Controllable Fake Document Infilling for Cyber Deception

Yibo Hu, Yu Lin, Erick Skorupa Parolin, Latifur Khan, Kevin Hamlen
The University of Texas at Dallas
{yibo.hu,yxl163430,erick.skorupaparolin,lkhan,hamlen}@utdallas.edu

Abstract

Recent works in cyber deception study how to deter malicious intrusion by generating multiple fake versions of a critical document to impose costs on adversaries who need to identify the correct information. However, existing approaches are context-agnostic, resulting in sub-optimal and unvaried outputs. We propose a novel context-aware model, Fake Document Infilling (FDI), by converting the problem to a controllable mask-then-infill procedure. FDI masks important concepts of varied lengths in the document, then infills a realistic but fake alternative considering both the previous and future contexts. We conduct comprehensive evaluations on technical documents and news stories. Results show that FDI outperforms the baselines in generating highly believable fakes with moderate modification to protect critical information and deceive adversaries.

1 Introduction

According to the statement of the U.S. Securities and Exchange Commission, the scope and severity of cyber risks have dramatically increased, and constant vigilance is needed to protect against intrusion (Clayton, 2017). Cyber Deception is a cybersecurity defense practice (Masud et al., 2007; Tu et al., 2008; Akbar et al., 2022) that aims at protecting critical documents once intruders penetrate the network system (Yuill et al., 2004; Bowen et al., 2009). The goal is to deceive attackers by deploying decoys such as fake documents and thus increase their cost to identify critical information.

In this work, we aim at designing a novel fake document generator that combines Cyber Deception and Natural Language Generation (NLG) technologies to generate controllable, diverse, and believable fakes at scale to protect critical information and deceive adversaries. Although recent works in Cyber Deception develop strategies to generate complicated fake technical documents such as patents, few consider adopting pretrained contextual features to enhance scalability and generation quality. For example, FORGE (Chakraborty et al., 2021) generates fake documents by replacing the concepts of a technical document with semantically similar alternatives from an expensive prerequisite ontology. WE-FORGE (Abdibayev et al., 2021) eliminates the need for ontologies by using word embedding distances. However, it identifies potential replacements only for unigrams (especially nouns) based on unbalanced word embedding clusters in a context-agnostic manner, resulting in sub-optimal or inadequate alternatives.

Meanwhile, recent studies in NLG have been driven by pre-trained contextual language models (LMs), which can generate increasingly realistic but less-controllable text (Radford et al., 2019; Raffel et al., 2020; Lewis et al., 2019; Yang et al., 2019). Sub-fields such as controllable text generation (Keskar et al., 2019; Dathathri et al., 2020), story generation (Clark et al., 2018; Fan et al., 2018), and text infilling (Fedus et al., 2018; Donahue et al., 2020) further study how to leverage LMs to generate content with desired attributes. However, few methods offer fine-grained control over concept levels or provide an efficient, controllable fake text generation strategy.

We propose a novel context-aware model, Fake Document Infilling (FDI), by converting fake document generation into a controllable mask-then-infill procedure. Specifically, we select and mask essential concepts of varied lengths in a document. Then we infill the masked spans with realistic but fake alternatives based on contextualized knowledge from an LM. To the best of our knowledge, we are the first to propose a complete controllable mask-then-infill model and design a comprehensive evaluation scheme to study fake text generation.

To briefly demonstrate the motivation for this work, Table 1 illustrates the difference between an LM (i.e., GPT-2 finetuned on the target dataset

Findings of the Association for Computational Linguistics: EMNLP 2022, pages 6534 - 6548

A. Original Article Tomographic Image Reconstruction using Training images We describe and examine an algorithm for tomographic image reconstruction where prior knowledge about the solution is available in the form of training images. We first construct a nonnegative dictionary based on prototype elements from the training images; this problem is formulated as a regularized non-negative matrix. Incorporating the dictionary as a prior in a convex reconstruction problem, we then find an approximate solution with a sparse representation in the dictionary...	
B. GPT-2: Generation given a prompt ...We describe and examine an algorithm for tomographic image reconstruction where prior knowledge about the solution is available in the form of training images. Instances were reconstructed from their images using image and pixel centroids. The concept of image reconstruction provides several advantages over previous techniques, such as indexing the solution to a representation with integral or submaximal number of cepstrates, ...	
C. WEF-Replacing nouns ...We first construct a nonnegative dictionary based on prototype elements from the training images; this problem is formulated as a regularized non-negative matrix factorization. Incorporating the dictionary as a prior in a convex reconstruction problem, we then find an approximate solution with a sparse representation in the dictionary... **D. WEF-Generation** ...We first construct a nonnegative sparsity based on prototype elements from the encoderdecoder images; this problem is formulated as a regularized non-negative matrix orthonormal. Incorporating the sparsity as a prior in a convex reconstruction problem, we then find an approximate strategy with a sparse representation in the sparsity...	
E. FDI-Replacing n-grams ...We first construct a nonnegative dictionary based on prototype elements from the training images; this problem is formulated as a regularized non-negative matrix factorization. Incorporating the dictionary as a prior in a convex reconstruction problem, we then find an approximate solution with a sparse representation in the dictionary... **F. FDI-Generation** ...We first construct a collection of missing patches based on images from the training images; this problem is formulated as a regularized non-negative matrix factorization. Incorporating the dictionary as a prior in the whole dictionary, we then find a similar estimate for missing patches in the dictionary...	

Table 1: Comparison of strategies and generated samples from different models. We preserve the document's head (the headline and the first sentence) and modify only the document's body. GPT-2 generates a new body given the document's head as a prompt. WE-FORGE (WEF) and FDI substitute certain concepts with alternatives.

(Radford et al., 2019)), WE-FORGE (WEF), and our FDI in generating fake samples of the same document. We preserve the document's head (the headline and the first sentence shown in A) and modify the document's body. GPT-2 generates a new body (shown in red in B) based on the original head in a left-to-right manner. The output is fluent but less controllable and may gradually go off-topic. Besides, GPT-2 cannot control text length and wrapping, hindering its application when following a layout is strictly required.

Both WE-FORGE and FDI adopt the strategy of replacing specific concepts (shown in blue in C and E) in the original document with alternatives (shown in red in D and F). However, WE-FORGE suffers from three major limitations. First, WE-FORGE needs to train word-embeddings from scratch for every custom dataset, requiring large training corpora (Pennington et al., 2014; Mikolov et al., 2018). Second, its word-embedding-based clustering of concepts is unbalanced and sensitive to the initialization and hyper-parameters, resulting in limited replacements for some given concepts. Finally, WE-FORGE only replaces nouns and is agnostic to context, limiting the diversity and quality of the generated text.

In contrast, FDI provides many advantages over previous methods. First, instead of training word embeddings from scratch, FDI finetunes a pre-trained LM to generate human-like text with limited data. Furthermore, FDI replaces spans of arbitrary lengths, considering the document context (through the LM) to improve the outputs' diversity and coherency. Finally, FDI implements strategies to select (mask) and find alternative concepts (infill), protecting essential details from original documents and producing realistic fake samples.

To validate the outperformance of FDI, we design an innovative set of experiments combining evaluation methods observed in distinct areas, i.e., cyber security and NLG. We collect reviews from more than 40 volunteers over $1.4k$ fake documents on technical and non-technical datasets. Finally, we compile the reviews to evaluate the model's ability to generate natural text and its effectiveness in protecting the original information and deterring attackers. Our code is publicly available.[1]

2 Related Work

Cyber Deception. Cyber Deception aims at deceiving attackers by misguiding them toward inaccurate information with deployed decoys in the network systems of enterprises. Early works generate decoy honey files (Yuill et al., 2004; White and Thompson, 2006; Bowen et al., 2009; Whitham, 2013) or simple documents with basic NLP methods (Voris et al., 2012; Wang et al., 2013) to entice attackers and improve intrusion and exfiltration detection. Recent works combine advanced NLP techniques to generate fake technical documents at scale while enhancing believability. These efforts

[1] https://github.com/snowood1/FDI

include substituting words or concepts based on part-of-speech tagging (Whitham, 2017), prerequisite ontologies (Chakraborty et al., 2021), concept occurrences graphs (Karuna et al., 2021), or word embeddings (Abdibayev et al., 2021). Nevertheless, these methods are context-agnostic, limiting producing diverse and natural outputs. The only exception (Ranade et al., 2021) uses vanilla contextualized LMs on short description texts instead of long technical documents.

Controllable Text Generation. Building costly conditional LMs for desired attributes, by either training from scratch (Zellers et al., 2019; Keskar et al., 2019) or back-propagating gradients (Dathathri et al., 2020), are extensively studied. The attributes are usually pre-defined by a list of control codes or keywords. Other lightweight alternatives are proposed by using discriminators or Bayes' rules to control the attributes of generated text during the decoding time (Krause et al., 2020; Yang and Klein, 2021; Liu et al., 2021). A subfield called **Story Generation** focuses on generating short stories given hints such as title, storyline, premise, entities, or rare words (Clark et al., 2018; Fan et al., 2018, 2019; Yao et al., 2019; Goldfarb-Tarrant et al., 2020; Rashkin et al., 2020; Tan et al., 2021; Ippolito et al., 2019, 2020b; Das and Verma, 2020). Specifically, Zellers et al. (2019) generate fake news stories conditioned on metadata from a list of propaganda websites. Nevertheless, these fields differ from our task. They mainly focus on non-technical domains (e.g., news and stories) and lack fine-grained control over concept levels.

Text Infilling. Text infilling is a generalization of the cloze task (Taylor, 1953) from single words to spans of varied lengths. Current works focus on correctly infilling the incomplete text for applications in text editing or ancient documents restoration (Fedus et al., 2018; Zhu et al., 2019; Liu et al., 2019; Zaidi et al., 2019; Donahue et al., 2020; Shen et al., 2020). However, the *controllable mask-then-infill* task addressed in this paper is more complex. It involves masking relevant concepts (text spans) in a document and infilling realistic yet misleading spans to replace such masks.

Adversarial augmentation. This task aims at generating perturbed augmented samples to improve the robustness of NLP models by heuristic rules that replace words from WordNet or word embeddings (Alzantot et al., 2018; Jia et al., 2019; Ren et al., 2019; Wei and Zou, 2019), contextualized perturbations (Garg and Ramakrishnan, 2020; Li et al., 2020, 2021), or comprehensive frameworks (Ribeiro et al., 2020; Morris et al., 2020; Wu et al., 2021). Again, these random perturbation methods lack precise control over concepts, hindering their usage for our task.

3 Approach

3.1 Framework

We follow the same convention of fake document generation proposed in FORGE (Chakraborty et al., 2021): Given a real document d as input, the model generates a set D' of fake documents. Each fake document $d' \in D'$ is similar to d to be believable, yet sufficiently different from d to be inaccurate. We obtain d' by replacing certain concepts c of d with alternatives c'. High-quality d' is expected to cost the attacker much time to identify the real d from the $|D'| + 1$ documents. Thus, a fake document generator needs to ensure believability by considering at least two aspects: (1) how to select the set of concepts C to be replaced; and (2) how to choose replacement concept c' for every $c \in C$.

We convert the formulation above into the controllable document infilling task. Given a document d, we first extract and mask text snippets of varied lengths expressing each important concept $c \in C$. Then, we use an LM to infill the masked spans with realistic but inauthentic alternatives c', considering the context of these spans. FDI addresses these sub-tasks by designing (1) a controllable masking function to select concepts and (2) a decoding strategy to replace the masked spans.

Figure 1 shows the (a) training and (b) inference procedures of FDI. First, we apply the random masking approach to train a robust and flexible LM to fill various types of masks. Then, we use a curated strategy to precisely steer text generation during the inference step. Specifically for inference, we first use controllable masking to produce masked examples, protecting essential information of d. Then, we use the trained LM to replace each masked concept $c \in C$ with a sampled c'. To ensure the fakeness of the generated document, we introduce a penalization factor in the decoding step to avoid the model predicting the original concept (i.e., let $c' \neq c$). Finally, we obtain a completed fake document by infilling the input text with the predicted alternatives. We detail each component of FDI in the following subsections.

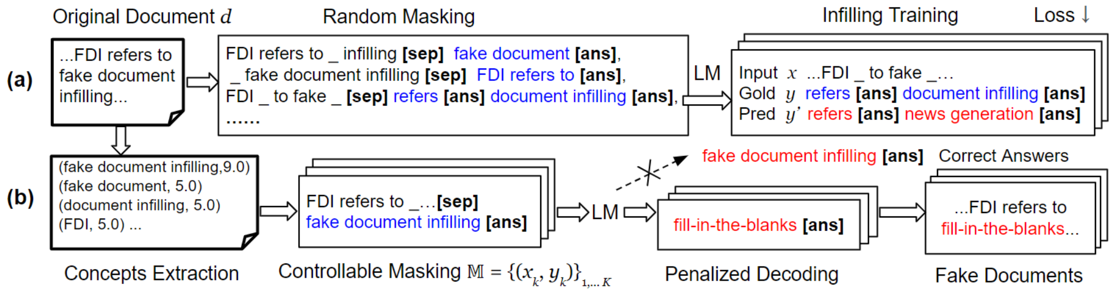

Figure 1: The (a) training and (b) inference steps of FDI. Random masking is used to train an LM for general text infilling. During inference, we mask important text spans and utilize the trained LM to replace such masked spans. Penalization mechanism is applied to encourage generating fake answers.

3.2 Training

The training step involves finetuning an LM to text infilling task, utilizing the random masking approach. Figure 1(a) illustrates three different training pairs. Each concatenates input x and target y by a separator token $[sep]$. x is generated by a random masking function $f(d)$, which replaces specific spans $C = \{c_1, ..., c_n\}$ in document d with special (blank) tokens. $y = c_1[ans]...c_n[ans]$ refers to the answers to the blanks concatenated with special tokens $[ans]$. We finetune LM(θ) to learn the distribution $p_\theta(y|x)$ by minimizing the cross-entropy loss between the target y and the probability distributions of prediction y'.

We design $f(d)$ to generate various masked examples with coarse control over the granularities. Similar to (Donahue et al., 2020), we use more special tokens instead of a universal blank to specify three granularities: words, n-grams, and sentences. For example, x in Figure 1(a) becomes "FDI $[masked\_word]$ to fake $[masked\_ngram]$." (we only show a universal blank "_" in the figure for simplicity). Next, we traverse the hierarchy of d to sample each mask type randomly and obtain a masked token rate of 15% suggested in (Devlin et al., 2018; Donahue et al., 2020) (details in Appendix B). Finally, we generate various masked examples of each d for training data augmentation.

3.3 Inference

The inference process (shown in Figure 1(b)) produces fake documents through the following steps: (1) extracting and selecting the appropriate set of concepts C; and (2) determining the fake replacement c' for every $c \in C$, through decoding method.

3.3.1 Concepts Extraction and Selection

Concepts extraction and selection are essential components in fake document generators and vary in schemas. Therefore, we define the following settings: First, instead of expensive annotation (Chakraborty et al., 2021), we followed recent works to use automatical keywords (e.g., based on TF-IDF (Abdibayev et al., 2021)) as critical information for scalable evaluation. We chose RAKE (Rose et al., 2010) to score n-grams to extract concepts with varying lengths without additional cost. Second, we only revised the document body with the head unchanged, as illustrated in Table 1. Completely altering the head may let intruders skip the forged document quickly. We expect the fake samples to alter critical details without changing the topics. Besides, this setting enables us to compare naive GPT-2 that needs to initiate with a given prompt and is commonly used in NLG evaluation (Clark et al., 2021).

Algorithm 1 illustrates the *Controllable Masking* procedure for selecting and extracting concepts, which includes two main parts: (1) Lines 1-6 build the candidate pool of concepts from document d, consisting of sets C (words or n-grams) and S (whole sentences); and (2) Lines 7-17 generate K masked examples $\mathbb{M} = \{M_1, ..., M_K\}$ by sampling from the candidate pool. Each masked example M_k results in various fake documents during the later decoding step. In this way, FDI can produce diverse examples which vary in masked locations and replacements of each mask.

In the first part of Algorithm 1, Line 1 splits document d by stop-words and delimiters to create the initial set of concepts C. Line 2 computes the importance score of each concept $c \in C$ through the degree $deg(w)$ and frequency $freq(w)$ in its word co-occurrence graph of a term w occurring in c, following (Rose et al., 2010). Next, we filter the concepts based on the quantile $Q_R(\cdot)$ of the importance scores in Line 3. Long concepts often get higher RAKE scores than short concepts. For

Algorithm 1: Controllable Masking

Input : document d, stop-words W_{st}, thresholds q_{min}, t_s, γ, masking probabilities p_s, p_c.
Output : list of masked examples $\mathbb{M}$ of size K.

1 $C \leftarrow splitConcepts\,(d,\, W_{st})$
2 $R_C \leftarrow \{r(c) : \sum_{w \in c} deg(w)/freq(w)$ for c in C $\}$
3 $C \leftarrow \{c$ if $r(c) \geq Q_R\,(C,\, q_{min})$ for c in C $\}$
4 $C \leftarrow \{c$ if $c \notin d.head\}$
5 $C \leftarrow concatDet\,(C, d)$
6 $S \leftarrow getSents\,(d,\, C,\, t_s)$
7 $\mathbb{M} \leftarrow [\emptyset]$
8 **for** k *in range* (K) **do**
9 $\quad y_k \leftarrow \{\emptyset\}$
10 $\quad$ **do**
11 $\quad\quad y_k \leftarrow y_k \cup \{randomSample\,(S,\, p_s)\,\}$
12 $\quad\quad y_k \leftarrow y_k \cup \{randomSample\,(C,\, p_c)\,\}$
13 $\quad$ **while** $maskedRate\,(d,\, y_k) < \gamma$
14 $\quad y_k \leftarrow mergeCloseMasks\,(y_k)$
15 $\quad x_k \leftarrow getMaskedInput\,(d,\, y_k)$
16 $\quad \mathbb{M} \leftarrow \mathbb{M} \cup (x_k,\, y_k)$
17 **return** $\mathbb{M}$

example, in Figure 1 (b), "fake document infilling" gets a higher score than its member phrases. Therefore, we empirically set the lower bound q_{min} to 40%, a trade-off between concepts' importance and diversity (in terms of length).

A document's head (e.g., the title and the first sentence) often contains topic words and summarizes the content. Thus, we ignore extracting these phrases to prevent generating entirely off-topic articles in Line 4, as discussed in the start of subsubsection 3.3.1. For instance, we remove the selected topic concept "tomographic image reconstruction" from the candidate set in Table 1 A.

RAKE removes masked phrases' determiners and may result in obvious plural noun errors during infilling. For example, LMs infill "find an approximate solution..." to "find an similar estimate..." in Table 1 (E). The easiest solution to alleviate such errors is to replace the extracted span with its determiner as a whole, e.g., "find an approximate solution...". Thus, we concatenate extracted spans with their determiners through function $concatDet(\cdot)$ in Line 5.

Besides candidate concepts C, we can optionally replace sentences with a high density of key concepts. In practice, replacing a whole sentence generally produces better results than densely infilling many blanks in the same sentence. Therefore, in Line 6, we collect the sentences from d whose percentage of tokens belonging to any concept in C is higher than the threshold t_s. Function $getSents(\cdot)$ returns such dense sentences to form the set S.

Once we obtain the candidate pool of concepts from d, we sample K masked examples $\mathbb{M} = \{M_1, ..., M_K\}$. Each M_k includes input x_k with special masked tokens, and the answer spans y_k (as shown in subsection 3.2). In Lines 8-13, for each masked example, we collect y_k by sampling the sets S and C with probabilities p_s and p_c, respectively. We iteratively sample until we get enough non-overlapping concepts that reach a threshold of masked token rate γ.

We also merge short masked spans located closely within the same sentence into longer spans to reduce the number of masked spans in M in Line 14. For example, we merge the two masked spans in "find an approximate solution with a sparse representation ..." to one span in Table 1 (E). We get the corresponding masked input x_k by replacing spans in y_k with special masked tokens in Line 15. We repeat the above sampling process to get our collections of (x_k, y_k) pairs.

3.3.2 Decoding

We design a penalized decoding strategy based on Top-p% sampling (Holtzman et al., 2019) to generate natural yet fake texts. The training step minimizes the cross-entropy loss between the answers and prediction probabilities to retrieve the original document. However, the inference uses sampling instead of greedy search to get various outputs that are unlikely to be identical to the original document. Furthermore, we discount the scores of the tokens for the correct answers to encourage fake outputs during the inference, similar to the mechanism for discouraging repetition (Keskar et al., 2019).

Specifically, we first get a subset of tokens A from the correct answers y of each M by filtering out too-short tokens, probably stopwords or insignificant sub-words such as prefixes. Then, given the input $\mathbf{x}$, the probability distribution over the next possible token being word i in the vocabulary V is the softmax:

$$p(y = i|\mathbf{x}) = \frac{\exp\left(z_i/(T \cdot I(i \in A)\right)}{\sum_j \exp\left(z_j/(T \cdot I(j \in A)\right)}, \quad (1)$$

where T is the temperature parameter and z_i is each i's score. $I(\cdot) = \delta$ if true else 1, and δ is the penalty parameter. A high δ discourages generating correct answers but also produces errors. Thus, we set $\delta = 1.2$ in our experiments based on our empirical observation. Finally, following (Holtzman et al., 2019), we sample from the most probable tokens whose cumulative probability comprises the top-95% of the entire vocabulary.

One concern of the inference step is to control the fakeness of the output. Substituting concepts with similar semantic replacements fails to protect critical information. Due to its unbalanced and unvaried candidate pool, WE-FORGE often suffers from replacing a noun concept with its synonyms, such as substituting "solution" with "strategy" in Table 1 C and D.

In contrast, FDI controls fakeness efficiently by masking various spans from words to sentences, significantly improving the diversity and thus reducing the chance of getting similar outputs. Moreover, FDI infills fake samples conditioned on the incomplete context hiding critical information. Even for the exact phrases that occur in different places, we do not replace them with an identical replacement. Instead, the LM decodes their plausible replacements based on different contexts. This infilling and sampling process favors common, safe, but lossy answers. For example, the document with masked concepts in Table 1 E can result in various outputs with the same structure but distinct details. Later, the sampled answers like "images" and "the whole dictionary" in Table 1 F seem natural but uninformative - they hide the critical information expressed in the original document. Finally, the penalty mechanism in Equation 1 also encourages the model to infill a fake answer.

4 Experiments

4.1 Datasets

Following previous cyber deception works, we conducted experiments on two technical datasets: the **CS** (Donahue et al., 2020) and the patent abstracts dataset (**PAT**)[2]. The first consists of abstracts from computer science papers on arXiv. The latter covers topics such as Electrical, Chemistry, and Biology. Additionally, we experimented on a non-technical dataset by crawling and filtering a subset of news from the Wall Street Journal (**WSJ**). Table 2 summarizes three datasets' statistics, document lengths, and training sequence lengths we chose.

4.2 Comparison Scheme

We considered various possible competitors discussed in section 2 as baselines. We first selected word-embedding-based **WE-FORGE**, the state-of-the-art fake document generator for Cyber Deception. Thus, we ignored other cyber deception and adversarial augmentation models using word embeddings. Instead, we chose **EDA** (Wei and Zou, 2019) as a typical context-agnostic adversarial augmentation baseline. We also compared **GPT-2** small model (which serves as FDI's base model) to validate the advantage of the proposed mask-then-infill strategy. We finetuned it and FDI on each training set (details in Appendix A). Finally, we ignored other controllable text generators or contextual perturbation models (Li et al., 2021). These methods are neither computationally efficient or show a clear advantage over the selected models on fine-grained control over concepts for this task.

Dataset	Train / dev / test	# tokens	seq-len
CS	409,555 / 8,547 / 8,498	205 ± 70	400
PAT	16,000 / 4,000 / 5,743	132 ± 58	256
WSJ	40,862 /2,270 / 2,270	292 ± 78	512

Table 2: The datasets used in our experiments.

4.3 Evaluation Design

We sampled documents from each test sets with similar lengths (e.g., 180 to 200 tokens for CS dataset) and generated their fake versions using 4 models. We combined NLG and Cyber Deception evaluation methods to design our experiments. We collected reviews from more than 40 computer science students. Our experiments consist of *Quiz-1 Detection* and *Quiz-2 Evaluation*.

Quiz-1 We followed a similar human evaluation schema utilized in cyber deception (Chakraborty et al., 2021; Abdibayev et al., 2021) and machine-generated text detection (Liu et al., 2016; Van Der Lee et al., 2019; Ippolito et al., 2020a; Zellers et al., 2021; Clark et al., 2021) to evaluate whether the fake samples can deter hackers. Reviewers were asked to identify the original document among three fake copies generated by a single (unknown) model in each *example set* (1 true + 3 fake). Each reviewer analyzed $4h$ example sets (i.e., $4h\times(1+3)$ documents) to evaluate all four models h times. Finally, we computed each model's average detection accuracy and evaluation time.

However, Quiz-1 ignores the effects of distinct generation patterns and amounts of fake content. For example, a generated sample with minor modifications (e.g., adding or deleting a few stopwords or replacing synonyms) is less distinguishable. Yet, it does not protect any original document's information for the cyber deception purpose.

[2] https://github.com/chirag-choudhary/Patent-Summarizer

	# of example sets				Config	# of fake samples
	CS	PAT	WSJ	all		
Quiz-1	160	88	136	384	1 true + 3 fake	1152
Quiz-2	32	18	32	82	1 true + 4 fake	328

Table 3: Statistics of experimented evaluation sets

Data	Metric	EDA	WEF	GPT	FDI
CS	Acc ↓	0.93	0.93	0.65	**0.60**
	Time ↑	1.53±1.0	2.53±1.5	3.31±0.3	**3.53**±1.3
PAT	Acc ↓	1.00	0.86	0.77	**0.64**
	Time ↑	2.91±2.2	4.03±2.5	**4.41**±2.5	4.17±2.0
WSJ	Acc ↓	0.82	0.82	**0.62**	0.74
	Time ↑	2.21±1.7	2.45±1.7	**3.98**±1.3	3.86±1.3
avg.	Acc ↓	0.91	0.88	0.67	**0.66**
	Time ↑	2.22±1.6	3.00±1.9	**3.90**±1.5	3.85±1.4

Table 4: Mean accuracy and time taken (in minutes) by participants to review one example in Quiz-1.

Data	Metric	EDA	WEF	GPT	FDI
CS	Flu ↑	1.97	2.97	3.06	**3.19**
	Coh ↑	2.28	2.94	2.84	**3.25**
	Exp ↑	1.78	2.84	2.81	**3.00**
	Pref ↑	1.66	2.66	2.56	**3.19**
PAT	Flu ↑	1.39	3.22	**3.33**	3.28
	Coh ↑	1.56	2.72	2.67	**3.17**
	Exp ↑	1.33	2.72	2.67	**3.06**
	Pref ↑	1.11	2.72	2.72	**3.44**
WSJ	Flu ↑	1.75	2.81	**3.28**	3.06
	Coh ↑	1.78	2.28	2.63	**2.78**
	Exp ↑	1.69	1.84	**2.72**	2.59
	Pref ↑	1.72	2.28	2.88	**3.13**
avg.	Flu ↑	1.76	2.96	**3.21**	3.16
	Coh ↑	1.93	2.63	2.72	**3.05**
	Exp ↑	1.65	2.43	2.74	**2.85**
	Pref ↑	1.56	2.52	2.72	**3.22**

Table 5: Mean scores of **flu**ency, **coh**erency, **exp**ertise, and **pref**erence in Quiz-2.

Quiz-2 To overcome Quiz-1's limitations, we designed Quiz-2 to evaluate fake samples' quality and effectiveness. Each question set includes one known original document and four fake copies generated by four models in an unknown order. Reviewers were asked to evaluate five metrics for the fake samples based on a 4-point Likert scale: (1) **fluency** of the article; (2) **coherency** of the article; (3) expert knowledge (**expertise**) required to identify the article is fake; (4) **fakeness** of the article; and (5) the overall **preference** in the articles.

The above scores combine standard NLG metrics (fluency and coherency) and metrics we design for cyber deception. Specifically, fakeness indicates the amount and the effectiveness of modification applied to the original document to deceive the adversary and protect certain essential facts. We define four fakeness categories: **1-inadequate**, **2-marginal**, **3-moderate**, and **4-excessive**. We do not use overlap-based metrics such as BLEU (Papineni et al., 2002) as they are inappropriate for evaluating many realistic infills without word-level overlap (Donahue et al., 2020). See Appendix C for more details of our questionnaire.

4.4 Results

Table 3 shows statistics of experimented evaluation sets. Specifically, we evaluated 384 example sets in Quiz-1 (96 sets per model and 1,152 fake examples overall. For Quiz-2, we tested 82 example sets, including 328 fake samples. These samples come from the same 30 articles and their 360 fake copies. In addition, each evaluated set was evaluated by at least two students.

Table 4 shows the mean detection accuracy and the average time taken for participants to review one example in each scenario in Quiz-1. Compared with other context-agnostic baselines, GPT-2 and FDI get lower accuracy and longer time. The results indicate that examining texts generated by current LMs requires more effort than a superficial judgment based on fluency-related quality aspects (Clark et al., 2021). Although time metrics are relatively similar for these models, FDI's superiority varies across domains. It presents lower accuracy (i.e., better at misleading humans) in CS and PAT but higher in WSJ.

Table 5 compiles the reviews of Quiz-2 for a more comprehensive analysis of the generated fake documents. We illustrate the fakeness metric separately in Figure 2 due to its particularity (higher fakeness doesn't mean superiority). Table 5 shows that EDA achieves the worst fluency and coherency due to its random perturbation strategy. GPT-2 generates the most fluent output with contextual knowledge in the unrestricted left-to-right manner. However, its output lacks fine-grained control and gradually goes off-topic, thus affecting its coherency. WE-FORGE and FDI preserve the article's logical and consistent relation by replacing specific snippets. Yet, WE-FORGE results in unstable performance due to its unigram replacement based on unbalanced word embeddings clusters. In contrast, FDI combines improved replacement strategies and contextual features, consistently reporting superior coherency and fluency.

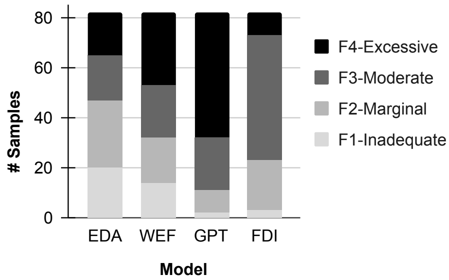

Figure 2: Distribution of the samples' fakeness. Most of FDI's samples (61.0%) have moderate fakeness.

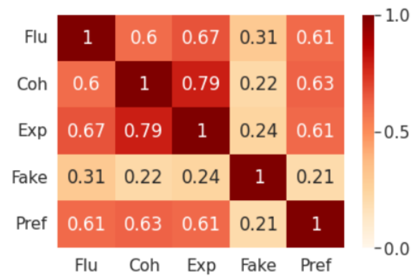

Figure 3: The Spearman correlation heatmap for fluency, coherency, expertise, fakeness, and preference scores.

Fakeness	Pref	# count
1-Inadequate	1.69	39
2-Marginal	2.20	74
3-Moderate	**2.98**	110
4-Excessive	2.52	105

Table 6: Mean preference and the number of example sets for each fakeness type.

The expertise score refers to the level of expert knowledge required for the reviewer to identify whether the article is fake or not. Given the low fluency and coherency, EDA's fake samples require lower expertise to be recognized. WE-FORGE prunes out all words other than nouns because such terms are unlikely to contribute to the content of a technical document (Abdibayev et al., 2021). Yet, this method hinders its outputs' diversity in a news story with fewer important nouns such as technical terms but more essential verbs. As a result, it may generate easily identifiable fake samples such as replacing "President Joe Biden" with "President Joe Trump". Therefore, WE-FORGE is competitive with GPT-2 in CS and PAT but performs poorly in WSJ. In contrast, GPT-2 avoids the above issues caused by replacing unigram, which also explains its superiority in accuracy and expertise score in WSJ. FDI addresses WE-FORGE's issue by replacing n-grams respecting both the preceding and the following context. Thus, its errors related to reviewers' knowledge are more subtle. Although we focus on technical datasets, these results suggest that FDI generalizes well in other domains.

The ideal fake samples should have moderate fakeness, neither too close nor too far away from the original text. Figure 2 illustrates that 61.0% of FDI's generated samples have moderate fakeness, achieving the best trade-offs. In contrast, EDA is ineffective in protecting critical information because 57.3% of its samples have marginal or inadequate fakeness. WE-FORGE applies more effective modification than EDA. Yet, the near-uniform distribution of WE-FORGE's fakeness is consistent with its unstable performance. GPT-2's samples tend to introduce excessive fakeness, substantially diverging from the original documents.

In the final question of Quiz-2, we asked the reviewers to rank their favorite fake articles from score 4 to score 1. Then we calculate each model's average results as the preference scores. Table 5 shows that FDI is the overall best model. Based on the participants' feedback, various factors influence their decision-making. For example, some reviewers like the most fluent samples, while others prefer those with realistic modification. Therefore, we analyze the relationships between these metrics in Figure 3, which illustrates that all the scores other than fakeness show strong positive correlations. The results are as expected as we prefer fluent, coherent fake documents that require expert knowledge to identify. In contrast, we observe weak positive correlations between fakeness and the other metrics.

To understand the human preference in fakeness, Table 6 summarizes the mean preference scores of the samples of different fakeness types. And it shows that the reviewers favor the samples with moderate fakeness. The above observation again validates a trade-off between the amount of fake content and the superiority of the fake samples. It also indicates that fakeness is a relatively independent metric from the other evaluation metrics. Thus, it is necessary to include fakeness in the future cyber deception study.

4.5 Parameter Study

Due to the extensive time and efforts associated with human-driven experiments, we used the same hyperparameters for all datasets based on evaluation results on small validation sets (details in Appendix B). A key hyperparameter is max masked rate γ, as shown in Algorithm 1. Samples with low γ (e.g.,10%) are likely to be labeled as inadequate fakeness. In contrast, high γ results in excessive fakeness and errors because the model needs to fill in more blanks given less context. As moderate fakeness is desired in cyber deception work, we set $\gamma = 20\%$. Yet, users can specify their preferred γ in custom datasets. Besides γ, many parameters provide randomness in the samples but do not significantly affect the human evaluation result.

5 Conclusion and Future Work

We propose a novel fake document generator, FDI, for network intrusion defense and intellectual property protection. FDI relies on a complete mask-then-infill process with a curated strategy for fake documents generation. Our experiments explore "how easily the original documents are identified" and "how critical information is protected" with more fake samples and generation patterns. FDI shows consistent superiority in generating realistic fake samples while protecting the information and deceiving the hackers.

While human evaluation remains the gold standard for evaluating various NLG applications, future work can explore automatic detection methods (Zellers et al., 2019; Gehrmann et al., 2019; Bakhtin et al., 2019; Schuster et al., 2020) to alleviate human efforts. Besides, this work focuses on technical documents and shows generalization in news stories. Future work can also extend its applications to other critical domains, such as political science (Parolin et al., 2022, 2021; Hu et al., 2022; Skorupa Parolin et al., 2022; Hu and Khan, 2021).

6 Limitations

Due to the expensive human evaluation, we empirically selected some configurations on small validation sets. Besides, we reduced the overlaps between the reviewers to cover more samples and reduce the randomness. Although at least two reviewers evaluated each article set, the overlap was small to calculate Kappa. We were aware that evaluators might calibrate the metrics differently without training, a commonly reported issue in NLG tasks (Ippolito et al., 2020a; Clark et al., 2021). However, pre-evaluation training on fakeness introduced bias because the reviewers may judge only based on the distinct patterns of different models (as shown in Table 1). Thus, we didn't intervene in the evaluation. Instead, we extensively analyzed reviewers' choices in Figures 2, 3, and Table 6. More work needs to be done by (1) designing simple but unbiased instructions to help reviewers score more consistently. (2) More overlapping experiments between reviewers to calculate Kappa.

Second, FDI is not flawless and suffers from similar weaknesses as all LMs. Text infilling models may generate repetitive text, incomplete words, or unmatched parenthesis, resulting in a high infilling failure rate (Shen et al., 2020). Therefore, we designed several heuristic steps in Lines 5, 6, and 14 of Algorithm 1 to simplify the infilling tasks and reduce errors. We believe a more powerful LM, such as GPT-3 (Brown et al., 2020), can improve the performance further. Besides, GPT-2 is originally pretrained for left-to-right text generation. Some alternative LMs, such as T-5 (Raffel et al., 2020) and BART (Lewis et al., 2019), have already learned elementary text-infilling tasks during the pretraining. Future work should also explore how these models perform in our framework.

Finally, we designed a simple penalized decoding strategy based on Top-$p\%$ schema to encourage diverse fake generations. Yet, it also generated errors like other constrained decoding methods. Future work should optimize the decoding algorithm and post-processing methods.

7 Ethical Considerations

We acknowledge that similar mechanisms may be abused to generate disinformation, such as fake news (Zellers et al., 2019). Besides, language models have been shown to encode biases from the training data (Barberá et al., 2021). Thus, we remove controversial and sensitive news samples to mitigate these issues during our evaluation. With the rapid evolution of Cyber Deception and NLG technologies, we believe this work creates more value than risks on balance.

Acknowledgments

The research reported herein was supported in part by NSF awards DMS-1737978, DGE-2039542, OAC-1828467, OAC-1931541, and DGE-1906630, ONR awards N00014-17-1-2995 and N00014-20-1-2738, Army Research Office Contract No. W911NF2110032 and IBM faculty award (Research). We sincerely thank all the participants in the questionnaire for their valuable contributions.

References

Almas Abdibayev, Dongkai Chen, Haipeng Chen, Deepti Poluru, and V. S. Subrahmanian. 2021. Using word embeddings to deter intellectual property theft through automated generation of fake documents. *ACM Trans. Manage. Inf. Syst.*, 12(2).

Khandakar Ashrafi Akbar, Sadaf Md Halim, Yibo Hu, Anoop Singhal, Latifur Khan, and Bhavani Thuraisingham. 2022. Knowledge mining in cybersecurity: From attack to defense. In *IFIP Annual Conference on Data and Applications Security and Privacy*, pages 110–122. Springer.

Moustafa Alzantot, Yash Sharma, Ahmed Elgohary, Bo-Jhang Ho, Mani Srivastava, and Kai-Wei Chang. 2018. Generating natural language adversarial examples. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2890–2896.

Anton Bakhtin, Sam Gross, Myle Ott, Yuntian Deng, Marc'Aurelio Ranzato, and Arthur Szlam. 2019. Real or fake? learning to discriminate machine from human generated text.

Pablo Barberá, Amber E Boydstun, Suzanna Linn, Ryan McMahon, and Jonathan Nagler. 2021. Automated text classification of news articles: A practical guide. *Political Analysis*, 29(1):19–42.

Lukas Biewald. 2020. Experiment tracking with weights and biases. Software available from wandb.com.

Brian M Bowen, Shlomo Hershkop, Angelos D Keromytis, and Salvatore J Stolfo. 2009. Baiting inside attackers using decoy documents. In *International Conference on Security and Privacy in Communication Systems*, pages 51–70. Springer.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Tanmoy Chakraborty, Sushil Jajodia, Jonathan Katz, Antonio Picariello, Giancarlo Sperli, and V. S. Subrahmanian. 2021. A fake online repository generation engine for cyber deception. *IEEE Transactions on Dependable and Secure Computing*, 18(2):518–533.

Elizabeth Clark, Tal August, Sofia Serrano, Nikita Haduong, Suchin Gururangan, and Noah A Smith. 2021. All that's 'human'is not gold: Evaluating human evaluation of generated text. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 7282–7296.

Elizabeth Clark, Yangfeng Ji, and Noah A Smith. 2018. Neural text generation in stories using entity representations as context. In *Association for Computational Linguistics: Human Language Technologies*.

Jay Clayton. 2017. Statement on cybersecurity. Accessed: 2021-11-10.

Avisha Das and Rakesh M Verma. 2020. Can machines tell stories? a comparative study of deep neural language models and metrics. *IEEE Access*, 8:181258–181292.

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

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Chris Donahue, Mina Lee, and Percy Liang. 2020. Enabling language models to fill in the blanks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*.

A. Fan, M. Lewis, and Y. Dauphin. 2018. Hierarchical neural story generation. *arXiv preprint arXiv:1805.04833*.

Angela Fan, Mike Lewis, and Yann Dauphin. 2019. Strategies for structuring story generation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2650–2660.

William Fedus, Ian Goodfellow, and Andrew M Dai. 2018. Maskgan: Better text generation via filling in the _. In *International Conference on Learning Representations*.

Siddhant Garg and Goutham Ramakrishnan. 2020. Bae: Bert-based adversarial examples for text classification. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6174–6181.

Sebastian Gehrmann, Hendrik Strobelt, and Alexander Rush. 2019. GLTR: Statistical detection and visualization of generated text. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 111–116, Florence, Italy. Association for Computational Linguistics.

Seraphina Goldfarb-Tarrant, Tuhin Chakrabarty, Ralph Weischedel, and Nanyun Peng. 2020. Content planning for neural story generation with aristotelian rescoring. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4319–4338.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2019. The curious case of neural text degeneration. In *International Conference on Learning Representations*.

Yibo Hu, MohammadSaleh Hosseini, Erick Skorupa Parolin, Javier Osorio, Latifur Khan, Patrick Brandt, and Vito D'Orazio. 2022. Conflibert: A pre-trained language model for political conflict and violence. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5469–5482.

Yibo Hu and Latifur Khan. 2021. Uncertainty-aware reliable text classification. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 628–636.

D. Ippolito, D. Grangier, C. Callison-Burch, and D. Eck. 2019. Unsupervised hierarchical story infilling. In *NAACL Workshop on Narrative Understanding*, pages 37–43.

Daphne Ippolito, Daniel Duckworth, Chris Callison-Burch, and Douglas Eck. 2020a. 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.

Daphne Ippolito, David Grangier, D. Eck, and Chris Callison-Burch. 2020b. Toward better storylines with sentence-level language models. In *ACL*.

Robin Jia, Aditi Raghunathan, Kerem Göksel, and Percy Liang. 2019. Certified robustness to adversarial word substitutions. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4129–4142.

Prakruthi Karuna, Hemant Purohit, Sushil Jajodia, Rajesh Ganesan, and Ozlem Uzuner. 2021. Fake document generation for cyber deception by manipulating text comprehensibility. *IEEE Systems Journal*, 15(1):835–845.

Nitish Shirish Keskar, Bryan McCann, Lav Varshney, Caiming Xiong, and Richard Socher. 2019. CTRL - A Conditional Transformer Language Model for Controllable Generation. *arXiv preprint arXiv:1909.05858*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Ben Krause, Akhilesh Deepak Gotmare, Bryan McCann, Nitish Shirish Keskar, Shafiq Joty, Richard Socher, and Nazneen Fatema Rajani. 2020. GeDi: Generative Discriminator Guided Sequence Generation. *arXiv preprint arXiv:2009.06367*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*.

Dianqi Li, Yizhe Zhang, Hao Peng, Liqun Chen, Chris Brockett, Ming-Ting Sun, and William B Dolan. 2021. Contextualized perturbation for textual adversarial attack. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5053–5069.

Linyang Li, Ruotian Ma, Qipeng Guo, Xiangyang Xue, and Xipeng Qiu. 2020. Bert-attack: Adversarial attack against bert using bert. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6193–6202.

Alisa Liu, Maarten Sap, Ximing Lu, Swabha Swayamdipta, Chandra Bhagavatula, Noah A Smith, and Yejin Choi. 2021. Dexperts: Decoding-time controlled text generation with experts and anti-experts. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 6691–6706.

Chia-Wei Liu, Ryan Lowe, Iulian Vlad Serban, Mike Noseworthy, Laurent Charlin, and Joelle Pineau. 2016. How not to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2122–2132.

Dayiheng Liu, Jie Fu, Pengfei Liu, and Jiancheng Lv. 2019. TIGS: An inference algorithm for text infilling with gradient search. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4146–4156, Florence, Italy. Association for Computational Linguistics.

Mohammad M Masud, Latifur Khan, and Bhavani Thuraisingham. 2007. A hybrid model to detect malicious executables. In *2007 IEEE International Conference on Communications*, pages 1443–1448. IEEE.

Tomas Mikolov, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *ICLR*.

Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. 2018. Advances in pre-training distributed word representations. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*.

John X. Morris, Eli Lifland, Jin Yong Yoo, Jake Grigsby, Di Jin, and Yanjun Qi. 2020. Textattack: A framework for adversarial attacks, data augmentation, and adversarial training in nlp. In *EMNLP*.

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

Erick Skorupa Parolin, Yibo Hu, Latifur Khan, Patrick T Brandt, Javier Osorio, and Vito D'Orazio. 2022. Confli-t5: An autoprompt pipeline for conflict related text augmentation. In *2022 IEEE International Conference on Big Data (Big Data)*. IEEE.

Erick Skorupa Parolin, Yibo Hu, Latifur Khan, Javier Osorio, Patrick T Brandt, and Vito D'Orazio. 2021. Come-ke: A new transformers based approach for knowledge extraction in conflict and mediation domain. In *2021 IEEE International Conference on Big Data (Big Data)*, pages 1449–1459. IEEE.

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

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

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, Peter J Liu, et al. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(140):1–67.

Priyanka Ranade, Aritran Piplai, Sudip Mittal, Anupam Joshi, and Tim Finin. 2021. Generating fake cyber threat intelligence using transformer-based models. *arXiv preprint arXiv:2102.04351*.

Hannah Rashkin, Asli Celikyilmaz, Yejin Choi, and Jianfeng Gao. 2020. Plotmachines: Outline-conditioned generation with dynamic plot state tracking. *arXiv preprint arXiv:2004.14967*.

Shuhuai Ren, Yihe Deng, Kun He, and Wanxiang Che. 2019. Generating natural language adversarial examples through probability weighted word saliency. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1085–1097.

Marco Túlio Ribeiro, Tongshuang (Sherry) Wu, Carlos Guestrin, and Sameer Singh. 2020. Beyond accuracy: Behavioral testing of nlp models with checklist. In *ACL*.

Stuart Rose, Dave Engel, Nick Cramer, and Wendy Cowley. 2010. Automatic keyword extraction from individual documents. *Text mining: applications and theory*, 1:1–20.

Tal Schuster, Roei Schuster, Darsh J Shah, and Regina Barzilay. 2020. The limitations of stylometry for detecting machine-generated fake news. *Computational Linguistics*, 46(2):499–510.

Tianxiao Shen, Victor Quach, Regina Barzilay, and Tommi Jaakkola. 2020. Blank Language Models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, Online. Association for Computational Linguistics.

Erick Skorupa Parolin, MohammadSaleh Hosseini, Yibo Hu, Latifur Khan, Patrick T Brandt, Javier Osorio, and Vito D'Orazio. 2022. Multi-coped: A multilingual multi-task approach for coding political event data on conflict and mediation domain. In *Proceedings of the 2022 AAAI/ACM Conference on AI, Ethics, and Society*, pages 700–711.

Bowen Tan, Zichao Yang, Maruan Al-Shedivat, E. Xing, and Zhiting Hu. 2021. Progressive generation of long text with pretrained language models. In *NAACL*.

Wilson L Taylor. 1953. "cloze procedure": A new tool for measuring readability. *Journalism quarterly*, 30(4):415–433.

Manghui Tu, Peng Li, I-Ling Yen, Bhavani M Thuraisingham, and Latifur Khan. 2008. Secure data objects replication in data grid. *IEEE Transactions on dependable and secure computing*, 7(1):50–64.

Chris Van Der Lee, Albert Gatt, Emiel Van Miltenburg, Sander Wubben, and Emiel Krahmer. 2019. Best practices for the human evaluation of automatically generated text. In *Proceedings of the 12th International Conference on Natural Language Generation*, pages 355–368.

Jonathan Voris, Nathaniel Boggs, and Salvatore J Stolfo. 2012. Lost in translation: Improving decoy documents via automated translation. In *2012 IEEE Symposium on Security and Privacy Workshops*, pages 129–133. IEEE.

Lei Wang, Chenglong Li, QingFeng Tan, and XueBin Wang. 2013. Generation and distribution of decoy document system. In *International Conference on Trustworthy Computing and Services*, pages 123–129. Springer.

Jason Wei and Kai Zou. 2019. Eda: Easy data augmentation techniques for boosting performance on text classification tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6382–6388.

Jonathan White and Dale Thompson. 2006. Using synthetic decoys to digitally watermark personally-identifying data and to promote data security. In *Security and Management*, pages 91–99. Citeseer.

Ben Whitham. 2013. Automating the generation of fake documents to detect network intruders. *International Journal of Cyber-Security and Digital Forensics*, 2(1):103.

Ben Whitham. 2017. Automating the generation of enticing text content for high-interaction honeyfiles. In *Proceedings of the 50th Hawaii International Conference on System Sciences*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi 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 M. 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.

Tongshuang Wu, Marco Tulio Ribeiro, Jeffrey Heer, and Daniel S Weld. 2021. Polyjuice: Generating counterfactuals for explaining, evaluating, and improving models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics*.

Kevin Yang and Dan Klein. 2021. Fudge: Controlled text generation with future discriminators. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3511–3535.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, 32.

L. Yao, N. Peng, R. Weischedel, K. Knight, D. Zhao, and R. Yan. 2019. Plan-and-write: Towards better automatic storytelling. In *Association for the Advancement of Artificial Intelligence (AAAI)*.

Jim Yuill, Mike Zappe, Dorothy Denning, and Fred Feer. 2004. Honeyfiles: deceptive files for intrusion detection. In *Proceedings from the Fifth Annual IEEE SMC Information Assurance Workshop, 2004.*, pages 116–122. IEEE.

Najam Zaidi, Trevor Cohn, and Gholamreza Haffari. 2019. Decoding as dynamic programming for recurrent autoregressive models. In *International Conference on Learning Representations*.

Rowan Zellers, Ari Holtzman, Elizabeth Clark, Lianhui Qin, Ali Farhadi, and Yejin Choi. 2021. Turingadvice: A generative and dynamic evaluation of language use. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4856–4880.

Rowan Zellers, Ari Holtzman, Hannah Rashkin, Yonatan Bisk, Ali Farhadi, Franziska Roesner, and Yejin Choi. 2019. Defending against neural fake news. In *Advances in Neural Information Processing Systems 32*.

Wanrong Zhu, Zhiting Hu, and E. Xing. 2019. Text infilling. *ArXiv*, abs/1901.00158.

A Implementation

We used the implementation of EDA from (Morris et al., 2020). We set the word swapping rate to 20%. For WE-FORGE, we learnt word-embeddings per dataset using Word2Vec (Mikolov et al., 2013). Based on silhouette scores and empirical observations on the validation sets, we set $k = 100$ for K-means and the number of Concept-Importance-Bins as 5 for all datasets.

For GPT-2 and FDI, we implemented the models with Huggingface API (Wolf et al., 2020) and monitored the training with Wandb (Biewald, 2020). We used Adam optimizer (Kingma and Ba, 2014) with a learning rate of 2e-5 and a batch size of 16. We chose proper sequence lengths for each dataset shown in Table 2. It took 1 to 2 days for these models to converge on the validation sets using a V-100 GPU.

B Other hyperparameters of FDI

For random masking in training, we traversed the document's hierarchy. We randomly masked sentences and then words with 5% probability. We then extended each selected word to a non-overlapped n-gram with a 50% probability. For controllable masking in inference, we set quantile lower bound q_{min} to 0.4, masking sentence's probability p_s to 0.7, masking concept's probability p_c to 0.5, sentence selection threshold t_s to 0.7, and max masked rate γ to 0.2 for all datasets.

C Questionnaire

Table 7 explains our quiz's instructions and questions. Figure 4 and Figure 5 show our designed user interfaces using Google Forms. Figure 6 and Figure 7 shows one example set of Quiz-1 and Quiz-2 in Google Forms, respectively.

Quiz-1 Assume you are a hacker. Can you distinguish the true document from the below 4 examples? Please choose the most likely option Top-1 and the 2nd possible option Top-2.

Quiz-2 Assume you are a cyber security expert. The ideal fake documents should be realistic and provide scalable protective coverage. They are "close enough" to the original to make the fakes believable, but sufficiently "far enough" to hide and protect private and confidential information. Now compared with the true document, would you evaluate the fakeness of the rest four fake samples? Below are the questions in details:

Q1. How do you rate the fluency of the article?
4. Overall flawless, with only minor typos.
3. Non-native, with minor but apparent errors.
2. Unnatural/synthetic, the apparent errors affect my reading.
1. Incomprehensible, with a lot of corrupted text.

Q2. How do you rate the coherency of the article? Does it make sense?
4. Coherent. There is a logical and consistent relation among the facts presented along the article.
3. Partially coherent, I can't understand what the author means in certain places.
2. Somehow confusing, with most parts of the document are confusing.
1. I have no (or almost no) idea what the author is trying to say.

Q3. Expert knowledge is required to identify this article is fake.
4. Agree, non-expert will find it difficult to distinguish if it is fake.
3. Partially agree, expert knowledge may help and speed up this process.
2. Somewhat disagree, expert knowledge might not be necessary.
1. Disagree, general audience can easily identify it is fake.

Q4. Is the sample "fake enough"? Does it apply necessary modifications (e.g., insert, replace and delete) to deceive the adversary and protect some essential facts? Note: High scores of fakeness do not mean superiority.
4. Excessive. The article may introduce too many changes, substantially diverging from the original topic/fact.
3. Moderate. The article introduces important changes, preserves the coherence, and seems realistic.
2. Marginal. The article introduces changes. However, considerable modifications do not significantly change facts presented in the original document.
1. Inadequate. Only insignificant modifications.

Q5. Based on your previous evaluation, how would you rank the fake documents? A good fake copy should look similar to the original document. But what's more important is that it also protects essential information and misleads hackers. Please rank your preference. (Top-1 the best to Top-4 the worst)

Table 7: The instructions and questions in the Quiz.

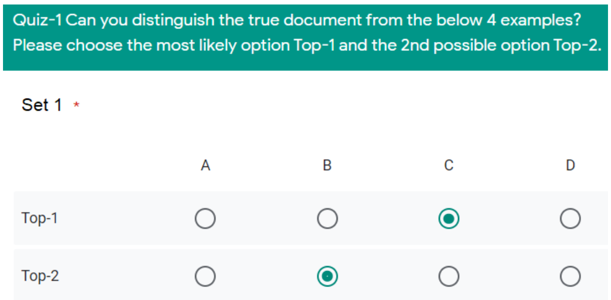

Figure 4: Quiz-1's user interface

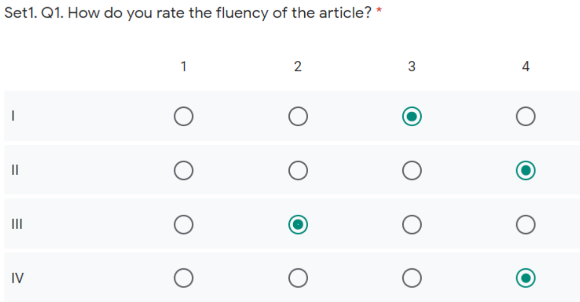

Figure 5: Quiz-2's user interface

A	B	C	D
Tomographic Image Reconstruction using Training images We describe and examine an algorithm for tomographic image reconstruction where prior knowledge about the solution is available in the form of training images. We first construct a nonnegative dictionary based on prototype elements from the training images; this problem is formulated as a regularized non-negative matrix factorization. Incorporating the dictionary as a prior in a convex reconstruction problem, we then find an approximate solution with a sparse representation in the dictionary. The dictionary is applied to non-overlapping patches of the image, which reduces the computational complexity compared to other algorithms. Computational experiments clarify the choice and interplay of the model parameters and the regularization parameters, and we show that in few-projection low-dose settings our algorithm is competitive with total variation regularization and tends to include more texture and more correct edges.	Tomographic Image Reconstruction using Training images We describe and examine an algorithm for tomographic image reconstruction where prior knowledge about the solution is available in the form of training images. We first construct Tensor Dictionaries based on a dictionary from the training images; this problem is formulated as a regularized non-negative matrix factorization. Incorporating the dictionary as a prior in videos requires 5 minutes, we then find nearly 3 times the amount of information in the dictionary. The dictionary is applied to 1.93 4x the volume of the image, which reduces the length of training images to other algorithms. This results in the choice and interplay of the dictionary and the size for the reconstruction of this problem, and we show that in many cases, our algorithm is competitive with previous algorithms and tends to include more texture and more accurate reconstruction results.	Tomographic Image Reconstruction using Training images We describe and examine an algorithm for tomographic image reconstruction where prior knowledge about the solution is available in the form of training images. We first construct a collection of missing patches based on images from the training images; this problem is formulated as a regularized non-negative matrix factorization. Incorporating the dictionary as a prior in the whole dictionary, we then find a similar estimate for missing patches in the dictionary. The dictionary is applied to the set of the image, which reduces the need of its own algorithm to other algorithms. We analyze the choice and interplay of the sparse dictionary of the image and the dictionary, and we show that in most cases our algorithm is competitive with well known algorithms and tends to include more texture and more robust visual features.	Tomographic Image Reconstruction using Training images We describe and examine an algorithm for tomographic image reconstruction where prior knowledge about the solution is available in the form of training images. We first construct an accelerated Newton method based on five images from the training images; this problem is formulated as a regularized non-negative matrix factorization. Incorporating the dictionary as a prior in such a scheme is very general, we then find an optimal local minimum in the dictionary. The dictionary is applied to the steps of reconstruction of the image, which reduces the reconstructed image compared to other algorithms. We show the choice and interplay of the dictionary and the last challenge is, and we show that in this paper, our algorithm is competitive with almost 70% reconstruction error in an IoU measurement set and tends to include more texture and more contrast.

Figure 6: A Quiz-1's example set consists of 1 true + 3 fake articles generated by an unknown model.

TRUE	I	II	III	IV
A Preliminary Study of Neural Network-based Approximation for HPC Applications Machine learning, as a tool to learn and model complicated (non)linear relationships between input and output data sets, has shown preliminary success in some HPC problems. Using machine learning, scientists are able to augment existing simulations by improving accuracy and significantly reducing latencies. Our ongoing research work is to create a general framework to apply neural network-based models to HPC applications. In particular, we want to use the neural network to approximate and replace code regions within the HPC application to improve performance (i.e., reducing the execution time) of the HPC application. In this paper, we present our preliminary study and results. Using two applications (the Newton-Raphson method and the Lennard-Jones (LJ) potential in LAMMP) for our case study, we achieve up to 2.7x and 2.46x speedup, respectively.	A Preliminary Study of Neural Network-based Approximation for HPC Applications Machine learning, as a tool to learn and model complicated (non)linear relationships between input and output data sets, has shown preliminary success in some HPC problems. Part of the range of successful strategies is the solution of complex hidden Markov decision processes, such as recommendation. We present and review the main algorithms of the neural network-based approximation approach for the error correction of these models by imposing the weights or predictors on input data with zero accuracy for each input and zero for all output values. We compare our algorithmic results with existing heuristic algorithms, based on the NRC-GALICE-CM algorithm. We thoroughly analyse the effect of the weights on the error, in addition to the quality of the solution and the related performance. Some representative examples, ranging from machine-learning to statistical inference, and we illustrate the relevance of our findings for further research.	A Preliminary Clarify of Neural Network-based Preconditioning for Floatingpoint Applications Avenue editing, as a tool to learn and model complicated (fiedler) linear relationships between input and unseen data sets, has shown preliminary capability in some Floatingpoint problems. Using avenue editing, scientists are able to augment existing simulations by improving accuracy and significantly reducing latencies. Our ongoing research work is to create a general framework to apply neural network-based models to Floatingpoint applications. In particular, we want to use the neural network to approximate and replace code regions within the Floatingpoint correctbyconstruction to improve performance (i.e., reducing the titan time) of the Floatingpoint correctbyconstruction. In this paper, we present our preliminary clarify and results. Using two applications (the Newton-Raphson method and the Lennard-Jones (RI1lae) potential in LAMMP) for our case clarify, we achieve up to 2.7x and 2.46x speedups, respectively.	A Preliminary Study of Neural Network-based Approximation for HPC Applications Machine learning, as a tool to learn and model complicated (non)betwixt linear relationships between input and output data determine sets, has shown preliminary success in some about HPC problems. Using machine learning, scientists are to augment simulations by improving and significantly reducing latencies. Our ongoing research general is apply create a work framework to to neural network-based models to HPC applications. In particular, we want to use neuronic the neural network to approximate and replace code regions within the HPC application to improve performance (i.inside e., reducing the eastward execution time) of the HPC application. IN this paper, we present our preliminary contemplate and results. Using two cogitation applications (the deuce Newton-Raphson method and the Lennard-Jones (two LJ) potential in LAMMP) for our case study, we achieve up to 2.7x and 2.46x speedup, respectively.	A Preliminary Study of Neural Network-based Approximation for HPC Applications Machine learning, as a tool to learn and model complicated (non)linear relationships between input and output data sets, has shown preliminary success in some HPC problems. Using machine learning, scientists are able to construct approximate models. Our first contribution is to create possible approximate model to apply neural network-based models to HPC applications. In particular, we want to use the first layer to use the error metric to approximate and encode subspaces within the Fused MultiLayer Perceptron (i.e., reducing the neighborhood factor) of the model. In this paper, we present our preliminary study and results. Using two applications (the oldset and the Lennard-Jones (LJ) potential in LAMMP) for our last two applications, we achieve up to 2.7x and 5x improvements in prediction, respectively.

Figure 7: A Quiz-2's example set includes 1 known true document + 4 fake samples generated by 4 models in an unknown order.