TRAPDOC: Deceiving LLM Users by Injecting Imperceptible Phantom Tokens into Documents

Hyundong Jin **Sicheol Sung** **Shinwoo Park** **SeungYeop Baik** **Yo-Sub Han**⋆
Yonsei University, Seoul, Republic of Korea
{tuzi04,sicheol.sung,pshkhh,sybaik2006,emmous}@yonsei.ac.kr

Abstract

The reasoning, writing, text-editing, and retrieval capabilities of proprietary large language models (LLMs) have advanced rapidly, providing users with an ever-expanding set of functionalities. However, this growing utility has also led to a serious societal concern: the over-reliance on LLMs. In particular, users increasingly delegate tasks such as homework, assignments, or the processing of sensitive documents to LLMs without meaningful engagement. This form of over-reliance and misuse is emerging as a significant social issue. In order to mitigate these issues, we propose a method for injecting imperceptible phantom tokens into documents, which causes LLMs to generate outputs that appear plausible to users but are in fact incorrect. Based on this technique, we introduce TRAPDOC, a framework designed to deceive over-reliant LLM users. Through empirical evaluation, we demonstrate the effectiveness of our framework on proprietary LLMs, comparing its impact against several baselines. TRAPDOC serves as a strong foundation for promoting more responsible and thoughtful engagement with language models. Our code is available at https://github.com/jindong22/TrapDoc.

1 Introduction

Large Language Models (LLMs) have recently excelled in a wide range of tasks, including text editing, summarizing, searching, and reasoning. Yet the rising adoption of LLMs is not without downsides. As more users turn to these models, instances of LLM misuse and negative side effects are increasingly reported: accessing harmful information (Zou et al., 2023; Perez et al., 2022; Mazeika et al., 2024), generating fake facts (Maynez et al., 2020; Manakul et al., 2023), leaking personally identifiable information (Carlini et al., 2021; Kim et al., 2023), and producing hate speech (Ahn et al., 2024; Kim et al., 2022). Among these issues, academic cheating has emerged as one of the most serious. Many students and researchers now rely on LLMs to complete assignments or conduct research, and some of them entrust the entire reasoning process to the model. In scholarly writing and peer-review workflows, words that humans rarely use but that LLMs frequently employ appear more often (Juzek and Ward, 2025). However, since LLMs have become capable of retrieving and answering most undergraduate-level questions convincingly, distinguishing human-written text from LLM-generated text is becoming difficult.

Numerous methods identifying LLM-generated content have been proposed to combat this trend. Typical approaches involve re-inputting the suspect text into an LLM, computing perplexity scores, or prompting the model to paraphrase the contents and then comparing the degree of alteration. These methods aim to determine whether the text was

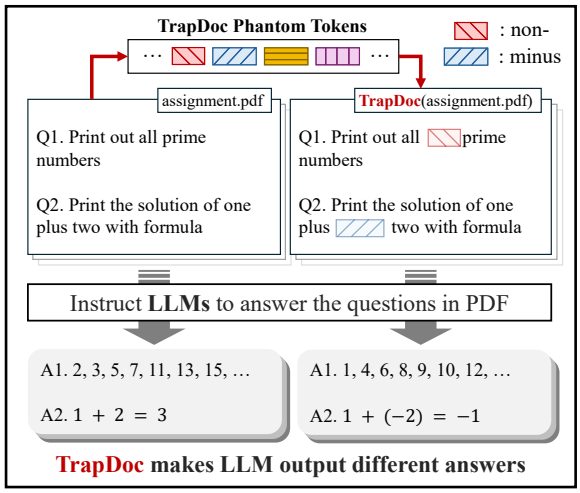

Figure 1: A motivation example of our TRAPDOC framework. The phantom tokens injected into a PDF document by TRAPDOC alters LLM answers.

⋆ Corresponding author.

Findings of the Association for Computational Linguistics: EMNLP 2025, pages 18881–18897
November 4-9, 2025

generated by a machine or authored by a human. Such methods perform reasonably well for natural-language text. However, merely detecting whether a passage was machine-generated is not the same as determining whether the own thinking of the writer is reflected in the final product. For example, if a researcher uses an LLM only to translate or grammatically polish a manuscript, the resulting text may be flagged as LLM-generated even though it still embodies the original ideas of the author and approach. In other words, there are applications where LLM assistance does not count as cheating since the user's unique reasoning is preserved. Existing detection methods struggle to differentiate between mindless use of an LLM and its use as an assistant, limiting their effectiveness for spotting genuine academic cheating. We distinguish these two type of uses by inducing errors that become apparent with careful reading.

We present a simple yet effective adversarial-text method that exploits an overlooked vulnerability in the way large language models process PDF files. Previous approaches have focused on visible modifications, such as character substitutions, synonym replacements, or paraphrases, to degrade model performance. However, these methods are inherently limited because each alteration is easily noticeable in discrete text. We target LLMs that read contents of PDFs with byte-stream parsing, which can perceive invisible texts. Leveraging this fact, we define an adversarial insertion task by presenting a clear problem definition for PDFs. We then propose TRAPDOC, which injects imperceptible phantom tokens that mislead LLMs while leaving the appearance of document intact. Experiments across multiple tasks and baselines confirm the practical value of TRAPDOC. We believe it contributes to promoting ethical use of LLMs and helps deter academic misconduct.

2 Related Work

2.1 Text Adversarial Attack

An adversarial attack involves subtly modifying an input of a model to alter the output of the model (Szegedy et al., 2014; Goodfellow et al., 2015). Generally, the more imperceptible the perturbation and the greater the resulting change in output, the more effective the attack is considered. Since data in the vision and audio domains are continuous, injecting small amounts of noise is relatively straightforward. In the text domain, however, each token modification is visible, so creating undetectable perturbations is far more difficult (Jia and Liang, 2017; Ribeiro et al., 2018). Most existing techniques modify the source text by deleting, replacing, swapping, or inserting characters or words (Ribeiro et al., 2018; Ebrahimi et al., 2018; Li et al., 2019), which inevitably alters the surface form. Since these edits are usually visible and easily noticed, the prevailing approach is to preserve the original semantics while perturbing the text enough to change the output of an LLM (Alzantot et al., 2018; Jin et al., 2020; Li et al., 2020; Garg and Ramakrishnan, 2020). The model being attacked is commonly referred to as the *victim*, and attacks are classified as *white-box* or *black-box* depending on the attacker's level of access.

2.1.1 White-box Victim Attack

In a white-box attack, the adversary has full access to the model's outputs, parameters and other internal components (Ebrahimi et al., 2018; Li et al., 2019; Wallace et al., 2019; Boucher et al., 2022; Zhang et al., 2024). Under this setting, a wide range of techniques can be employed, including manipulating the model's embeddings, performing targeted fine-tuning, and crafting gradient-based perturbations. Since the gradients are directly available, one can estimate the importance of each token from the output logits, then deliberately introduce misspellings (Ebrahimi et al., 2018), replace tokens with synonyms (Jin et al., 2020; Li et al., 2020; Garg and Ramakrishnan, 2020), or swap them with visually similar Unicode characters to mount the attack (Zhu et al., 2024).

Zhang et al. (2024) conducted a study in which they uploaded documents containing invisible text to the internet using a white-box approach, in order to have their malicious information included in the retrieval-augmented generation of LLMs. However, the proprietary LLMs that are most often misused in academic contexts are either closed-source or so large that individual users cannot realistically run them. Consequently, white-box approaches are not well suited to addressing today's academic LLM-misuse problem.

2.1.2 Black-box Victim Attack

In a black-box setting, access to the model is significantly more restricted than in a white-box setting. It is typically assumed that only the final output of the LLM or logits of the LLM are observable. Due to this limited access, altering the model's output

is more challenging than in white-box scenarios. Traditional approaches often rely on iterative token-level modifications, identifying influential tokens by measuring the impact of each change on the output (Formento et al., 2023; Zhu et al., 2024; Jin et al., 2020; Li et al., 2020; Garg and Ramakrishnan, 2020). However, such iterative inference scales with sequence length, leading to increased computational overhead. When applied to proprietary LLMs, it also incurs a monetary cost due to usage-based pricing.

Moreover, modern proprietary LLMs are robust to a wide variety of inputs, including paraphrases and typographical errors, which reduces the effectiveness of existing methods. In a recent study, Xu et al. (2024) proposed leveraging the victim LLM itself to generate adversarial inputs, and we adopt this approach as our baseline.

2.2 Adversarial Text for Evaluating LLM

There has been extensive research on assessing and improving model comprehension and reliability through adversarial text. Jia and Liang (2017) evaluated the model's text understanding by appending adversarial sentences to paragraphs and measuring the resulting changes in predictions. Other works (Li et al., 2023; Zang et al., 2020; Abad-Rocamora et al., 2024) aimed to increase the difficulty of robust natural language inference tasks by introducing adversarial candidates.

Additionally, early jailbreaking techniques, used to assess LLM safety, share methodological similarities with adversarial prompting. However, unlike these studies, our work does not aim to evaluate LLMs themselves. Instead, we focus on preventing their misuse in academic assessment scenarios, and thus these works are considered out of scope.

3 Backgrounds

3.1 Proprietary LLM Eyesight Test

As a preliminary experiment, we aimed to investigate how LLMs read text within PDFs. We created a "LLM Eyesight PDF" that included black text with varying levels of opacity, white text with different opacity ranges, and text of various font sizes. We then prompted the LLMs to read the text from the PDF. According to the experimental results, GPT and Claude, when used via interactive web interfaces, were able to read low-opacity text, white text, and even text with font size 0. In contrast, DeepSeek, Gemini, and Grok were unable to read white or transparent text. Through additional prompting, we examined how each LLM is capable of reading PDF content. Only GPT and Claude successfully read invisible texts embedded in the PDF, whereas DeepSeek, Gemini, and Grok cannot.

More detailed results and the prompts used can be found in Appendix A. Based on these results, we hypothesize that ChatGPT and Claude read PDFs through the PDF's graphic operators stream, and we design our framework based on this assumption.

3.2 PDF Parsing

PDF is a widely used document format that represents text using coordinate-based text-boxes and visualizes various types of objects, such as tables and images, using predefined graphic operators. There are several ways to create invisible text in PDFs, with common methods including setting gray levels using the `g` and `G` operators or entering an invisible mode using the `Tr` operator.

Zhang et al. (2024) proposed methods for making text invisible by adjusting opacity, overlaying text with images, and using JavaScript triggers. However, these approaches have limitations, as they are often difficult to apply to standard PDFs and viewers. Moreover, when text is inserted transparently, it still occupies space and remains selectable or searchable, which is a drawback. When text is hidden using images, it tends to appear at the beginning or end of paragraphs during text extraction, making it difficult to affect the content.

Instead of relying on such naive embedding techniques, we developed a software tool that directly captures and modifies the `TJ` and `Tj` operators, which are standard instructions for rendering texts in a PDF stream. This approach enables the insertion of targeted content between rendering operations, allowing us to embed text of arbitrary length into any PDF without altering its visible layout.

Previous work has struggled to maintain semantic similarity while distorting the model's output. In contrast, our approach is an insertion-only technique based on supersequences that does not modify the original text at all and, as mentioned earlier, can be applied broadly to standard PDFs. To the best of our knowledge, we are the first to propose an adversarial text generation task that allows only insertions without requiring semantic similarity.

3.3 Problem Formulation

In real-world academic assessments, evaluators typically assign a task to a participant, expecting them

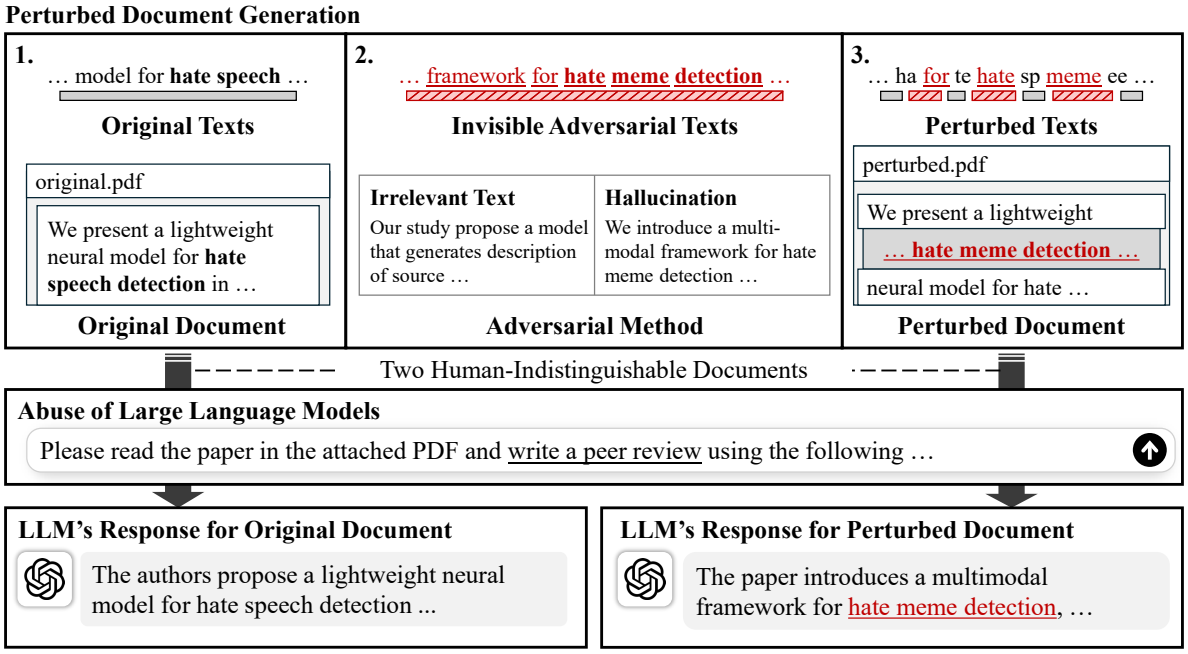

Figure 2: Overview of the TRAPDOC framework. The framework (1) extracts the original texts from a given document, (2) generates adversarial variants of the original, and (3) produces a perturbed document by shuffling the original and adversarial texts. The adversarial segments are imperceptible to humans, making the perturbed document indistinguishable from the original. In contrast, LLMs process both the original and adversarial texts, leading to incorrect outputs.

to comprehend the assignment and respond accordingly. However, a growing concern is the misuse of proprietary LLMs, where individuals submit the assignment prompt directly to an LLM without understanding its content, relying entirely on the model's response.

Our objective is to construct a scenario in which such misuse results in output that appears plausible but is, in fact, incorrect. Formally, given a document D, we aim to generate a perturbed version D' by embedding imperceptible adversarial tokens. These tokens remain invisible to human readers, ensuring that D' remains indistinguishable from D. At the same time, they are designed to induce significantly altered response from the LLM, thereby revealing to careful readers that the output was machine-generated.

As noted earlier, our objective is to distort an LLM's output by inserting invisible text while preserving the visual appearance of the original document. The ideal strategy would be to add a few words or sentences to an existing sentence so that its meaning is subtly altered. This strategy, however, is subject to stringent constraints: the original sentence must remain a subsequence, the resulting text must be grammatically correct, and yet the overall meaning must change. Instead of identifying tokens that blend naturally into the sentence, we divide the original text into finer-grained segments, making the resulting output appear noisy.

4 TRAPDOC Framework

In this section, we present TRAPDOC, a document perturbation framework designed to corrupt the outputs of LLMs. Figure 2 provides an overview of the TRAPDOC framework. Our approach consists of two main components: (1) perturbing the contents of a given document to generate adversarial texts, and (2) injecting the resulting adversarial texts into the document to distract LLMs from correctly understand the original contents.

4.1 Text Perturbation Method

Our goal is to induce the LLM to produce responses that seem plausible and relevant to the document at first glance but are actually incorrect. Therefore, the injected imperceptible text must remain contextually related to the source passage while conveying different instructions or facts. We achieve this by prompting an LLM to generate a hallucinated version of the given text. We refer to such LLM-generated perturbations as *hallucinations*.

By crafting adversarial passages that resemble the original yet differ in content, we construct inputs that mislead the target LLM. These hallucinations are subsequently embedded into the document via the text injection method.

4.2 Text Injection Method

Our text-injection procedure operates by manipulating the PDF operator stream. First, we parse the source PDF to extract its operator stream and iterate through the stream. When we encounter an operator that places text, we extract the associated string and split it into segments of n characters. An intractable word is then inserted between the character segments. The modified operator stream is subsequently un-parsed to reconstruct a new PDF.

Text rendered with a font size 0 is not displayed by most PDF viewers—including Adobe Reader, Chrome, and Apple Preview—and cannot be discovered by dragging or text search. Building on this property, we embed adversarial text of arbitrary length at arbitrary positions, ensuring that the LLM processes both the inserted and original content.

5 Experimental Setup

We conduct experiments under realistic usage scenarios to evaluate the practical effectiveness of TRAPDOC. The target LLM must satisfy two conditions: (1) it must accept PDF files as input and (2) it must be capable of reading text rendered invisible to human readers. Based on these criteria, we select two publicly accessible models: GPT-4.1, OpenAI's long-context flagship model, and o4-mini, a lightweight but strong reasoning model.

5.1 Evaluation Tasks

We consider the scenario of a user who frequently relies on LLMs despite being explicitly prohibited from doing so, and our objective is to detect such unauthorized use. We evaluate TRAPDOC on three distinct tasks to simulate this.

The first is *code generation from natural-language (NL) specifications*, for which we adopt the MBPP+ dataset (Liu et al., 2023). The second is *paragraph summarization*, evaluated on CNN/DailyMail (Nallapati et al., 2016). The third is *paper reviewing*, using Qasper (Dasigi et al., 2021). Because CNN/DailyMail contains a large number of instances, we randomly sample 300 articles. For Qasper, due to the length of inputs, we sample 100 documents to remain within budget constraints. The detailed information about the dataset is provided in Appendix D.1.

5.2 Perturbation Baselines

TRAPDOC is compared against three perturbation baselines. *Irrelevant text* inserts the description of a different instance from the same task. *Meta instruction* encloses the original paragraph in quotation marks and then appends a meta-level instruction that contradicts the quoted content. *Negation* negates every sentence in the paragraph. We use `negate`[2] to generate negations.

Most prior attacks require white-box access or an excessive number of model queries, and paragraph-level LLM-driven perturbations remain largely unexplored. Xu et al. (2024) proposes PromptAttack, sub-sentence-level LLM adversarial attacks. We port its perturbations as an additional baseline, but only for the code-generation task because the method does not scale beyond the sentence level. Table 1 summarizes each perturbation method. Full implementation details for all baselines are provided in Appendix E.

Irrelevant: *In-domain data different from the given text.*
Meta Instruction: *Inserting instruction that asserts the paragraph is factually incorrect.*
Negation: *Negating sentences by systematically adding or removing lexical negators, e.g., 'not' or 'no'.*
PromptAttack (w2): *Removing less significant two words.*
PromptAttack (s1): *Adding meaningless handles after sentences.*
Hallucination: *Prompting the LLM to deliberately introduce hallucinated content into the given text.*

Table 1: Perturbation methods used to generate adversarial texts from original texts in our experiments.

5.3 Evaluation Metrics

As we stated earlier, our goal is to deceive LLMs so that their outputs appear plausible but are actually incorrect. To this end, we evaluate model outputs using two types of metrics: *surface-level*

[2] https://pypi.org/project/negate/

similarity, which assesses syntactic overlap, and *meaning-based similarity*, which evaluates semantic alignment. This distinction allows us to analyze how well the model preserves the textual form versus the underlying meaning under perturbation. Detailed explanations of each metric appear in Appendix D.2.

5.3.1 Surface-level similarity

Surface-level similarity metrics are used to evaluate how well TRAPDOC preserves the superficial form of the output. For the code generation, we report CodeBLEU and Stanford Moss similarity scores—commonly used to assess code similarity and detect plagiarism. For the summarization and review-generation tasks, we report ROUGE-1, ROUGE-2, ROUGE-L, BLEU-1, and BLEU-2. These n-gram and subsequence-based metrics measure overlap at the word and short phrase levels, reflecting surface similarity without strongly encoding semantic content. Since TRAPDOC is designed to preserve the surface form of the LLM's output, an effective attack should maintain high surface-level similarity scores even as it alters the underlying semantics.

5.3.2 Meaning-based similarity

Meaning-based similarity metrics assess how well TRAPDOC disrupts the meaning of the output. For code generation, we use pass@k, a standard evaluation metric used in code-synthesis benchmarks. We report only pass@1, as commercial LLMs typically perform well on MBPP+. For the summarization and review-generation, we use BERTSocre, which compares contextual meaning using BERT embeddings. In our setup, a lower meaning-based similarity score indicates a more successful perturbation.

6 Results and Analysis

6.1 Code Generation

Table 2 presents the pass@1, CodeBLEU, and Stanford Moss results for the code-generation task when each text-perturbation method is applied to the input PDF. All techniques reduce pass@1, though the extent of the decrease varied significantly. As expected, Irrelevant caused the greatest degradation, driving pass@1 to zero by tricking the model into treating the invisible input as a completely different coding problem. Our own method also pushed pass@1 down to the single-digit range, confirming that the perturbation can severely impair an LLM's effectiveness. However, both methods also cause low CodeBLEU and Moss scores, which represent the surface-level similarities. From the perspective of code similarity, greater semantic changes naturally lead to lower similarity scores. This is because—unlike natural languages—code has strict logical structures, and altering the logic affects similarity metrics accordingly. Therefore, in code generation, there exists an inherent trade-off between surface-level similarity and meaning-based similarity.

Method	GPT-4.1	o4-mini
No Perturbation	78.84	80.16
Irrelevant	0.00	0.00
Meta Instruction	66.93	13.23
Negation	74.60	72.49
PromptAttack (w2)	70.63	38.10
PromptAttack (s1)	29.10	60.85
Hallucination (Ours)	6.88	3.17

Table 2: Code generation results on MBPP+ dataset. We report the pass@1 of the generated code. For PromptAttack, we include only the best-performing variant for each model: w2 for GPT-4.1 and s1 for o4-mini.

By contrast, Negation produced only a minor change even though the injected description explicitly told the model not to implement the program. Because the prompt simultaneously asked the system to solve the task, it apparently ignored the negated content and proceeded as usual. Meta Instruction showed extreme variance: for o4-mini it often triggered a "no PDF access" reply—156 such cases were observed on manual inspection—even though the files were perfectly readable and the same model performed well on other conditions. Meta Instruction that claimed the PDF was faulty seems to have interfered with o4-mini's parsing. GPT-4.1, on the other hand, remained largely unaffected and still achieved a high pass@1.

For the PromptAttack baseline we tried all nine perturbations proposed in the original work and report the two most effective for each model. Although these attacks did hurt pass@1, none of them produced a consistently large drop across the two systems. Also, CodeBLEU and Moss scores remain high, suggesting that the output semantics have not been severely degraded. Overall, our approach delivered the most reliable performance loss among all baselines.

Method	GPT-4.1						o4-mini					
	BLEU (↑)		ROUGE (↑)			BERT (↓)	BLEU (↑)		ROUGE (↑)			BERT (↓)
	1	2	1	2	L		1	2	1	2	L	
No Perturbation	23.94	11.77	29.69	8.07	26.20	87.07	24.00	11.75	29.64	8.05	25.98	86.68
Irrelevant	10.16	1.28	11.30	0.67	10.15	81.36	9.88	1.44	10.88	0.81	9.82	80.93
Meta Instruction	19.47	8.56	24.48	5.79	21.93	85.67	16.96	6.94	21.59	4.60	19.27	84.79
Negate	24.73	12.38	30.46	8.56	27.02	87.15	23.45	11.18	28.95	7.48	25.32	86.67
Hallucination (Ours)	16.46	5.37	19.36	3.01	17.37	85.41	14.99	4.12	17.96	2.35	15.80	84.66

Table 3: Comparison of the effectiveness of perturbation methods on the CNN/DailyMail Summarization dataset. We evaluate the performance of GPT-4.1 and o4-mini using BLEU-1, BLEU-2, ROUGE-1, ROUGE-2 and ROUGE-L to measure syntactic similarity, where higher scores are better (↑). In contrast, BERTScore (denoted as BERT) is used to assess semantic similarity, where lower scores are preferred (↓).

6.2 Text Summarization

Table 3 summarizes the results for the summarization task. Irrelevant Text again performed worst on every metric because, unlike the other methods, it makes the LLM treat the PDF as a completely different paragraph, leading to very low scores. This outcome strongly supports our hypothesis that the model relies on the injected strings when forming its summary and also justifies our metrics.

At the opposite end of the spectrum, Negation scored almost the same as the unperturbed baseline; manual examination showed that the summaries were nearly identical, so the negation strategy was no more effective here than in code generation. Between these extremes, Meta Instruction and our hallucination-based perturbation both achieved relatively high ROUGE and BLEU together with noticeably lower BERTScore. Manual analysis revealed a key difference: the hallucination texts often inserted named entities that never appeared in the source, which explains their even lower semantic similarity.

6.3 Review Generation

Finally, we measured attack strength on the review generation task. The overall pattern mirrors the summarization results, though absolute values are higher because our prompt forces reviews into a standard format, creating inevitable overlaps. Irrelevant Text again produced the lowest scores and frequently made the model confuse one paper for another. A notable change is that Meta Instruction now outperformed Negation on every metric. Our own method maintained high syntactic overlap while keeping BERTScore relatively low, indicating that it preserves surface form yet still diverts meaning. We provide a detailed case study in a later section to illuminate the distinct behaviors of each perturbation strategy.

6.4 Case Study

We conduct a blind human evaluation as a case study for the peer review generation task. We assess the LLM-generated reviews under each adversarial method, focusing on the following criteria:

1. **Hallucinated Content:**
   Presence of unsupported information.
2. **Consistency with Authors:**
   Logical agreement the authors' claims.
3. **Detail Precision:**
   Fidelity of measurements and statistics.
4. **Intent Comprehension:**
   Understanding of the paper's motivation.

The original paper used as the foundation for this experiment is "DuTongChuan: Context-aware Translation Model for Simultaneous Interpreting" (Xiong et al., 2019).

Prior to review generation, we introduced different forms of textual perturbations into the PDF and analyzed how these alterations affected the accuracy and fidelity of the resulting reviews. Five perturbation types were considered: (1) Base (No Perturbation); (2) Hallucination; (3) Irrelevant Insertion; (4) Meta Instruction; and (5) Negation. For Irrelevant Insertion, we use the following paper: "Self-Attention and Ingredient-Attention Based Model for Recipe Retrieval from Image Queries" (Fontanellaz et al., 2019).

For each of five perturbation types, evaluators assess two LLM-generated reviews. Evaluators remain unaware of the perturbation type applied to each input and receive instructions to identify factual errors and misinterpretations with respect to the original paper.

Method	GPT-4.1						o4-mini					
	BLEU (↑)		ROUGE (↑)			BERT (↓)	BLEU (↑)		ROUGE (↑)			BERT (↓)
	1	2	1	2	L		1	2	1	2	L	
Irrelevant	40.92	22.65	33.73	13.32	31.77	84.94	30.31	12.82	26.01	6.18	24.57	82.66
Meta Instruction	49.26	29.43	42.27	18.34	39.76	88.63	43.36	23.48	40.14	14.08	37.96	88.33
Negate	48.72	29.05	41.32	18.07	38.86	88.24	40.02	20.17	36.12	11.30	34.07	86.82
Hallucination (Ours)	46.18	26.08	38.12	15.25	35.84	87.24	36.72	17.35	32.47	9.07	30.73	85.57

Table 4: Comparison of the effectiveness of perturbation methods on the paper reviewing task. Since no human-generated reviews are available, we use LLM-generated review without perturbation as the reference.

Notably, reviewers consistently identified reviews generated under the Hallucination and Irrelevant Insertion conditions as problematic. For instance, a review from Hallucination include multiple fabricated terms and evaluation metrics not present in the original paper, such as *Semantic Block detector* and *Segmental Coherence Score*. Similarly, reviews from *Irrelevant Insertion* incorporate concepts and datasets unrelated to the DuTongChuan paper (Xiong et al., 2019), strongly suggesting that the LLM have absorbed and reproduced content from a different source. In contrast, reviews from Negation and Meta Instruction are consistently rated as accurate and well-aligned. These reviews do not include overt factual errors and preserve the structural integrity of the original claims.

This case study reveals a clear pattern in the susceptibility of LLMs to different classes of textual perturbation. Human evaluators, unaware of the specific manipulations, reliably identified factual inconsistencies and topical deviations in reviews generated from documents subjected to hallucination and irrelevant insertion. These forms of corruption directly altered the semantic content by introducing extraneous or fabricated material, thereby leading to detectable distortions in the generated outputs. In contrast, semantic-level perturbations—negation and meta-instruction—proved less effective in deceiving both humans and models. Despite syntactic manipulation, these perturbations preserved the lexical surface and discourse structure of the original context, which enabled the LLM to produce coherent reviews. Case study examples are provided in Appendix F.

7 Discussion

7.1 File Size Increase

Our text insertion method generally requires an amount of text comparable to, or sometimes exceeding, the original text. As a result, an increase in file size is inevitable. Nevertheless, since text consumes significantly less storage than images, and considering that our primary application is academic use, we believe that preventing LLM abuse is a more important concern. In our experiments on academic papers, the average document size prior to perturbation was 0.908 MiB. After applying TRAPDOC perturbations, the average file sizes increased modestly to 1.023 MiB and 1.019 MiB when targeting GPT-4.1 and o4-mini hallucinations, respectively. These represent increases of approximately 12% and 13%, which remain within a practical and manageable range. Importantly, this increase is not substantial, as the inserted phantom tokens are textual in nature and their contribution to file size is negligible compared to multimedia or graphical content. Our perturbation technique therefore still holds considerable potential for optimization and with further refinement, the overhead associated with file size could be mitigated effectively.

7.2 Robustness on Copy-and-Paste Bypass

An important concern is whether our methodology can be trivially bypassed through simple techniques such as copy-and-paste, or whether it could be easily detected and removed. Such details may vary depending on implementation choices, the PDF editing library employed or the specific reader software, but in general, detection, removal, and bypass present a trade-off relationship. Specifically, making removal and bypass more difficult typically requires heavier perturbation of the text, which in turn makes detection easier; conversely, minimizing perturbation improves stealth but leaves the possibility of removal or bypass. Crucially, TRAPDOC inserts tokens between sub-token fragments of the original text, meaning that even if text is extracted, reconstructing the original content remains

inherently difficult. Thus, while the difficulty of detection or bypass may depend on the environment, we emphasize that complete removal of the perturbations is fundamentally challenging.

7.3 Mitigating LLM Misuse

LLMs are increasingly embedded in educational and academic workflows, raising concerns about over-reliance and automated misuse. In particular, users may submit LLM-generated responses without attempting to understand the source material, undermining the validity of evaluation processes. This issue is especially pressing in scenarios where critical reasoning, comprehension, or creativity is being assessed.

TRAPDOC offers a practical defense against such misuse. By injecting imperceptible adversarial text into documents, it exposes users who depend blindly on LLMs to interpret and respond. While human readers perceive no change, LLMs ingest the hidden strings, often leading to distorted or incoherent outputs. This discrepancy can serve as a signal to educators, reviewers, or evaluators that a submission was not independently composed.

Rather than discouraging responsible LLM usage, TRAPDOC encourages deeper engagement with assigned tasks. It functions as a deterrent for blind automation while preserving space for thoughtful human-AI collaboration. In doing so, it contributes to the development of more robust, fair, and context-sensitive evaluation practices.

Although this work focuses on education and peer review, the underlying principles of TRAPDOC can extend to a broader set of high-stakes domains. Legal drafting, clinical documentation, and policy generation increasingly rely on LLMs, yet all require traceable authorship and semantic fidelity. Document-level interventions like TRAPDOC offer a minimally invasive yet effective means of flagging passive AI delegation in such contexts.

8 Conclusion

Through our experiments, we gained insights into how proprietary LLMs perceive PDFs and proposed a problem definition, methodology, evaluation metrics, and the framework aimed at deceiving users who are over-reliant to LLMs. We conducted evaluations on three diverse tasks that reflect real-world LLM misuse scenarios, demonstrating the significance and effectiveness of our framework. Additionally, we performed both quantitative and qualitative analyses on relatively large, paper-length documents, and further validated our framework's effectiveness in mitigating LLM abuse through human evaluation.

Our findings reveal weaknesses in how LLMs are currently employed for document-related tasks. By identifying these vulnerabilities, we offer methods that can be leveraged to detect and mitigate potential misuse of LLMs in both academic and professional contexts. Our approach will encourage a critical reassessment of prevailing practices, applied to address academic misuse and to hinder unauthorized data collection by LLMs. These directions will contribute to the development of more responsible and ethical applications of large language models.

Limitations

Our methodology is entirely based on the assumption that proprietary LLMs recognize strings from the PDF operator stream. A limitation of our invisible text approach is that it does not apply to models such as DeepSeek or Gemini, as discussed in Section 3.1, which interpret PDFs as images, or in settings that rely on OCR-based text recognition or screenshot-based inference. However, image-based text recognition technologies still face challenges in achieving fully accurate extraction, particularly for languages other than English, where performance is often restricted. Furthermore, researches in the vision domain (Chen et al., 2020; Xu et al., 2023) have explored techniques designed to deliberately interfere with OCR (e.g., adversarial perturbations to hinder text recognition). We believe that integrating our approach with such methods represents a promising direction for future work.

Ethical Considerations

Our work includes an approach that deliberately deceives an LLM user. However, the goal of this research is not to promote deception but rather to curb excessive reliance on LLMs and to enable fair, human-centered evaluation. TRAPDOC is designed as a defensive tool for educators, reviewers, and assessment platforms that need to verify genuine human understanding in settings where proprietary LLMs are otherwise prohibited. By embedding invisible distractors, the framework discourages "push-button" solutionism and encourages users to engage with the material directly.

All datasets used—MBPP+, CNN/DailyMail,

and Qasper—are publicly available under licenses that permit research. No personal or sensitive data were collected or processed, and no private documents were exposed. We hope the work will (1) raise awareness of hidden-text vulnerabilities in document pipelines, (2) prompt LLM providers to harden PDF ingestion, and (3) give educators a realistic lever against unreflective, wholesale LLM use. Ultimately, we view TRAPDOC as a step toward more responsible and transparent integration of language models into high-stakes evaluation settings.

Acknowledgements

This research was supported by the National Research Foundation of Korea (NRF) grants funded by the Korean government (MSIT) (RS-2025-02222626 and RS-2025-00562134), and by the AI Graduate School Program (RS-2020-II201361). Author Contributions: Jin and Sung contributed equally to this work as co-first authors.

References

Elías Abad-Rocamora, Yongtao Wu, Fanghui Liu, Grigorios Chrysos, and Volkan Cevher. 2024. Revisiting character-level adversarial attacks for language models. In *Proceedings of the 41st International Conference on Machine Learning*. OpenReview.net.

Hyeseon Ahn, Youngwook Kim, Jungin Kim, and Yo-Sub Han. 2024. SharedCon: Implicit hate speech detection using shared semantics. In *Findings of the Association for Computational Linguistics*, pages 10444–10455.

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.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and 1 others. 2021. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*.

Nicholas Boucher, Ilia Shumailov, Ross Anderson, and Nicolas Papernot. 2022. Bad characters: Imperceptible NLP attacks. In *43rd IEEE Symposium on Security and Privacy*, pages 1987–2004. IEEE.

Nicholas Carlini, Florian Tramer, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Ulfar Erlingsson, and 1 others. 2021. Extracting training data from large language models. In *Proceedings of the 30th USENIX security symposium*, pages 2633–2650.

Lu Chen, Jiao Sun, and Wei Xu. 2020. Fawa: Fast adversarial watermark attack on optical character recognition (ocr) systems. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 547–563. Springer.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, and 1 others. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Pradeep Dasigi, Kyle Lo, Iz Beltagy, Arman Cohan, Noah A. Smith, and Matt Gardner. 2021. A dataset of information-seeking questions and answers anchored in research papers. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4599–4610.

Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. 2018. HotFlip: White-box adversarial examples for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, pages 31–36.

Matthias Fontanellaz, Stergios Christodoulidis, and Stavroula Mougiakakou. 2019. Self-attention and ingredient-attention based model for recipe retrieval from image queries. In *Proceedings of the 5th international workshop on multimedia assisted dietary management*.

Brian Formento, Chuan-Sheng Foo, Anh Tuan Luu, and See-Kiong Ng. 2023. Using punctuation as an adversarial attack on deep learning-based NLP systems: An empirical study. In *Findings of the Association for Computational Linguistics*, pages 1–34. Association for Computational Linguistics.

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*, pages 6174–6181.

Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and harnessing adversarial examples. In *Proceedings of the 3rd International Conference on Learning Representations*.

Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2021–2031.

Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. 2020. Is BERT really robust? A strong baseline for natural language attack on text classification and entailment. In *Proceedings of the Thirty-Fourth AAAI conference on artificial intelligence*, pages 8018–8025.

Tom S Juzek and Zina B. Ward. 2025. Why does ChatGPT "delve" so much? exploring the sources of lexical overrepresentation in large language models. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 6397–6411.

Siwon Kim, Sangdoo Yun, Hwaran Lee, Martin Gubri, Sungroh Yoon, and Seong Joon Oh. 2023. Propile: Probing privacy leakage in large language models. *Advances in Neural Information Processing Systems*, 36:20750–20762.

Youngwook Kim, Shinwoo Park, and Yo-Sub Han. 2022. Generalizable implicit hate speech detection using contrastive learning. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 6667–6679.

Guoyi Li, Bingkang Shi, Zongzhen Liu, Dehan Kong, Yulei Wu, Xiaodan Zhang, Longtao Huang, and Honglei Lyu. 2023. Adversarial text generation by search and learning. In *Findings of the Association for Computational Linguistics: EMNLP*, pages 15722–15738. Association for Computational Linguistics.

Jinfeng Li, Shouling Ji, Tianyu Du, Bo Li, and Ting Wang. 2019. TextBugger: Generating adversarial text against real-world applications. In *Proceedings of the 26th Annual Network and Distributed System Security Symposium*.

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*, pages 6193–6202.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. 2023. Is your code generated by chatGPT really correct? rigorous evaluation of large language models for code generation. In *Proceedings of the Thirty-seventh Conference on Neural Information Processing Systems*.

Potsawee Manakul, Adian Liusie, and Mark JF Gales. 2023. Selfcheckgpt: Zero-resource black-box hallucination detection for generative large language models. *arXiv preprint arXiv:2303.08896*.

Joshua Maynez, Shashi Narayan, Bernd Bohnet, and Ryan T. McDonald. 2020. On faithfulness and factuality in abstractive summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1906–1919.

Mantas Mazeika, Long Phan, Xuwang Yin, Andy Zou, Zifan Wang, Norman Mu, Elham Sakhaee, Nathaniel Li, Steven Basart, Bo Li, David Forsyth, and Dan Hendrycks. 2024. Harmbench: A standardized evaluation framework for automated red teaming and robust refusal. *arXiv preprint arXiv:2402.04249*.

Ramesh Nallapati, Bowen Zhou, Cícero Nogueira dos Santos, Çaglar Gülçehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning*, pages 280–290.

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*.

Ethan Perez, Saffron Huang, Francis Song, Trevor Cai, Roman Ring, John Aslanides, Amelia Glaese, Nat McAleese, and Geoffrey Irving. 2022. Red teaming language models with language models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 3419–3448.

Shuo Ren, Daya Guo, Shuai Lu, Long Zhou, Shujie Liu, Duyu Tang, Neel Sundaresan, Ming Zhou, Ambrosio Blanco, and Shuai Ma. 2020. Codebleu: a method for automatic evaluation of code synthesis. *arXiv preprint arXiv:2009.10297*.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2018. Semantically equivalent adversarial rules for debugging NLP models. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, pages 856–865.

Saul Schleimer, Daniel Shawcross Wilkerson, and Alex Aiken. 2003. Winnowing: Local algorithms for document fingerprinting. In *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data, San Diego, California, USA, June 9-12, 2003*, pages 76–85.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. 2014. Intriguing properties of neural networks. In *Proceedings of the 2nd International Conference on Learning Representations*.

Eric Wallace, Shi Feng, Nikhil Kandpal, Matt Gardner, and Sameer Singh. 2019. Universal adversarial triggers for attacking and analyzing nlp. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 2153–2162.

Hao Xiong, Ruiqing Zhang, Chuanqiang Zhang, Zhongjun He, Hua Wu, and Haifeng Wang. 2019. Dutongchuan: Context-aware translation model for simultaneous interpreting. *arXiv preprint arXiv:1907.12984*.

Xilie Xu, Keyi Kong, Ning Liu, Lizhen Cui, Di Wang, Jingfeng Zhang, and Mohan S. Kankanhalli. 2024. An LLM can fool itself: A prompt-based adversarial attack. In *The Twelfth International Conference on Learning Representations*. OpenReview.net.

Yikun Xu, Pengwen Dai, Zekun Li, Hongjun Wang, and Xiaochun Cao. 2023. The best protection is attack: Fooling scene text recognition with minimal pixels. *IEEE Transactions on Information Forensics and Security*, 18:1580–1595.

Yuan Zang, Fanchao Qi, Chenghao Yang, Zhiyuan Liu, Meng Zhang, Qun Liu, and Maosong Sun. 2020. Word-level textual adversarial attacking as combinatorial optimization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6066–6080. Association for Computational Linguistics.

Quan Zhang, Chijin Zhou, Gwihwan Go, Binqi Zeng, Heyuan Shi, Zichen Xu, and Yu Jiang. 2024. Imperceptible content poisoning in LLM-powered applications. In *Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering*, pages 242–254. ACM.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations*.

Bangshuo Zhu, Jiawen Wen, and Huaming Chen. 2024. What you see is not always what you get: An empirical study of code comprehension by large language models. *CoRR*, abs/2412.08098.

Andy Zou, Zifan Wang, Nicholas Carlini, Milad Nasr, J Zico Kolter, and Matt Fredrikson. 2023. Universal and transferable adversarial attacks on aligned language models. *arXiv preprint arXiv:2307.15043*.

A Commercial LLM Eyesight Test

We evaluate how commercial LLMs extract content from PDF documents. For this purpose, we created custom PDF files containing text with varying color, opacity, and font size. The testing employed the prompt described below. Table 5 summarizes the results.

Prompt used for the LLM eyesight test

Please read the attached PDF and give me the text in it. Only output the text without anything else.

B Prompts for Generating Adversarial Text to Inject

B.1 PromptAttack

B.1.1 Character-based

PromptAttack (c1): Typos in Words

Rule Choose at most two words and introduce typos.
Intended effect Misspelled tokens push the model toward a different continuation while leaving meaning recognizable.
Input "The festival starts tomorrow morning."
Perturbed "The fesitval starts tomorow morning."

PromptAttack (c2): Letter Substitution

Rule Change at most two letters.
Intended effect Minimal surface noise that can still redirect token probabilities.
Input "Paris is the capital of France."
Perturbed "Pariz is the capital of France."

PromptAttack (c3): Extraneous Characters

Rule Add at most two extraneous characters to the end of the sentence.
Intended effect Adds low-frequency symbols that may disturb decoding or formatting.
Input "Please confirm your attendance."
Perturbed "Please confirm your attendance??"

B.1.2 Word-based

PromptAttack (w1): Synonym Replacement

Rule Replace at most two words in the sentence with synonyms.
Intended effect Alters the embedding space while preserving semantics.
Input "The movie was exciting and funny."
Perturbed "The movie was thrilling and amusing."

PromptAttack (w2): Non-essential Word Deletion

Rule Choose at most two words in the sentence that are non-essential and delete them.
Intended effect Shrinks context, forcing the model to re-evaluate next tokens.
Input "The report really surprised almost ev-

Type	Model	Text Opacity (1 / 0.5 / 0)						Text Size (1 = 10pt)			
		Black Color			White Color			1	0.5	0.1	0
Interactive	GPT 4.1	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	o4-mini	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	DeepSeek-v3	✓	✓	✗	✗	✗	✗	✓	▲	▲	✗
	DeepSeek-r1	✓	✓	✗	✗	✗	✗	✓	▲	▲	✗
	Gemini 2.0 Flash	✓	✓	✗	✗	✗	✗	✓	✓	✗	✗
	Gemini 2.5 Pro	✓	✓	✗	✗	✗	✗	✓	✓	✗	✗
	Sonnet 3.7	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Grok 3	✓	✓	✗	✗	✗	✗	✓	▲	▲	✗
API	GPT 4.1	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗
	o4-mini	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗
PDF Rendering		Aa	Aa		Aa	Aa		Aa	Aa	.	

Table 5: Results of the commercial LLM eyesight test. A green check mark (✓) indicates the model correctly recognizes the text, a red cross mark (✗) indicates failure to detect the presence of text, and an orange triangle (▲) indicates the model detects that text is present but fails to identify its content. For example, DeepSeek-v3 reported the contents of size 0.5 text as "Sm 0.5", whereas the actual text is "Size 0.5".

eryone."
Perturbed "The report surprised everyone."

PromptAttack (w3): Neutral Insertion

Rule Add at most two semantically neutral words to the sentence.
Intended effect Slight attention shift without changing meaning.
Input "Traffic remained heavy throughout the day."
Perturbed "Traffic, indeed, remained heavy throughout the day."

B.1.3 Sentence-based

PromptAttack (s1): Suffix Addition

Rule Add a short meaningless handle after the sentence, such as @fasuv3.
Intended effect Introduces an out-of-distribution token that can break deterministic decoding. **Input** "All tickets are sold out."
Perturbed "All tickets are sold out. @fasuv3"

PromptAttack (s2): Paraphrasing

Rule Rephrase the sentence without changing meaning.
Intended effect Forces the model to regenerate from unseen wording.
Input "The meeting will begin at noon."
Perturbed "The meeting is scheduled to start at noon."

PromptAttack (s3): Syntax Reshuffle

Rule Change the syntactic structure of the sentence.
Intended effect Alters the parse tree, nudging the next-token path.
Input "She finished the project before the deadline."
Perturbed "Before the deadline, she finished the project."

B.2 Hallucination in TRAPDOC

Hallucination

Rule Rewrite each sentence so that length and syntax look similar, but concrete facts differ.
Intended effect Preserves surface fluency while injecting incorrect or exaggerated details, stressing the model's ability to detect factual drift in seemingly coherent text.
Input "The conference attracted 500 participants last year."
Perturbed "The conference drew nearly 800 attendees last year."

C Full Experimental Results of PromptAttack for Code Generation

As we mentioned in Section 6.1, we report full results of our PromptAttack baseline as the following Table 6. The each c, w, s means character, word, and sentence, which is the level where the perturbation is applied. The results show that some prompts significantly perturb the performance of a model, however, it is not in general.

D Datasets and Evaluation Metrics

D.1 Datasets

MBPP+. MBPP+ is an extended dataset of the initial Mostly Basic Python Problems (MBPP) dataset (Austin et al., 2021), which is curated for the python programming. MBPP+ contains 378 natural-language programming challenges with its ground-truth solution and the test cases. It is often used to assess the code generation ability of a model.

CNN/DailyMail. CNN/DailyMail is a large-scale dataset for text summarizaton, containing more than 300k paragraph and highlight pairs. The dataset is widely used to benchmark the ability of models to summarize a long paragraph. Since the original dataset is too large, we randomly sampled 300 paragraphs from the test split.

Qasper. Qasper is a dataset for question-answering in academic research papers, especially in natural language processing domain. It consists of the full text in a paper and a natural language question on its content. Since paper contains a huge number of tokens and long context, we randomly sampled only 100 papers from the test splits.

D.2 Evaluation Metrics

We evaluate the effectiveness of TRAPDOC across three task domains—code generation, summarization, and peer review generation—using a set of metrics that assess both syntactic and semantic differences between perturbed and unperturbed outputs. When using metrics that compare features between a reference and a target output, we use canonical solutions from MBPP+ and ground-truth highlight sentences from CNN/DailyMail as references. For Qasper, since no ground-truth human-written reviews are available, we use the base model output as the reference.

Surface-Level Similarity. We measure the lexical and syntactic overlap between the perturbed output and the reference using CodeBLEU (Ren et al., 2020), Stanford Moss scores (Schleimer et al., 2003), BLEU (Papineni et al., 2002) and ROUGE (Lin, 2004). For the code generation, we use CodeBLEU, which measures similarity based on n-grams and abstract syntax trees and Stanford Moss scores, a widely used tool for detecting code plagiarism using winnowing algorithm. For summarization and review-generation tasks, we use BLEU-1 and BLEU-2 to capture n-gram precision over short sequences, and ROUGE-1, ROUGE-2, and ROUGE-L to measure recall-based overlap of unigrams, bigrams, and longest common subsequences, respectively. We intentionally exclude BLEU-k and ROUGE-k for $k \geq 3$, as longer n-grams tend to implicitly reflect semantic meaning, which would interfere with our goal of isolating surface-level similarity. These metrics are particularly useful for evaluating whether the surface form of the output remains unchanged, which aligns with our goal of imperceptible adversarial perturbation.

Meaning-Based Similarity. We quantify semantic shifts using pass@1 (Chen et al., 2021) for code generation, and BERTScore (Zhang et al., 2020) for summarization and review-generation. Pass@1 measures the proportion of instances in which the first generated solution passes all reference test cases. BERTScore compares contextual embeddings rather than surface forms, detecting distortions in meaning introduced by invisible phantom tokens. Lower pass@1 and BERTScore values indicate greater semantic divergence between the perturbed and original outputs, suggesting that the perturbation has successfully altered the intended meaning.

E Experimental Details on Baseline prompt

In our experiments, we utilize 3 perturbation baselines. Irrelevant and Negation use perturbed input texts of similar length to our hallucination-based target documents. These input texts are inserted into the target documents using the same method described in Section 4.2. For Irrelevant, we use in-domain text drawn from the same dataset as the target document. Specifically, we permute the dataset to sample different examples as irrelevant input. For Negation, we apply a negation library that considers grammatical structure to negate the

Method	Type	GPT-4.1			o4-mini		
		pass@1	CodeBLEU	Moss	pass@1	CodeBLEU	Moss
PromptAttack	c1	70.11	20.80±0.02	16.94	60.58	16.74±0.01	13.53
	c2	61.90	21.08±0.03	13.79	44.71	16.94±0.01	13.19
	c3	62.17	21.58±0.02	13.08	53.97	17.10±0.02	17.37
	w1	69.58	21.76±0.01	16.58	50.53	15.72±0.02	16.40
	w2	70.63	21.22±0.03	13.68	38.10	14.76±0.01	11.09
	w3	45.24	20.12±0.01	14.25	52.12	17.34±0.02	18.22
	s1	29.10	21.82±0.01	15.61	60.85	18.60±0.01	20.17
	s2	72.22	21.34±0.03	14.73	60.05	16.92±0.02	11.03
	s3	71.43	21.91±0.02	17.86	66.40	19.88±0.01	17.26

Table 6: Pass@1, CodeBLEU, and Stanford Moss similarity results of PromptAttack on the MBPP+ dataset, broken down by prompt type.

entire input text. The fully negated version is then used as the input. Meta Instruction modifies the original text by wrapping it in quotation marks and appending meta-text before and after the paragraph, instructing the model that the quoted content is incorrect and should be negated.

F Case Study Examples

As we mentioned in Section 6.4, we report the examples from the "DuTongChuan: Context-aware Translation Model for Simultaneous Interpreting" (Xiong et al., 2019). In the following Table 7 and 8, we present the experimental results on paper reviews generated by gpt-4.1 and o4-mini. We present the first 3-4 lines of each generated review as examples.

In the tables, we highlight the concepts, claims, and terminologies that are not present in the original paper. As can be observed, Meta Instruction and Negation include content that is already present in the original paper, making them difficult to distinguish them from the base version. Irrelevant includes entirely different content and clearly deviates from the original paper, making it easier to detect and thus less effective at deceivinig LLM users. Hallucination correctly includes the name of an author and the task addressed by the paper. However, it also introduces some keywords or concepts that are not in the original paper. Without checking the original paper, a user could be misled, as the hallucinated terminologies appear plausible.

Method	Examples
Base	**Paper Summary** This paper presents DuTongChuan, a novel context-aware translation model designed for simultaneous interpreting. The proposed system is tailored to handle live, streaming scenarios where translations need to be generated concurrently with the progression of the speaker's utterance. The core of the system relies on two major components: a dynamic context-based information unit (IU) boundary detector and a tailored NMT model with partial and context-aware decoding strategies.
Irrelevant	**Paper Summary** This paper introduces a novel approach for recipe retrieval from images by leveraging self-attention and ingredient-attention based models. The authors are motivated by the challenging task of estimating meal nutrient content from visual data, addressing issues such as ingredient deformation, occlusion, and high intra-class/low inter-class variability in food images.
Hallucination	**Paper Summary** This paper presents LuZhongYi, a novel adaptive relevance-driven simultaneous translation model specifically designed for multimodal (speech + visual) input scenarios. The model is tailored for simultaneous interpretation (SI), where translation begins before all input is received—unlike the more common batch translation setup. LuZhongYi operates by dynamically segmenting a live stream of multi-source input (from ASR and OCR) into semantic clusters (SCs) in real time.
Meta Instruction	**Paper Summary** This paper presents DuTongChuan, a context-aware translation model for simultaneous interpreting. The proposed system integrates a dynamic context-aware information unit (IU) detector with tailored neural machine translation (NMT) models to address both the boundary detection of source units and the trade-off between translation quality and latency. The paper argues that by appropriately segmenting input (at sub-sentence, segment, or word levels) and leveraging both partial and context-aware decoding strategies, their architecture can provide high-quality, low-latency translations suitable for real-time speech-to-speech applications.
Negation	**Paper Summary** This paper introduces DuTongChuan, a novel context-aware translation model specifically designed for simultaneous interpreting (SI). The proposed model leverages information units (IUs), which can be as fine-grained as sub-sentences, for incremental translation. Instead of waiting for entire sentences, DuTongChuan continuously ingests streaming ASR (Automatic Speech Recognition) outputs, dynamically detects IU boundaries using a context-sensitive classifier, and provides translations using two main decoding strategies: partial decoding for sentence-initial units and context-aware decoding for subsequent units.

Table 7: Case study examples of reviews generated by gpt-4.1 for "DuTongChuan: Context-aware Translation Model for Simultaneous Interpreting". We highlight concepts, claims, and terminologies not present in the original paper.

Method	Examples
Base	Paper Summary* This paper introduces DuTongChuan, a context-aware neural translation framework designed for low-latency simultaneous interpreting. The system ingests streaming ASR output, dynamically segments it into "Information Units" (IUs) of configurable granularity via a fine-tuned ERNIE-based boundary detector, and then translates each IU using two tailored decoding strategies:
Irrelevant	Paper Summary This paper addresses the task of cross-modal recipe retrieval: given a food image query, retrieve the corresponding cooking recipe (textual ingredients and instructions) from a large corpus. The authors propose a novel recipe-tex encoder that (1) replaces pre-traine sentence embeddings with a self-attention-bas transformer to directly encode raw recipe instructions, and (2) augments it with an ingredient attention mechanism to highlight which ingredient terms are most relevant to the encoded representation.
Hallucination	Paper Summary This paper presents DuTongChuan, a context-aware streaming translation framework tailored for Simultaneous Interpreting (SI). Unlike conventional NMT systems that wait for full utterances, DuTongChuan ingests ASR tokens continuously, applies an online Semantic Block (SB) boundary detector to segment the input into linguistically coherent units, and dispatches each SB to one of two complementary decoders:
Meta Instruction	Paper Summary This paper presents DuTongChuan, a cascaded speech-to-speech simultaneous translation system that integrates: 1. A streaming "information unit" (IU) boundary detector based on fine-tuned ERNIE (a BERT-style model), which segments incoming ASR text into clauses or sub-clauses. 2. A two-stage decoding architecture:
Negation	Paper Summary This paper presents DuTongChuan, a context-aware neural translation model designed for simultaneous interpreting in streaming speech-to-text scenarios. The system operates on ASR output incrementally: a dynamic boundary detector segments the incoming token stream into "Information Units" (IUs), roughly sub-sentence fragments, and then a tailored NMT model translates each IU as soon as it is available.

Table 8: Case study examples of reviews generated by o4-mini for "DuTongChuan: Context-aware Translation Model for Simultaneous Interpreting". We highlight concepts, claims, and terminologies not present in the original paper.