in a probabilistic rather than consistent manner.

Table 13 presents statistics on the number of VP character variants and the frequency of perturbation attacks in the `BitCore` dataset. The overall statistical results for this analysis are provided in Table 13. The number of variations for a single character, as shown in Table 13, is highest for ‘i’, followed by ‘a’, ‘o’, and ‘e.’ This aligns with the high connectivity of these characters in the VP character-word association graph visualized in Figure 1 This suggests that characters with more variations are broadly associated with a wide range of words. For example, the character ‘i’ has 33 variations and is strongly connected to various words in Figure 1, more so than other characters. This indicates the significant role that ‘i’ plays within VP sentences.

Figure 3 shows the histogram of the number of VP sentences according to the occurrence ratio of VP characters against the length of the sentence in `BitCore`, `BitViper`, and `BitAbuse` datasets, respectively. The x- and y-axes of each figure represent the ratio of VP characters included and the number of corresponding VP sentences, respectively. As shown in Figure 3(a), the VP sentences collected from bitcoinabuse[.]com does not yield unimodal distribution regarding the number of VP characters included. Rather, it has three peaks regarding the VP character ratio, such as 0.07 to 0.09, 0.32 to 0.34, and 0.66 to 0.68, that may be useful for devising VP restoration methods. Figure 3(b) representing the histogram of `BitViper` dataset indicates that its distribution of significantly different to that of `BitCore` dataset. Figure 3(c) shows the histogram of `BitAbuse` dataset. Table 14 lists VP sentence examples of the three peaks in the histogram of `BitCore` dataset shown in Figure 3(a).

Prompt
Restore the In Text to its original Out Text (Provide only output text): In Text: {vp text} Out Text:

Table 15: The prompt used in GPT-4o mini for the restoration experiment. “{vp text}” refers to the VP text to be restored.

We argued that the artificially synthesized datasets may have a gap to real phishing attack situations. For example, because Viper modifies a fixed ratio of characters in the sentence where the user sets the ratio value, all the modified sentences have approximately the same portion of VP characters as shown in Figure 3(b), which is not aligned with the observation given from Figure 3(a). The figure also indicates that there are three peaks, with prominent ones appearing between 0.07 and 0.09, 0.32 and 0.34, and 0.66 and 0.68. These peaks suggest that VP texts can be categorized into distinct groups. In Table 14, three VP sentences, each corresponding to each peak, are presented. The VP sentence associated with the first peak frequently contains vowels with added accents whereas that with the second peak exhibits a pattern of using Greek letters as VP characters. Lastly, the VP sentence related to the third peak contains the use of characters from various languages as VP characters, with a notable example being the substitution of the letter ‘h’ with the Armenian character “հ.”

F Experimental Details

We provide additional details on the experimental settings and methods used in the experiments.

F.1 Character BERT Based Method

In the experiment shown in Table 2, the training process of Character BERT was configured with a learning rate of 5×10^{-5}, a batch size of 32, and ten training epochs. Additionally, the AdamW optimizer was used with settings of $\beta_1 = 0.9$,

$\beta_2 = 0.999$, and a weight_decay $= 0$, along with a linear learning rate scheduler. The experiment shown in Table 4 uses the same hyperparameters as the previously mentioned experiment, except the number of training epochs is set to 20.

F.2 GPT-4o mini Based Method

When employing the GPT-4o mini model, we designed a prompt for VP text restoration, as detailed in Table 15. The experiment used OpenAI's batch API, with a total cost of approximately 3.47 USD.

F.3 Experimental Environment

The implementations were done by using the Pytesseract (Lee, 2024), Pyspellchecker (Barrus, 2024), and Transformers (Wolf et al., 2020) library. The experiment was performed on the computing hardware with an Intel i9-10980XE processor, two NVIDIA GeForce RTX 3090 GPUs, and 128GB of RAM. Additionally, the textual content was rendered using the Noto Sans Runic (Cozens et al., 2024) and GoNotoCurrent font (B, 2024).

G Performance regarding VP Character Ratio in Each Sentence

Figures 4 and 5 show the Word Level Jaccard and BLEU performance of each method regarding the VP character ratio in each sentence. As shown in Figure 2, the Character BERT-based method outperformed SimChar DB, OCR, and Spell Checker-based methods. Similar to the experimental results regarding Word Level Accuracy, the Character BERT-based method showed robust performance on both `BitCore` and `BitAbuse` datasets, whereas it loses its robustness on `BitViper` dataset that does not include `BitCore` dataset.

H Statistics of One-to-Many Corresponding VP Characters

As mentioned in the Discussion section, simple mapping-based methods like SimChar, used in the experiments, have a fundamental limitation in handling one-to-many VP character relationships, as they can only output a single non-VP character for each VP character. To verify this, we analyzed how frequently one-to-many VP characters appear in the dataset.

Table 16 lists VP characters, sorted by how often each one is mapped to different non-VP characters, showing that up to six options can arise when restoring a single VP character.

VP Character	Corresponding Characters
ɑ	a, o, u, d, g, q
o	o, c, d, g, q
ο	o, q, g, c, d
о	o, d, c, g, q
ᴏ	o, d, c, g, q
օ	d, g, c, q
ọ	o, c, d, q
ḽ	i, l, j, k
ḷ	i, l, k, j
i	i, l, j
ι	i, l, r
Ι	l, i, k
h	h, n, b
σ	o, d, q
µ	u, m, p
ɢ	g, c, o
ʊ	v, u, o
þ	p, h, b
q	q, d, g

Table 16: List of the top 20 VP characters with one-to-many mappings

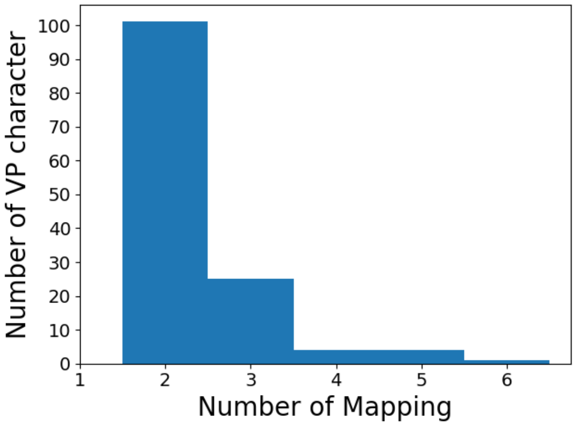


Figure 6: The number of VP characters with two or more corresponding mappings

Additionally, figure 6 presents the number of VP characters in the dataset that correspond to two or more non-VP characters. This demonstrates that a significant number of VP characters have one-to-many relationships, supporting the idea that simple mapping-based methods are not effective in `BitAbuse`.