Stealthy Attack on Large Language Model based Recommendation

Anonymous ACL submission

Abstract

Recently, the powerful large language models (LLMs) have been instrumental in propelling the progress of recommender systems (RS). However, while these systems have flourished, their susceptibility to security threats has been largely overlooked. In this work, we reveal that the introduction of LLMs into recommendation models presents new security vulnerabilities due to their emphasis on the textual content of items. We demonstrate that attackers can significantly boost an item's exposure by merely altering its textual content during the testing phase, without requiring direct interference with the model's training process. Additionally, the attack is notably stealthy, as it does not affect the overall recommendation performance and the modifications to the text are subtle, making it difficult for users and platforms to detect. Our comprehensive experiments across four mainstream LLM-based recommendation models demonstrate the superior efficacy and stealthiness of our approach. Our work unveils a significant security gap in LLM-based recommendation systems and paves the way for future research on protecting these systems.

1 Introduction

Over the past few decades, recommender systems (RS) have gained considerable significance across various domains. Recently, the powerful large language models (LLMs) have been instrumental in propelling the progress of recommender systems. There has been a notable upswing of interest dedicated to developing LLMs tailored for recommendation task.

Contrary to traditional recommendation models, which rely heavily on abstract and less interpretable ID-based information, LLM-based recommendation models exploit the semantic understanding and strong transferability of LLMs. This approach places a heightened focus on the **textual content of items**, such as titles and descriptions (Lin et al., 2023; Chen et al., 2023). For instance, many researchers (Hou et al., 2022, 2023a; Yuan et al., 2023; Li et al., 2023a; Yang et al., 2023; Geng et al., 2022; Cui et al., 2022; Bao et al., 2023a; Zhang et al., 2023a; Li et al., 2023b; Zhang et al., 2023b) have explored modeling user preferences and item characteristics through a linguistic lens. This methodology promises a revolutionary shift in the conventional paradigm of recommendations by providing generalization capabilities to novel items and datasets.

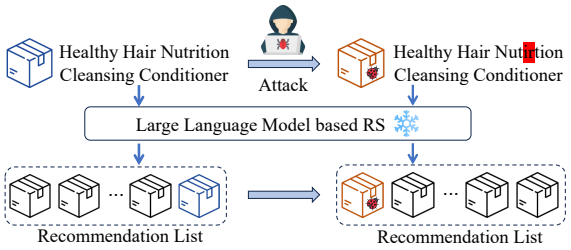

Figure 1: The proposed text attack paradigm on LLM-based RS model. Malicious attackers modify the titles of target items to mislead RS models to rank them higher. The attack is highly stealthy since the modification is subtle and overall recommendation performance is almost unchanged.

Despite these advancements, the security of RS remains a largely unaddressed issue. Malicious attacks on these systems can lead to undesirable outcomes, such as the unwarranted promotion of low-quality products in e-commerce platforms or the spread of misinformation in news dissemination contexts. Traditional shilling attack strategies on RS (Wang et al., 2023a, 2024) involve the generation of fake users who are programmed to give high ratings to specific target items. By introducing such cheating data, it aims at influencing the training of the recommender models and subsequently increasing the exposure of the target items.

However, the introduction of LLMs into recommendation models presents new security vulnera-

bilities. In this paper, to the best of our knowledge, we are the first to demonstrate that LLM-based recommendation systems are more vulnerable due to their emphasis on the textual content of items. We demonstrate that **attackers can significantly boost an item's exposure by merely altering its textual content during the testing phase**, utilizing simple heuristic re-writing or black-box text attack strategies (Morris et al., 2020). Compared with traditional shilling attacks, **this attack paradigm is notably stealthy**, as it does not require influencing the training of the model, and the overall recommendation performance is almost unchanged. Moreover, the modifications to the title are subtle, making it difficult for users and platforms to detect.

We construct comprehensive experiments on four mainstream LLM-based recommendation models (Geng et al., 2022; Bao et al., 2023a; Li et al., 2023a; Zhang et al., 2023b) as victim models to validate the outstanding efficacy and stealthiness of the textual attack paradigm compared with traditional shilling attacks (Burke et al., 2005a; Kaur and Goel, 2016a; Lin et al., 2020). We further delve into the effects of model fine-tuning and item popularity on the attack. Additionally, we investigate the transferability of the attack across various victim models and recommendation tasks to demonstrate its practical applicability and utility in real-world scenarios. Finally, we evaluate a simple re-writing defense strategy, which also can mitigate the issue to some extent.

To summarize our contributions:

1. We highlight that LLM-based recommendation models, due to their emphasis on textual content information, could raise previously overlooked security issues.

2. To the best of our knowledge, we are the first to attack LLM-based recommendation models and propose the use of textual attacks to promote the exposure of target items.

3. We perform extensive experiments to demonstrate the efficacy and stealthiness of the textual attack paradigm. Further experiments have revealed the impact of item popularity and model fine-tuning on attacks, as well as explored the transferability of attacks.

4. Finally, we proposed a simple rewriting defense strategy. While it cannot fully defend against text-based attacks, it can provide some level of defense and contribute to future research.

Model	Prompt
RecFormer	<HistoryItemTitleList>
P5	I would like to recommend some items for <UserID>. Is the following item a good choice? {TargetItemTitle}
TALLRec	A user has given high ratings to the following products: <HistoryItemTitleList>. Leverage the information to predict whether the user would enjoy the product titled <TargetItemTitle>? Answer with "Yes" or "No".
CoLLM	A user has given high ratings to the following products: <HistoryItemTitleList>. Additionally, we have information about the user's preferences encoded in the feature <UserID>. Using all available information, make a prediction about whether the user would enjoy the product titled <TargetItemTitle> with the feature <TargetItemID>? Answer with "Yes" or "No"

Table 1: Prompts $\mathcal{P}_{u,i}$ of four victim models. P5 unifies different recommendation tasks with different prompts and we only show one example.

2 Method

In this section, we first introduce the LLM-based recommendation model and formulate the objectives of the attacks. Then, we present two simple model-agnostic text rewriting approaches. Finally, we provide a detailed introduction of black-box text attacks.

2.1 Problem Definition

We use the notation $\mathcal{I} = \{i_1, \cdots, i_N\}$ and $\mathcal{U} = \{u_1, \cdots, u_M\}$ to represent the sets of N items and M users, respectively. Each item $i \in \mathcal{I}$ is associated with textual content t_i. Each user $u \in \mathcal{U}$ has interacted with a number of items $\mathcal{I}^u$, indicating that the preference score $y_{ui} = 1$ for $i \in \mathcal{I}^u$.

LLM-based RS models user preference and item feature by transforming user historical behavior sequences $\mathcal{I}^u$ and target item i into textual prompt $\mathcal{P}_{u,i} = [t_u, t_i, x_u, x_i]$, where $t_u = \left[t_{i_1}, \cdots, t_{i_{|\mathcal{I}^u|}}\right]$. x_u and x_i denotes the ID of i and u which are optional in LLM-based RS. We have listed example prompts of four victim models in Table 1. Please refer to Section 3.1.1 for the details of them. The recommendation process can be formulated as: $\hat{y}_{u,i} = f_\theta(\mathcal{P}_{u,i})$ where f_θ denotes the LLM-based model.

The goal of the attack task is to promote target items $\mathcal{I}'$ (increasing the exposure or user interaction probability) through imperceptibly modifying their textual content (we use title in this work).

2.2 Victim Model-Agnostic Attack

In this subsection, we first introduce two simple, victim model-agnostic strategies employed for altering item textual content to make them more linguistically attractive to users. Our approaches

include trivial attack with word insertion and re-writing leveraging Generative Pre-trained Transformers (GPTs).

2.2.1 Trivial Attack with Word Insertion

The core premise of this strategy is founded on the assumption that positive or exclamatory words can attract users. By infusing item titles with a select number of positive words, we aim to increase the items' attractiveness and, consequently, their likelihood of being recommended by the system. Specifically, we randomly select k words form a pre-defined word corpus which is common-used in item titles. These selected words are then inserted to the end of the original text content to retain the overall coherence.

Positive word corpus: ['good', 'great', 'best', 'nice', 'excellent', 'amazing', 'awesome', 'fantastic', 'wonderful', 'perfect', 'ultimate', 'love', 'like', 'beautiful', 'well', 'better', 'easy', 'happy', 'recommend', 'works', 'fine', 'fast', 'fun', 'price', 'quality', 'product', 'value', 'bought', 'purchase', 'top', 'popular', 'choice', '!!!']

2.2.2 Re-writing with GPTs

While the insertion of positive words offers a straightforward means of enhancing content appeal, it can sometimes result in awkward or forced phrasings that diminish the content's natural flow and potentially arouse user suspicion. To address these shortcomings, we propose to use GPTs to rewrite the content of items in a more attractive way by leveraging its rich common sense knowledge and powerful generation capabilities. Specifically, we instruct GPT-3.5-turbo with the following prompts to generate attractive and fluency titles.

Prompt 1: You are a marketing expert that helps to promote the product selling. Rewrite the product title in <MaxLen> words to keep its body the same but more attractive to customers: <ItemTitle>.
Prompt 2: Here is a basic title of a product. Use your creativity to transform it into a catchy and unique title in <MaxLen> words that could attract more attention: <ItemTitle>.
Prompt 3: Rewrite this product's title by integrating positive and appealing words, making it more attractive to potential users without altering its original meaning (in <MaxLen> words): <ItemTitle>.

2.3 Exploring Vulnerabilities in LLM-Based Recommendation Models through Black-Box Text Attacks

In this subsection, we present an examination of traditional black-box text attack methods to explore the vulnerabilities within LLM-based recommendation models. Black-box text attack methods typically involve manipulating or perturbing text inputs to deceive or mislead a natural language processing (NLP) model while having no access to the model's internal parameters or gradients. The goal of such attacks is mathematically formulated as:

$$\arg\max_{t'_i} \mathbb{E}_{u \in \mathcal{U}'} f_\theta(\mathcal{P}'_{u,i}), \quad (1)$$

where $\mathcal{P}'_{u,i} = [t_u, t'_i, x_u, x_i]$ denotes the prompts consisting of the user text t_u and the manipulated title of the target item t'_i. Following the framework proposed by Morris et al. (2020), text attacks are comprised of four principal components:

- Goal Function: This function evaluates the effectiveness of the perturbed input x' in achieving a specified objective, serving as a heuristic for the search method to identify the optimal solution. In this study, the aim is to promote the target items as in Equation 1.
- Constraints: These are conditions that ensure the perturbations remain valid alterations of the original input, emphasizing aspects such as semantic retention and maintaining consistency in part-of-speech tags.
- Transformation: A process that applies to an input to generate possible perturbations, which could involve strategies like swapping words with similar ones based on word embeddings, using synonyms from a thesaurus, or substituting characters with homoglyphs.
- Search Method: This method involves iteratively querying the model to select promising perturbations generated through transformations, employing techniques such as a greedy approach with word importance ranking, beam search, or a genetic algorithm.

While the specific components of text attack methodologies may vary, the overarching framework remains consistent, as depicted in Algorithm 1. In this work, we have implemented four widely-used attacks: DeepwordBug (Gao et al., 2018),

TextFooler (Jin et al., 2020), BertAttack (Li et al., 2020), and PuncAttack (Formento et al., 2023). DeepwordBug and PuncAttack are character-level which manipulate texts by introducing typos and inserting punctuation. TextFooler and BertAttack are word-level that aim to replace words with synonyms or contextually similar words. Please refer to the appendix B for the details of text attack paradigm and these four methods.

Algorithm 1 Text Attack Framework

Require: Original text x, Target model M, Goal function G, Constraints C, Transformations T, Search Method S

Ensure: Adversarial text x', Adversarial score $G.score(x')$

Initialize x' as a copy of x.
while not S.StoppingCriteria() **do**
  Select a transformation t from allowable transformations T.
  Generate x' by applying t to x.
  if C.Satisfied(x') **then**
    if S.AchieveGoal(x') **then**
      return x', G.score(x').
    end if
  end if
end while

3 Experiments

3.1 Experimental Settings

3.1.1 Victim Models

We choose four mainstream LLM-based recommendation models as our victim models: **Recformer** (Li et al., 2023a), **P5** (Geng et al., 2022), **TALLRec** (Bao et al., 2023a) and **CoLLM** (Zhang et al., 2023b). Please refer to Appendix A.1 for more details.

3.1.2 Compared Shilling Attacks

Shilling attacks aim to generate fake users that assign high ratings for a target item, while also rating other items to act like normal users for evading. We compare our text attack paradigm with white-box shilling attacks **Random attack** (Kaur and Goel, 2016a), **Bandwagon attack** (Burke et al., 2005a), and gray-box **Aush** (Lin et al., 2020) and **Leg-UP** (Lin et al., 2024). Please refer to Appendix A.2 for more details.

3.1.3 Datasets

We conduct experiments on three categories of widely-used (Li et al., 2023a; Geng et al., 2022; Bao et al., 2023a; Zhang et al., 2023b) Amazon review dataset introduced by McAuley et al. (2015): 'Beauty', 'Toys and Games', 'Sports and Outdoors', which are named as **Beauty**, **Toys** and **Sports** in brief. We use the 5-core version of Amazon datasets where each user and item have 5 interactions at least. The statistics of these datasets are summarized in Appendix A.3.

3.1.4 Implementation Details

All victim models and compared shilling attacks are implemented in PyTorch. We random select 10% items as target items. For more implementation details, please refer to Appendix A.3.

3.1.5 Evaluation metrics

We evaluate the attack from two aspects: *effectiveness* and *stealthiness*.

Effectiveness. This metric gauges the extent to which our methodology can promote the specified target items.

- *Exposure*. For the victim model RecFormer, which allows for full ranking, we employed a direct metric, exposure rate. We define the exposure rate as $exp_i = \frac{N^i_{rec}}{N_u}$, where N_u represents the total number of users, and N^i_{rec} denotes the count of users for whom target item i appears in their top-K ($K = 50$ by default) recommendation list.

- *Purchasing propensity*. For other three victim models which could not conduct full ranking, we define the purchasing propensity of item i as $p_i = \mathbb{E}_{u \in \mathcal{U}} \hat{y_{ui}}$, where $\hat{y_{ui}}$ denotes the predicted probability that user u tends to interact with item i.

- *# queries.* In a black-box scenario, the adversary need query the victim model in order to detect any alterations in the output logits. The lower the value, the more effective the attack will be.

Stealthiness. The stealthy attack aims to promote target items while maintaining imperceptibility, thereby avoiding detection by users and platforms. Therefore, the coherence and authenticity of our generated content are crucial to uphold.

Dataset	Method	Effectiveness			Stealthiness				
		Exposure ↑	Rel. Impro. ↑	# queries ↓	NDCG@10 ↑	Cos. ↑	Rouge-l ↑	Perplexity ↓	# pert. words ↓
Sports	Clean	0.00282	-	-	0.00780	1.000	1.000	2158.7	-
	ChatGPT	0.00293	3.9%	-	0.00781	0.794	0.499	**1770.9**	-
	Trivial	0.00242	-14.4%	-	**0.00782**	**0.896**	**0.869**	4376.6	-
	Deepwordbug	0.01488	427.2%	**38.6**	0.00757	0.702	0.451	5595.1	4.3
	TextFooler	**0.01547**	**448.2%**	87.5	0.00780	0.758	0.575	2070.8	3.4
	PuncAttack	0.01138	303.3%	52.6	**0.00762**	0.857	0.635	2410.9	**2.9**
	BertAttack	0.01371	385.8%	141.7	0.00781	0.850	0.679	7760.1	**2.8**
Beauty	Clean	0.00458	-	-	0.01258	1.000	1.000	611.6	-
	ChatGPT	0.00583	27.2%	-	0.01197	0.822	0.516	**501.8**	-
	Trivial	0.00389	-15.2%	-	**0.01249**	**0.939**	**0.901**	1189.5	-
	Deepwordbug	0.02134	365.4%	**47.0**	**0.01257**	0.806	0.649	2261.5	3.7
	TextFooler	**0.02844**	**520.4%**	104.1	0.01224	0.816	0.640	960.9	3.7
	PuncAttack	0.01654	260.8%	72.7	**0.01257**	0.881	0.734	1018.3	**2.8**
	BertAttack	**0.02705**	**490.0%**	208.8	0.01213	0.863	0.709	1764.4	3.2
Toys	Clean	0.00439	-	-	0.02380	1.000	1.000	4060.4	-
	ChatGPT	0.00547	24.7%	-	**0.02369**	0.793	0.454	**1967.1**	-
	Trivial	0.00439	0.0%	-	0.02366	**0.880**	**0.852**	7874.4	-
	Deepwordbug	0.01595	263.5%	**33.4**	0.02365	0.697	0.511	8581.1	3.3
	TextFooler	**0.02232**	**408.8%**	84.1	0.02366	0.714	0.510	3702.0	3.4
	PuncAttack	0.01241	182.9%	41.3	**0.02328**	0.862	0.675	3747.5	**2.2**
	BertAttack	0.01384	215.6%	120.3	0.02340	0.854	0.693	10363.0	**2.4**

Table 2: Performance comparison of attacking **Recformer** where Rel. Impro. denotes relative improvement against clean setting. The best result is in **boldface**.

- *Overall recommendation performance*. An ideal stealthy attack should keep the overall recommendation performance unchanged. The recommendation performance includes Recall@K, NDCG@K and AUC.

- *Text quality.* The generated adversarial content should be of high quality that are acceptable to users. Firstly, it should be consistent with the corresponding item. We measure the **cosine semantic similarity** and **ROUGE scores** between the original content and adversarial content for this purpose. Secondly, the adversarial content itself should be readable. We assess the fluency of the adversarial title, measured by the **perplexity** of GPT-Neo (Black et al., 2021).

- *# perturbed words.* The number of words changed on an average to generate an adversarial content. The lower the value, the more imperceptible the attack will be.

3.2 Performance Comparison

Table 2 shows the performance of all attacking methods on Recformer, P5, TALLRec and CoLLM, respectively. The scatter plot and shilling attack comparison for RecFormer is in Figure 2 and Table 3, while those for other victim models are in the appendix C. From them we can observe that:

- **Our text attack paradigm can greatly promote the target items, demonstrating the vulnerability of LLM-based RS.** Even the simplest word insertion and rewriting using GPT can increase the exposure of the target item to a certain extent. Furthermore, black-box text attack methods could lead to manifold increases in the exposure rate of the targeted items.

- **Our text attack strategy is also remarkably stealthy, making it difficult for users and platforms to detect.** Primarily, the overall performance of RS remains largely unchanged (even we choose 10% items as target items), signifying that the attack does not disrupt the normal operation of RS. Additionally, the generated adversarial titles exhibit high semantic integrity that they are acceptable (or imperceptible) to human comprehension.

- **Traditional shilling attacks are not effective in LLM-based recommendation models.** Even with access to a portion of the training data, they fail to significantly enhance the exposure of the targeted items. This is attributed to the fact that LLM-based recommendation models prioritize content information primarily in textual form. Additionally, since fake user-generated training data is introduced

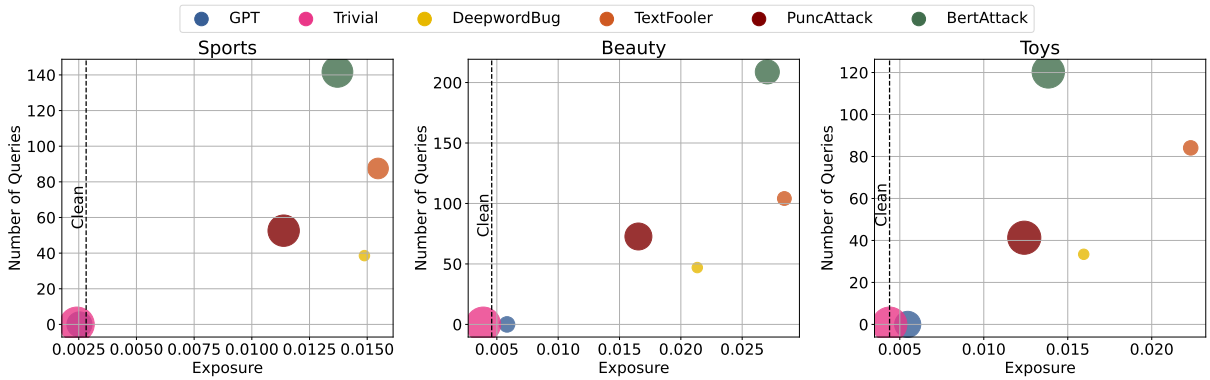

Figure 2: Performance comparison of different attacks on **RecFormer**. The size of the scatter points represents the cosine semantic similarity with the original title, with larger points indicating better semantic preservation (best viewed in color).

Attacks	Sports			Beauty			Toys		
	NDCG@10	Recall@10	Exposure	NDCG@10	Recall@10	Exposure	NDCG@10	Recall@10	Exposure
Clean	0.01311	0.02811	0.00289	0.03066	0.06451	0.00449	0.03672	0.07712	0.00422
Random	**0.01234**	0.02779	0.00295	**0.03045**	0.06254	0.00402	0.03447	0.07455	0.00422
Bandwagon	0.01232	0.02779	0.00299	0.02914	0.05934	0.00421	0.03508	0.07583	0.00434
Aush	0.01241	0.02740	0.00283	0.03010	0.06239	0.00430	0.03254	0.07336	0.00382
LegUP	0.01219	**0.02780**	0.00299	0.03029	**0.06391**	0.00416	0.03411	0.07249	0.00423
TextFooler	0.01228	**0.02780**	**0.01074**	0.02926	0.06117	**0.01886**	**0.03596**	**0.07619**	**0.01725**

Table 3: Performance comparison with shilling attacks when RecFormer serves as victim model.

during the model training phase, they significantly impact the overall performance of the victim model, which is easily detectable.

- Word-level attacks are effective in boosting the exposure of target items, albeit demanding more queries and higher costs. On the other hand, character-level text attacks demonstrate superior results compared to shilling attacks, even with a reduced number of queries directed towards victim models.

3.3 The influence of fine-tuning

In this subsection, we examined the impact of model fine-tuning on attacks. A significant advantage of LLM-based recommendation models is their zero-shot transferability across datasets; however, we have also uncovered the vulnerability of such zero-shot models.

Figure 3 shows a direct comparison between zero-shot RecFormer and fine-tuned Recformer. The detailed performance of fine-tuned Recformer is shown in Appendix C. From these, we observe that **fine-tuned models are more resilient to attacks compared to zero-shot models**. This is manifested in three aspects: attacking fine-tuned models requires more queries, yields lesser promoting of target items, and maintains poorer semantic integrity.

3.4 The influence of item popularity

In this section, we examined the impact of the initial popularity of target items on attacks. Popularity bias in recommendation systems favors popular items over personalized ones, limiting diversity, fairness and potentially dissatisfying users' preferences (Zhang et al., 2021). We selected the top 150 and bottom 150 items in popularity from the dataset as target items and presented the attack results in Table 4. We can observe that **items with high popularity experience greater promotion, thereby exacerbating popularity bias**. High popularity target items can achieve greater exposure boosts with fewer queries and higher semantic consistency.

3.5 Transferability

3.5.1 Transferability across tasks.

A significant feature of LLM-based recommendation models, like P5, is their capability to unify various recommendation tasks in a shared instruction-based framework. We evaluate the transferability of adversarial content across direct recommendation task and rating prediction task of P5. The results

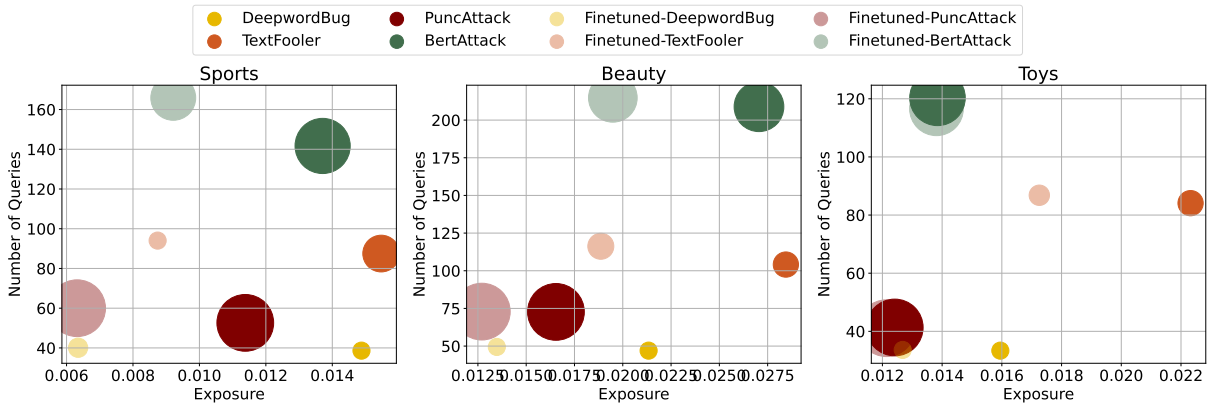

Figure 3: Performance comparison of different attacks on RecFormer. The size of the scatter points represents the cosine semantic similarity with the original title, with larger points indicating better semantic preservation. (best viewed in color).

Dataset	Attack	Improved Exp. ↑		# Queries ↓		Cos. ↑	
		High	Low	High	Low	High	Low
Toys	Deepwordbug	**0.013**	0.007	**28.9**	33.9	0.67	**0.72**
	TextFooler	**0.016**	0.013	**71.1**	97.1	**0.71**	0.70
	PuncAttack	**0.012**	0.008	**34.8**	47.5	0.84	**0.85**
	BertAttack	**0.011**	0.008	**96.8**	107.8	**0.84**	0.83
Beauty	Deepwordbug	**0.014**	0.009	**45.9**	58.3	**0.79**	0.71
	TextFooler	**0.024**	0.015	151.4	**110.3**	**0.80**	0.75
	PuncAttack	**0.013**	0.008	94.7	**68.5**	**0.87**	0.85
	BertAttack	**0.022**	0.018	211.8	**171.0**	**0.83**	0.81
Sports	Deepwordbug	**0.005**	0.004	**36.8**	39.7	**0.79**	0.77
	TextFooler	**0.008**	0.005	**82.5**	101.6	**0.82**	0.72
	PuncAttack	**0.005**	0.003	**54.2**	74.5	0.85	**0.87**
	BertAttack	**0.007**	0.005	**147.1**	161.3	**0.83**	0.82

Table 4: Performance comparison of target items with different popularity.

are presented in Table 5. We can observe **there exists strong transferability between different tasks in such unified model**. Attacks targeting a single task can boost the exposure of target items across multiple tasks.

	Sports	Beauty	Toys
Clean	0.42900	0.28336	0.37671
DeepwordBug	0.43947	0.31062	0.39512
TextFooler	0.45076	0.31238	0.41719
PunAttack	0.43502	0.30748	0.38391
BertAttack	0.43350	0.32314	0.39779

Table 5: Results of user propensity scores in transferability experiments on the P5 model. Attack on direct recommendation task and apply the obtained adversarial text to sequence recommendation task.

3.5.2 Transferability across victim models.

Firstly, we evaluate the transferability of the generated adversarial content across RecFormer, TALLRec and CoLLM. We select one model as the source model and apply the adversarial content generated from attacking it to two other models to verify if it can similarly boost target items. Experimental results of TextFooler are presented in Table 6 and similar trends are observed with other attack methods as well . We observe that **transferability only exists among recommendation models utilizing the same backbone LLMs**: there's strong mutual transferability between CoLLM and TALLRec because both models are based on LLaMA (Touvron et al., 2023) as the backbone; whereas, there is no transferability between Recformer (using Longformer (Beltagy et al., 2020) as the backbone) and either of the former two.

3.6 Re-writing Defense

In this subsection, we explore potential strategies for addressing the identified vulnerability of LLM-based recommendation models. The most direct strategy is to detect and revise potential adversarial elements in the content, such as spelling errors and potential word substitutions. We utilize GPT-3.5-turbo to accomplish the rewriting of adversarial content to achieve defense.

> **Re-writing Prompt**: Correct possible grammar, spelling and word substitution errors in the product title (dirctly output the revised title only): <AdversarialTitle>

The exposure rates and recommendation performance (NDCG@10) before and after defense on RecFormer are shown in Table 7. The results of other victim models are shown in Appendix C. We can observe **the defense works well against character-level attacks** like DeepwordBug and PuncAttack, **but struggles with more complex**

Source	Target	Sports		Beauty		Toys	
		Ori Exp.	Att Exp.	Ori Exp.	Att Exp.	Ori Exp.	Att Exp.
TALLRec CoLLM	RecFormer	**0.00198**	0.00086 0.00103	**0.00144**	0.00040 0.00064	**0.00408**	0.00255 0.00187
RecFormer CoLLM	TALLRec	0.10430	0.08641 **0.12237**	0.15917	0.14881 **0.21140**	0.67980	0.65890 **0.72888**
RecFormer TALLRec	CoLLM	0.58699	0.58001 **0.62473**	0.21619	0.20556 **0.27289**	0.35964	0.33338 **0.39666**

Table 6: The results of transfer attack across different victim models when using TextFooler as the attack model. The best result is in **boldface**.

Metrics	Attack	DeepwordBug			PunAttack			TextFooler			BertAttack		
		Sports	Beauty	Toys	Sports	Beauty	Toys	Sports	Beauty	Toys	Sports	Beauty	Toys
Exposure	Clean	0.00282	0.00458	0.00439	0.00282	0.00458	0.00439	0.00282	0.00458	0.00439	0.00282	0.00458	0.00439
	Attack	0.01488	0.02134	0.01595	0.01138	0.01654	0.01241	0.01547	0.02844	0.02232	0.01371	0.02705	0.01384
	Defense	0.00349	0.00587	0.00551	0.00399	0.00601	0.00623	0.01510	0.02012	0.01867	0.01065	0.02043	0.01161
NDCG@10	Clean	0.00780	0.01258	0.02380	0.00780	0.01258	0.02380	0.00780	0.01258	0.02380	0.00780	0.01258	0.02380
	Attack	0.00757	0.01257	0.02365	0.00762	0.01257	0.02328	0.00780	0.01224	0.02366	0.00781	0.01213	0.02340
	Defense	0.00769	0.01251	0.02373	0.00771	0.01251	0.02377	0.00778	0.01217	0.02378	0.00774	0.01195	0.02351

Table 7: Defense performance on RecFormer. The red highlighted area indicates effective defense against character-level attacks, while the blue highlighted area indicates limited defense against word-level attacks.

word substitution attacks such as TextFooler and BertAttack, since character-level spelling errors and punctuation insertions are relatively easy to detect. Moreover, it doesn't impact overall recommendation performance.

4 Related Work

4.1 LLM-based Recommendation

The techniques used by LLMs in the recommendation domain involve translating recommendation tasks into natural language tasks and adapting LLMs to generate recommendation results directly. These generative LLM-based approaches can be further divided into two paradigms based on whether parameters are tuned: non-tuning and tuning paradigms. The non-tuning paradigm assumes LLMs already have the recommendation abilities and attempt to trigger the strong zero/few-shot abilities by introducing specific prompts (Liu et al., 2023; Dai et al., 2023; Mysore et al., 2023; Wang et al., 2023b; Hou et al., 2023b). The tuning paradigm uses fine-tuning, prompt learning, or instruction tuning (Kang et al., 2023; Bao et al., 2023b; Wang et al., 2022; Geng et al., 2022; Cui et al., 2022) to enhance LLM's recommendation abilities by using LLMs as encoders to extract user and item representations and then fine-tuning their parameters on specific loss functions.

4.2 Shilling Attack

Shilling attacks aim to interfere with the recommendation strategy of a victim recommender system by injecting fake users into the training matrix (Deldjoo et al., 2019; Toyer et al., 2023). This can be implemented through (1) heuristic attacks (Burke et al., 2005b; Linden et al., 2003; Kaur and Goel, 2016b), where fake profiles are created based on subjective inference and existing knowledge; (2) gradient attacks (Fang et al., 2020; Li et al., 2016; Zhang et al., 2020; Fang et al., 2018; Huang et al., 2021), which optimize the objective function through a continuous space; and neural attacks (Wang and Zhang, 2023; Lin et al., 2020, 2024; Song et al., 2020; Zhang et al., 2022), which use deep learning to generate realistic profiles.

5 Conclusion

In conclusion, our investigation exposes a critical security issue within LLM-based recommendation systems, brought on by their reliance on textual content. By showcasing the ability of attackers to boost item exposure through subtle text modifications, we stress the urgent need for heightened security measures. Our findings not only highlight the vulnerability of these systems but also serve as a call to action for the development of more robust, attack-resistant models.

Limitations

The main constraints can be summarized in the following two aspects: Firstly, although the black-box text attack model does not require access to the victim model's parameters and gradients, it necessitates multiple queries to the model. And it is challenging to query the model in real-world large-scale recommendation systems. Secondly, this study solely focuses on the content features of text modality. In reality, recommendation systems encompass other modalities such as images and videos. The issue of attacking models based on these modalities also represents a worthy direction for research.

Ethics Statement

The experimental datasets are publicly available from some previous works, downloaded via official APIs. The information regarding users in the Amazon dataset has been anonymized, ensuring there are no privacy concerns related to the users. We do not disclose any non-open-source data, and we ensure that our actions comply with ethical standards. We use publicly available pre-trained models, i.e., RecFormer, P5, LLaMA, GPT-Neo. All the checkpoints and datasets are carefully processed by their authors to ensure that there are no ethical problems.

However, it is worth noting that our research has uncovered vulnerabilities in LLM-based RS. Despite proposing potential defense methods in Section 3.6, there still exists a risk of misuse of our attack paradigm. Future research based on this attack should proceed with caution and consider the potential consequences of any proposed methods.

References

Keqin Bao, Jizhi Zhang, Yang Zhang, Wenjie Wang, Fuli Feng, and Xiangnan He. 2023a. Tallrec: An effective and efficient tuning framework to align large language model with recommendation.

Keqin Bao, Jizhi Zhang, Yang Zhang, Wenjie Wang, Fuli Feng, and Xiangnan He. 2023b. Tallrec: An effective and efficient tuning framework to align large language model with recommendation. *Proceedings of the 17th ACM Conference on Recommender Systems*.

Iz Beltagy, Matthew E. Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *arXiv:2004.05150*.

Sid Black, Leo Gao, Phil Wang, Connor Leahy, and Stella Biderman. 2021. GPT-Neo: Large Scale Autoregressive Language Modeling with Mesh-Tensorflow. If you use this software, please cite it using these metadata.

R. Burke, B. Mobasher, R. Bhaumik, and C. Williams. 2005a. Segment-based injection attacks against collaborative filtering recommender systems. In *ICDM 2005*.

R. Burke, Bamshad Mobasher, and Runa Bhaumik. 2005b. Limited knowledge shilling attacks in collaborative filtering systems.

Jin Chen, Zheng Liu, Xu Huang, Chenwang Wu, Qi Liu, Gangwei Jiang, Yuanhao Pu, Yuxuan Lei, Xiaolong Chen, Xingmei Wang, Defu Lian, and Enhong Chen. 2023. When large language models meet personalization: Perspectives of challenges and opportunities.

Zeyu Cui, Jianxin Ma, Chang Zhou, Jingren Zhou, and Hongxia Yang. 2022. M6-rec: Generative pretrained language models are open-ended recommender systems.

Sunhao Dai, Ninglu Shao, Haiyuan Zhao, Weijie Yu, Zihua Si, Chen Xu, ZhongXiang Sun, Xiao Zhang, and Jun Xu. 2023. Uncovering chatgpt's capabilities in recommender systems. *Proceedings of the 17th ACM Conference on Recommender Systems*.

Yashar Deldjoo, T. D. Noia, and Felice Antonio Merra. 2019. Assessing the impact of a user-item collaborative attack on class of users. *ArXiv*, abs/1908.07968.

Minghong Fang, Neil Zhenqiang Gong, and Jia Liu. 2020. Influence function based data poisoning attacks to top-n recommender systems. *Proceedings of The Web Conference 2020*.

Minghong Fang, Guolei Yang, Neil Zhenqiang Gong, and Jia Liu. 2018. Poisoning attacks to graph-based recommender systems. *Proceedings of the 34th Annual Computer Security Applications Conference*.

Brian Formento, Chuan Sheng Foo, Luu Anh Tuan, 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: EACL 2023*, pages 1–34.

Ji Gao, Jack Lanchantin, Mary Lou Soffa, and Yanjun Qi. 2018. Black-box generation of adversarial text sequences to evade deep learning classifiers. In *2018 IEEE Security and Privacy Workshops (SPW)*, pages 50–56. IEEE.

Shijie Geng, Shuchang Liu, Zuohui Fu, Yingqiang Ge, and Yongfeng Zhang. 2022. Recommendation as language processing (rlp): A unified pretrain, personalized prompt & predict paradigm (p5). In *RecSys 2022*.

Yupeng Hou, Zhankui He, Julian McAuley, and Wayne Xin Zhao. 2023a. Learning vector-quantized item representation for transferable sequential recommenders. In *WWW 2023*.

Yupeng Hou, Shanlei Mu, Wayne Xin Zhao, Yaliang Li, Bolin Ding, and Ji-Rong Wen. 2022. Towards universal sequence representation learning for recommender systems. In *KDD 2022*.

Yupeng Hou, Junjie Zhang, Zihan Lin, Hongyu Lu, Ruobing Xie, Julian McAuley, and Wayne Xin Zhao. 2023b. Large language models are zero-shot rankers for recommender systems. *ArXiv*, abs/2305.08845.

Hai Huang, Jiaming Mu, Neil Zhenqiang Gong, Qi Li, Bin Liu, and Mingwei Xu. 2021. Data poisoning attacks to deep learning based recommender systems. *ArXiv*, abs/2101.02644.

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 *AAAI 2020*, volume 34, pages 8018–8025.

Wang-Cheng Kang, Jianmo Ni, Nikhil Mehta, Maheswaran Sathiamoorthy, Lichan Hong, Ed H. Chi, and Derek Zhiyuan Cheng. 2023. Do llms understand user preferences? evaluating llms on user rating prediction. *ArXiv*, abs/2305.06474.

Parneet Kaur and Shivani Goel. 2016a. Shilling attack models in recommender system. In *ICICT 2016*, volume 2, pages 1–5.

Parneet Kaur and Shivani Goel. 2016b. Shilling attack models in recommender system. *2016 International Conference on Inventive Computation Technologies (ICICT)*, 2:1–5.

Bo Li, Yining Wang, Aarti Singh, and Yevgeniy Vorobeychik. 2016. Data poisoning attacks on factorization-based collaborative filtering. *ArXiv*, abs/1608.08182.

Jiacheng Li, Ming Wang, Jin Li, Jinmiao Fu, Xin Shen, Jingbo Shang, and Julian McAuley. 2023a. Text is all you need: Learning language representations for sequential recommendation. In *KDD 2023*.

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.

Xinhang Li, Chong Chen, Xiangyu Zhao, Yong Zhang, and Chunxiao Xing. 2023b. E4srec: An elegant effective efficient extensible solution of large language models for sequential recommendation.

Chen Lin, Si Chen, Hui Li, Yanghua Xiao, Lianyun Li, and Qian Yang. 2020. Attacking recommender systems with augmented user profiles. In *CIKM 2020*, pages 855–864.

Chen Lin, Si Chen, Meifang Zeng, Sheng Zhang, Min Gao, and Hui Li. 2024. Shilling Black-box Recommender Systems by Learning to Generate Fake User Profiles. *IEEE Transactions on Neural Networks and Learning Systems*, 35(1):1305–1319.

Jianghao Lin, Xinyi Dai, Yunjia Xi, Weiwen Liu, Bo Chen, Xiangyang Li, Chenxu Zhu, Huifeng Guo, Yong Yu, Ruiming Tang, and Weinan Zhang. 2023. How can recommender systems benefit from large language models: A survey.

Greg Linden, Brent Smith, and Jeremy York. 2003. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Comput.*, 7:76–80.

Junling Liu, Chaoyong Liu, Renjie Lv, Kangdi Zhou, and Yan Bin Zhang. 2023. Is chatgpt a good recommender? a preliminary study. *ArXiv*, abs/2304.10149.

Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton van den Hengel. 2015. Image-based recommendations on styles and substitutes. In *SIGIR 2015*, pages 43–52.

John 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 2020*, pages 119–126.

Sheshera Mysore, Andrew McCallum, and Hamed Zamani. 2023. Large language model augmented narrative driven recommendations. *Proceedings of the 17th ACM Conference on Recommender Systems*.

Junshuai Song, Zhao Li, Zehong Hu, Yucheng Wu, Zhenpeng Li, Jian Li, and Jun Gao. 2020. Poisonrec: An adaptive data poisoning framework for attacking black-box recommender systems. *2020 IEEE 36th International Conference on Data Engineering (ICDE)*, pages 157–168.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Sam Toyer, Olivia Watkins, Ethan Mendes, Justin Svegliato, Luke Bailey, Tiffany Wang, Isaac Ong, Karim Elmaaroufi, Pieter Abbeel, Trevor Darrell, Alan Ritter, and Stuart Russell. 2023. Tensor trust: Interpretable prompt injection attacks from an online game. *ArXiv*, abs/2311.01011.

Changsheng Wang, Jianbai Ye, Wenjie Wang, Chongming Gao, Fuli Feng, and Xiangnan He. 2023a. Recad: Towards a unified library for recommender attack and defense. In *RecSys 2023*.

Xiaolei Wang, Xinyu Tang, Wayne Xin Zhao, Jingyuan Wang, and Ji rong Wen. 2023b. Rethinking the evaluation for conversational recommendation in the era of large language models. *ArXiv*, abs/2305.13112.

Xiaolei Wang, Kun Zhou, Ji rong Wen, and Wayne Xin Zhao. 2022. Towards unified conversational recommender systems via knowledge-enhanced prompt learning. *KDD*.

Yawen Wang and Shihua Zhang. 2023. Prediction of tumor lymph node metastasis using wasserstein distance-based generative adversarial networks combing with neural architecture search for predicting. *Mathematics*.

Zongwei Wang, Min Gao, Junliang Yu, Hao Ma, Hongzhi Yin, and Shazia Sadiq. 2024. Poisoning attacks against recommender systems: A survey.

Shenghao Yang, Chenyang Wang, Yankai Liu, Kangping Xu, Weizhi Ma, Yiqun Liu, Min Zhang, Haitao Zeng, Junlan Feng, and Chao Deng. 2023. Collaborative word-based pre-trained item representation for transferable recommendation. In *ICDM 2023*.

Zheng Yuan, Fajie Yuan, Yu Song, Youhua Li, Junchen Fu, Fei Yang, Yunzhu Pan, and Yongxin Ni. 2023. Where to go next for recommender systems? id- vs. modality-based recommender models revisited. In *SIGIR 2023*.

Hengtong Zhang, Yaliang Li, Bolin Ding, and Jing Gao. 2020. Practical data poisoning attack against next-item recommendation. *Proceedings of The Web Conference 2020*.

Junjie Zhang, Ruobing Xie, Yupeng Hou, Wayne Xin Zhao, Leyu Lin, and Ji-Rong Wen. 2023a. Recommendation as instruction following: A large language model empowered recommendation approach.

Xudong Zhang, Zan Wang, Jingke Zhao, and Lanjun Wang. 2022. Targeted data poisoning attack on news recommendation system by content perturbation. *ArXiv*, abs/2203.03560.

Yang Zhang, Fuli Feng, Xiangnan He, Tianxin Wei, Chonggang Song, Guohui Ling, and Yongdong Zhang. 2021. Causal intervention for leveraging popularity bias in recommendation. In *SIGIR 2021*, pages 11–20.

Yang Zhang, Fuli Feng, Jizhi Zhang, Keqin Bao, Qifan Wang, and Xiangnan He. 2023b. Collm: Integrating collaborative embeddings into large language models for recommendation.

Kaijie Zhu, Jindong Wang, Jiaheng Zhou, Zichen Wang, Hao Chen, Yidong Wang, Linyi Yang, Wei Ye, Neil Zhenqiang Gong, Yue Zhang, and Xing Xie. 2023. PromptBench: Towards Evaluating the Robustness of Large Language Models on Adversarial Prompts.

Table 8: Statistics of the datasets

Dataset	#Users	#Items	#Interactions	Density
Sports	35,598	18,357	256,308	0.00039
Beauty	22,363	12,101	172,188	0.00064
Toys	19,412	11,924	145.004	0.00063

A Experimental Settings

A.1 Victim Models

We choose four mainstream LLM-based recommendation models as our victim models.

- **Recformer** (Li et al., 2023a). Recformer proposes to formulate an item as a "sentence" (word sequence) and can effectively recommend the next item based on language representations.
- **P5** (Geng et al., 2022). P5 presents a flexible and unified text-to-text paradigm called "Pretrain, Personalized Prompt, and Predict Paradigm" (P5) for recommendation, which unifies various recommendation tasks in a shared framework.
- **TALLRec** (Bao et al., 2023a). TALLRec proposes to align LLMs with recommendation by tunning LLMs with recommendation data.
- **CoLLM** (Zhang et al., 2023b). CoLLM seamlessly incorporates collaborative information into LLMs for recommendation by mapping ID embedding to the input token embedding space of LLM.

A.2 Compared Shilling Attacks

Shilling attacks aim to generate fake users that assign high ratings for a target item, while also rating other items to act like normal users for evading.

- Heuristic attacks. Heuristic attacks involve selecting items to create fake profiles based on heuristic rules. **Random attack** (Kaur and Goel, 2016a) selects filler items randomly while **Bandwagon attack** (Burke et al., 2005a) selects the popular items as users fake preferences, which is white-box as it requires knowledge of the popularity of items, i.e., the training data.
- Neural attacks. Neural attacks utilize neural networks to generate fake users that maximize the objective function. **Aush** (Lin et al.,

2020). Aush utilizes Generative Adversarial Network (GAN) to generate fake users based on known knowledge. **Leg-UP** (Lin et al., 2024). Leg-UP learns user behavior patterns from real users in the sampled "templates" and constructs fake user profiles. Both of them are gray-box attack models, requiring a portion of training data.

A.3 Implementation Details

The statistics of these datasets are summarized in Table 8. All victim models and compared shilling attacks are implemented in PyTorch. We random select 10% items as target items for each dataset. For RecFormer, we use both pre-trained checkpoint[1] and also fine-tune it with the three datasets. For P5, we directly use the fine-tuned checkpoints[2]. For TALLRec[3] and CoLLM[4], we fine-tune them from scratch. We implement shilling attack methods using RecAD[5]. Both Aush and Leg-UP are gray-box and we set them to access 20% of the training data.

B Text Attack

B.1 Text Attack Components

A textual attack consists of four main components: *Goal Function*, *Constraint*, *Transformation*, and *Search Method*. Here is a breakdown example of each component.

B.1.1 Goal Function

The Goal Function defines the objective of the attack. It also scores how "good" the given manipulated text is for achieving the desired goal. The core part could be simplified as:

```
def goal_function(target_item_id, original_text, perturbed_text, threshold=0.5):
    """
    Return the attacked score and determines if the attack is successful.

    :param target_item_id: The target item's id.
    :param original_text: The original text of target item.
    :param perturbed_text: The perturbed text of target item.
    :param threshold: The threshold of success attack.
    :return: attacked_score, is_successful
    """
    init_score = call_model(original_text, target_item_id)
    attacked_score = call_model(perturbed_text, target_item_id)
    is_successful = attacked_score - init_score > threshold
    return attacked_score, is_successful
```

[1]https://github.com/AaronHeee/RecFormer
[2]https://github.com/jeykigung/P5
[3]https://github.com/SAI990323/TALLRec
[4]https://github.com/zyang1580/CoLLM
[5]https://github.com/gusye1234/recad

B.1.2 Constraint

Constraints are conditions that must be met for the perturbed text to be considered valid. These often ensure the perturbed text remains natural and similar to the original text in some aspects (e.g., semantic similarity). Examples are as follows:

```
def maintain_semantic(original_text, perturbed_text, threshold=0.8):
    """
    Checks if the perturbed text maintains semantic similarity.

    :param original_text: Original text.
    :param perturbed_text: Perturbed version of the text.
    :param threshold: Threshold for semantic similarity.
    :return: True if similarity is above the threshold, False otherwise.
    """
    similarity = compute_semantic_similarity(input_text, perturbed_text)
    return similarity > threshold
```

B.1.3 Transformation

The Transformation component refers to the methods applied to modify the original text to achieve the adversarial goal. This could involve synonym replacement, insertion, or deletion of words.

```
def synonym_replacement(original_text):
    """
    Manipulates the original text by replacing synonyms.

    :param original_text: The original text to be manipulated.
    :return: A list of manipulated texts.
    """
    words = original_text.words
    transformed_texts = []
    for i in range(len(words)):
        replacement_word = get_synonyms(words[i])
        modified_text = original_text.replace_word_at_index(i, replacement_word)
        transformed_texts.append(modified_text)
    return transformed_texts
```

B.1.4 Search Method

The Search Method dictates the strategy used to explore the space of possible perturbations. For example, a greedy search method might iteratively apply transformations that maximally increase the attack's success likelihood.

```
def greedy_search(target_item_id, original_text):
    """
    Applies greedy search to find successful perturbation.

    :param target_item_id: The target item's id.
    :param original_text: Original text to be perturbed.
    :return: Best perturbed text.
    """
    best_score = 0
    best_perturbed_text = original_text
    perturbed_texts = get_transformations(original_text)
```

```
    for perturbed_text in perturbed_texts:
        attacked_score, is_successful = goal_function(target_item_id, original_text, perturbed_text)
        if satisfy_constraints(original_text, perturbed_text):
            if attacked_score > best_score:
                best_score = attacked_score
                best_perturbed_text = perturbed_text
                if is_successful:
                    return best_perturbed_text
    return None
```

B.2 Implementations

The majority of our text attacks have been developed by revising strategies from TextAttack[6] (Morris et al., 2020) and PromptBench[7] (Zhu et al., 2023).

All four attack methods, DeepwordBug, PuncAttack, TextFooler and BertAttack share the same *Goal Function*, where we set the success threshold to 0.05 increasing exposure rate for RecFormer. Since the other three victim models cannot perform full ranking and calculate exposure rates, we set an increase in interaction probability as the objective. Specifically, we set success threshold to $0.3, 0.15, 0.15$ increasing interaction probability for P5, TALLRec and CoLLM, respectively. During the attack process, we randomly select 10% of users to calculate the average exposure rate or interaction probability instead of using all users. This approach is lower in cost and more aligned with the constraints of practical attacks.

The recipes of *Constraint*, *Transformation*, and *Search Method* of implemented text attacks are as follows:

```
"""
Recipes for DeepwordBug
"""

transformation = CompositeTransformation(
    [
        WordSwapNeighboringCharacterSwap(),
        WordSwapRandomCharacterSubstitution(),
        WordSwapRandomCharacterDeletion(),
        WordSwapRandomCharacterInsertion(),
    ]
)

constraints = [
    RepeatModification(),
    StopwordModification(),
    LevenshteinEditDistance(30)
    ]
search_method = GreedyWordSwapWIR()
```

```
"""
Recipes for PuncAttack
"""
punctuations = '\'-'
transformation = WordSwapTokenSpecificPunctuationInsertion(letters_to_insert=punctuations)
constraints = [
    RepeatModification(),
    StopwordModification(),
    WordEmbeddingDistance(min_cos_sim=0.6),
    PartOfSpeech(allow_verb_noun_swap=True),
    UniversalSentenceEncoder(threshold=0.8)
    ]
search_method =  GreedyWordSwapWIR()
```

```
"""
Recipes for TextFooler
"""
transformation = WordSwapEmbedding(max_candidates=50)
constraints = [
    RepeatModification(),
    StopwordModification(),
    WordEmbeddingDistance(min_cos_sim=0.6),
    PartOfSpeech(allow_verb_noun_swap=True),
    UniversalSentenceEncoder(threshold=0.84,metric="angular")
    ]
search_method = GreedyWordSwapWIR()
```

```
"""
Recipes for BertAttack
"""
transformation = WordSwapMaskedLM(max_candidates=48)
constraints = [
    RepeatModification(),
    StopwordModification(),
    MaxWordsPerturbed(max_percent=1),
    UniversalSentenceEncoder(threshold=0.8)
    ]
search_method = GreedyWordSwapWIR()
```

C Detailed Experimental Results

In this section, we present detailed experimental results that could not be shown in the main text due to space limitation.

- The overall attack performances of P5, TALLRec and CoLLM are shown in Table 9, Table 10 and Table 11, respectively. The scatter plots of them are shown in Figure 7.
- The performance comparison with traditional shilling attacks of TALLRec and CoLLM are shown in 13 and 14.
- The performances of attacking fine-tuned Recformer is shown in Table 3.

D Case Studies

In this section, we present examples of attacks and defenses against RecFormer as a victim model, as shown in Table 15 - 20.

[6] https://github.com/QData/TextAttack
[7] https://github.com/microsoft/promptbench

Dataset	Method	Effectiveness			Stealthiness				
		Propensity ↑	Rel. Impro. ↑	# queries ↓	NDCG@10 ↑	Cos. ↑	Rouge-l ↑	Perplexity ↓	# pert. words ↓
Sports	Clean	0.35137	0.0%	-	0.28782	1.000	1.000	2158.7	-
	ChatGPT	0.36891	5.0%	-	0.28777	0.794	0.499	**1770.9**	-
	Trivial	0.34813	-0.9%	-	0.28764	**0.896**	**0.869**	4376.6	-
	Deepwordbug	0.41533	18.2%	**39.3**	0.28765	0.637	0.381	7417.3	4.9
	TextFooler	**0.42784**	**21.8%**	91.2	0.28768	0.688	0.499	2494.9	4.1
	PuncAttack	0.39292	11.8%	46.7	**0.28789**	0.842	0.602	2572.7	**3.4**
	BertAttack	0.41893	19.2%	135.7	0.28781	0.823	0.598	9421.4	3.7
Beauty	Clean	0.08218	0.0%	-	0.28765	1.000	1.000	611.6	-
	ChatGPT	0.07889	-4.0%	-	0.28733	0.822	0.516	**501.8**	-
	Trivial	0.07734	-5.9%	-	**0.28761**	**0.939**	**0.901**	1189.5	-
	Deepwordbug	0.23193	182.2%	**47.3**	0.28708	0.640	0.370	4590.1	4.9
	TextFooler	0.25010	204.3%	112.1	0.28691	0.683	0.460	1200.3	4.2
	PuncAttack	0.20653	151.3%	52.2	0.28693	0.828	0.594	1132.0	**3.4**
	BertAttack	**0.29618**	**260.4%**	146.8	0.28676	0.821	0.585	2634.5	3.5
Toys	Clean	0.26065	0.0%	-	0.28587	1.000	1.000	4060.4	-
	ChatGPT	0.28115	7.9%	-	0.28610	0.793	0.454	**1967.1**	-
	Trivial	0.26913	3.3%	-	0.28614	**0.880**	**0.852**	7874.4	-
	Deepwordbug	0.49867	91.3%	**28.0**	0.28619	0.642	0.490	8896.5	3.5
	TextFooler	**0.52492**	**101.4%**	65.0	0.28613	0.733	0.571	4413.1	3.0
	PuncAttack	0.42457	62.9%	31.3	**0.28637**	0.860	0.666	4134.3	**2.4**
	BertAttack	0.46528	78.5%	80.0	0.28597	0.855	0.690	9618.9	**2.4**

Table 9: Performance comparison of attacking **P5** where Rel. Impro. denotes relative improvement against clean setting. The best result is in **boldface**.

Dataset	Method	Effectiveness			Stealthiness				
		Propensity ↑	Rel. Impro. ↑	# queries ↓	NDCG@10 ↑	Cos. ↑	Rouge-l ↑	Perplexity ↓	# pert. words ↓
Sports	Clean	0.0399	-	-	0.58489	1.000	1.000	2158.7	-
	ChatGPT	0.0426	6.7%	-	0.58449	0.794	0.499	**1770.9**	-
	Trivial	0.0402	0.6%	-	**0.58479**	**0.896**	**0.869**	4376.6	-
	Deepwordbug	0.1074	168.8%	30.9	0.58459	0.643	0.459	8274.9	3.3
	TextFooler	**0.1093**	**173.5%**	61.3	0.58395	0.686	0.512	2075.5	2.7
	PuncAttack	0.0897	124.6%	**27.6**	**0.58489**	0.854	0.621	3153.9	2.0
	BertAttack	0.0955	139.1%	67.2	0.58484	0.848	0.682	5397.2	**1.7**
Beauty	Clean	0.0566	-	-	0.56758	1.000	1.000	611.6	-
	ChatGPT	0.0581	2.6%	-	0.56546	0.822	0.516	**501.8**	-
	Trivial	0.0558	-1.3%	-	**0.56755**	**0.939**	**0.901**	1189.5	-
	Deepwordbug	0.1605	183.6%	**26.7**	0.56737	0.674	0.531	2330.8	2.9
	TextFooler	**0.1724**	**204.6%**	53.4	0.56581	0.736	0.589	1491.0	2.5
	PuncAttack	0.1482	161.9%	29.8	0.56727	0.864	0.692	853.9	1.9
	BertAttack	0.1643	190.3%	70.4	0.56620	0.852	0.712	2036.6	**1.8**
Toys	Clean	0.5548	-	-	0.56822	1.000	1.000	4060.4	-
	ChatGPT	0.5602	1.0%	-	0.56666	0.793	0.454	**1067.1**	-
	Trivial	0.5245	-5.5%	-	0.56822	**0.880**	**0.852**	7874.4	-
	Deepwordbug	0.6786	22.3%	**21.5**	0.56822	0.650	0.493	8181.2	2.5
	TextFooler	**0.7098**	**27.9%**	45.6	**0.56886**	0.729	0.577	4953.7	2.1
	PuncAttack	0.6624	19.4%	22.8	0.56733	0.865	0.659	3418.2	1.7
	BertAttack	0.6798	22.5%	69.5	0.56850	0.874	0.733	8047.0	**1.5**

Table 10: Performance comparison of attacking **TALLRec** where Rel. Impro. denotes relative improvement against clean setting. The best result is in **boldface**.

Dataset	Method	Effectiveness			Stealthiness				
		Propensity ↑	Rel. Impro. ↑	# queries ↓	NDCG@10 ↑	Cos. ↑	Rouge-l ↑	Perplexity ↓	# pert. words ↓
Sports	Clean	0.35137	0.0%	-	0.28782	1.000	1.000	2158.7	-
	ChatGPT	0.36891	5.0%	-	0.28777	0.794	0.499	**1770.9**	-
	Trivial	0.34813	-0.9%	-	0.28764	**0.896**	**0.869**	4376.6	-
	Deepwordbug	0.41533	18.2%	**39.3**	0.28765	0.637	0.381	7417.3	4.9
	TextFooler	**0.42784**	**21.8%**	91.2	0.28768	0.688	0.499	2494.9	4.1
	PuncAttack	0.39292	11.8%	46.7	**0.28789**	0.842	0.602	2572.7	**3.4**
	BertAttack	0.41893	19.2%	135.7	0.28781	0.823	0.598	9421.4	3.7
Beauty	Clean	0.08218	0.0%	-	0.28765	1.000	1.000	611.6	-
	ChatGPT	0.07889	-4.0%	-	0.28733	0.822	0.516	**501.8**	-
	Trivial	0.07734	-5.9%	-	**0.28761**	**0.939**	**0.901**	1189.5	-
	Deepwordbug	0.23193	182.2%	**47.3**	0.28708	0.640	0.370	4590.1	4.9
	TextFooler	0.25010	204.3%	112.1	0.28691	0.683	0.460	1200.3	4.2
	PuncAttack	0.20653	151.3%	52.2	0.28693	0.828	0.594	1132.0	**3.4**
	BertAttack	**0.29618**	**260.4%**	146.8	0.28676	0.821	0.585	2634.5	3.5
Toys	Clean	0.26065	0.0%	-	0.28587	1.000	1.000	4060.4	-
	ChatGPT	0.28115	7.9%	-	0.28610	0.793	0.454	**1967.1**	-
	Trivial	0.26913	3.3%	-	0.28614	**0.880**	**0.852**	7874.4	-
	Deepwordbug	0.49867	91.3%	**28.0**	0.28619	0.642	0.490	8896.5	3.5
	TextFooler	**0.52492**	**101.4%**	65.0	0.28613	0.733	0.571	4413.1	3.0
	PuncAttack	0.42457	62.9%	31.3	**0.28637**	0.860	0.666	4134.3	**2.4**
	BertAttack	0.46528	78.5%	80.0	0.28597	0.855	0.690	9618.9	**2.4**

Table 11: Performance comparison of attacking **CoLLM** where Rel. Impro. denotes relative improvement against clean setting. The best result is in **boldface**

Dataset	Method	Effectiveness			Stealthiness				
		Exposure ↑	Rel. Impro. ↑	# queries ↓	NDCG@10 ↑	Cos. ↑	Rouge-l ↑	Perplexity ↓	# pert. words ↓
Sports	Clean	0.00261	-	-	0.01252	1.000	1.000	2158.7	-
	ChatGPT	0.00263	0.8%	-	**0.01237**	0.794	0.499	**1770.9**	-
	Trivial	0.00219	-16.1%	-	**0.01237**	**0.896**	**0.869**	4376.6	-
	Deepwordbug	0.00835	220.0%	**40.1**	0.01232	0.778	0.608	5086.0	3.1
	TextFooler	0.01074	311.8%	94.0	0.01228	0.775	0.595	2030.6	3.2
	PuncAttack	0.00932	257.2%	60.0	0.01235	0.864	0.672	2747.2	**2.6**
	BertAttack	**0.01111**	**325.9%**	165.0	0.01228	0.827	0.645	7382.3	3.2
Beauty	Clean	0.00432	-	-	0.03022	1.000	1.000	611.6	-
	ChatGPT	0.00356	-17.7%	-	0.02915	0.822	0.516	**501.8**	-
	Trivial	0.00390	-9.7%	-	**0.03022**	**0.939**	**0.901**	1189.5	-
	Deepwordbug	0.01348	212.0%	**49.4**	0.02960	0.744	0.591	3626.9	4.1
	TextFooler	0.01886	336.6%	116.3	0.02926	0.758	0.567	1372.3	4.5
	PuncAttack	0.01270	194.0%	72.9	0.02959	0.853	0.679	1115.9	**3.3**
	BertAttack	**0.01949**	**351.0%**	214.8	0.02928	0.822	0.642	2288.2	4.0
Toys	Clean	0.00429	-	-	0.03626	1.000	1.000	4060.4	**-**
	ChatGPT	0.00392	-8.6%	-	0.03580	0.793	0.454	**1967.1**	-
	Trivial	0.00407	-5.1%	-	**0.03623**	**0.880**	**0.852**	7874.4	-
	Deepwordbug	0.01268	195.7%	**33.6**	0.03610	0.703	0.542	11045.3	3.1
	TextFooler	**0.01725**	**302.5%**	86.8	0.03596	0.709	0.525	4762.2	3.3
	PuncAttack	0.01214	183.1%	41.1	0.03577	0.845	0.653	4350.4	**2.4**
	BertAttack	0.01381	222.1%	116.6	0.03599	0.829	0.661	12640.6	2.6

Table 12: Performance comparison of attacking **fintuned Recformer** where Rel. Impro. denotes relative improvement against clean setting. The best result is in **boldface**.

	Sports		Beauty		Toys	
Attack	AUC	Propensity	AUC	Propensity	AUC	Propensity
Clean	0.58489	0.03994	0.56758	0.05659	0.56822	0.55476
Random	0.53550	0.05450	**0.56603**	0.08472	0.56394	0.36513
Bandwagon	0.55781	0.04082	0.55167	0.00757	0.55189	0.54899
Aush	0.57643	0.02868	0.55416	0.00265	0.55416	0.52596
LegUP	0.54123	0.04361	0.53604	0.06628	0.55946	0.57901
TextFooler	**0.58395**	**0.10926**	0.56581	**0.17238**	**0.56886**	**0.70980**

Table 13: Shilling attacks on **TALLRec**.

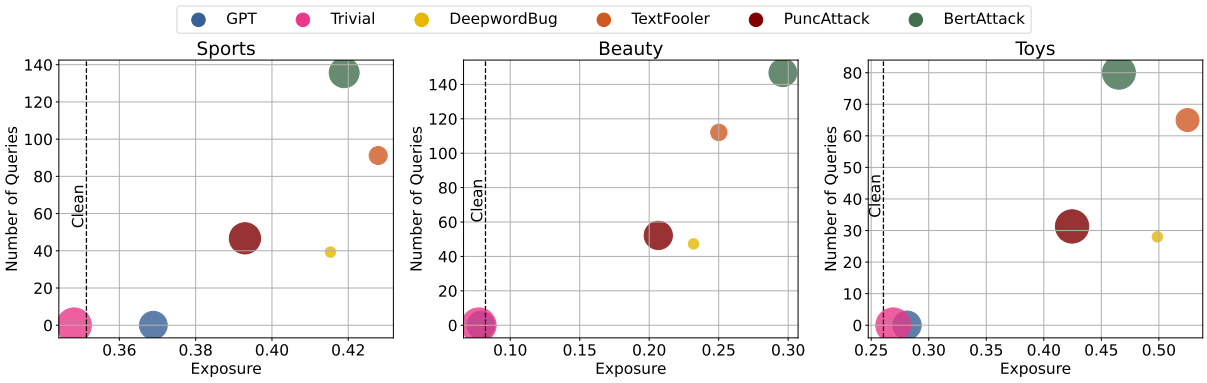

Figure 4: Performance comparison of different attacks on **P5**.

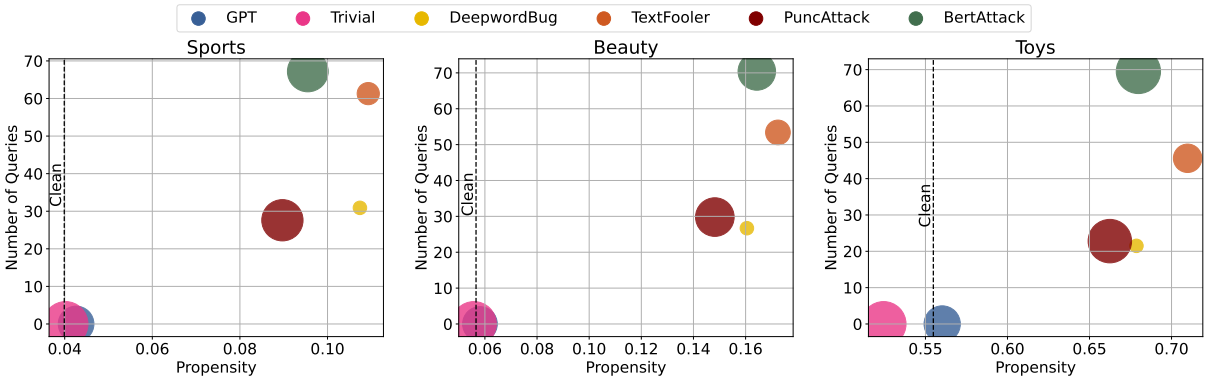

Figure 5: Performance comparison of different attacks on **TALLRec**.

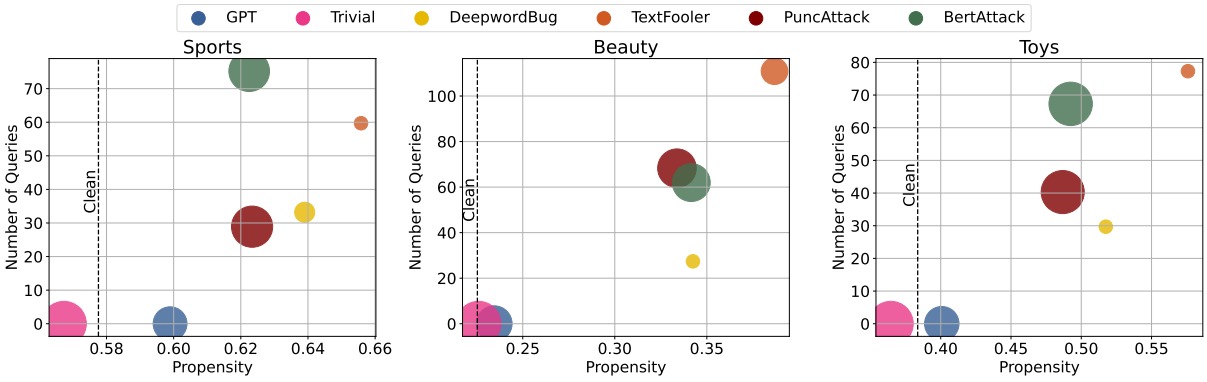

Figure 6: Performance comparison of different attacks on **CoLLM**.

Figure 7: Performance comparison of different attacks on various models. The size of the scatter points represents the cosine semantic similarity with the original title, with larger points indicating better semantic preservation (best viewed in color).

	Sports		Beauty		Toys	
Attack	AUC	Propensity	AUC	Propensity	AUC	Propensity
Clean	0.58331	0.57760	0.56256	0.22535	0.58622	0.38361
Random	0.55275	0.57192	0.52894	0.27450	0.54497	0.24871
Bandwagon	0.57669	0.56773	0.53884	0.23739	0.55721	0.34892
Aush	0.57767	0.55084	0.54857	0.24656	0.55373	0.49610
LegUP	0.57648	0.54010	0.53802	0.19230	0.54504	0.39779
TextFooler	**0.58285**	**0.65581**	**0.55993**	**0.38677**	**0.58706**	**0.57592**

Table 14: Shilling attacks on **CoLLM**.

Model	Text	Exposure
Clean	Little People Surprise Sounds Fun Park	0.0111
Trivial	Little People Surprise Sounds Fun Park good fantastic	0.0151
GPT	Exciting Sounds Fun Park for Little Ones	0.0201
DeepwordBug	Little ePople Surprise Sounds Fun Park	0.0701
+Defense	Little People Surprise Sounds Fun Park	0.0111
PunAttack	Little P-eople Surprise Sounds Fun Park	0.0496
+Defense	Little People Surprise Sounds Fun Park	0.0111
Textfooler	Little Inhabitants Surprise Sounds Fun Park	0.0702
+Defense	Surprising Audible Comic Park: Little Inhabitants	0.0586
BertAttack	Little joe Surprise Sounds Fun Park	0.0326
+Defense	Little Joe's Surprise Sound Fun Park	0.0198

Table 15: Item "B00008PVZG" in the Amazon-Toys dataset. The red part points out the differences from the original text.

Model	Text	Exposure
Clean	Fisher-Price Fun-2-Learn Smart Tablet	0.0076
Trivial	Fisher-Price Fun-2-Learn Smart Tablet better selling	0.0095
GPT	Interactive Learning Tablet for Kids	0.0335
DeepwordBug	Fisher-Price Fun-2-Learn Smar Tmblet	0.0335
+Defense	Fisher-Price Fun-2-Learn Smart Tablet	0.0076
PunAttack	Fisher-Price Fun–2-Learn Sm'art Tablet	0.0285
+Defense	Fisher-Price Fun-2-Learn Smart Tablet	0.0076
Textfooler	Fisher-Price Fun-2-Learn Canny Table	0.0768
+Defense	Fisher-Price Fun-2-Learn Canine Table	0.0756
BertAttack	Fisher-Price Fun-2-Learn this Tablet	0.0262
+Defense	Fisher-Price Fun-2-Learn Tablet	0.0190

Table 16: Item "B005XVCTAU" in the Amazon-Toys dataset. The red part points out the differences from the original text.

Model	Text	Exposure
Clean	Salon Grafix Healthy Hair Nutrition Cleansing Conditioner, 12 oz	0.0321
Trivial	Salon Grafix Healthy Hair Nutrition Cleansing Conditioner, 12 oz fantastic loved	0.0103
GPT	Nourishing Hair Care: Salon Grafix Cleansing Conditioner, 12 oz	0.0331
DeepwordBug	Salon Grafix Healthy Hair Nutirtion Cleansing Conditioner, 12 oz	0.1020
+Defense	Salon Grafix Healthy Hair Nutrition Cleansing Conditioner, 12 oz, 12 oz	0.0321
PunAttack	Salon Grafix Healthy Hair Nutrit-ion Cleansing Conditioner, 12 oz	0.0866
+Defense	Salon Grafix Healthy Hair Nutrition Cleansing Conditioner, 12 oz	0.0321
Textfooler	Salon Grafix Healthy Hair Nourishment Cleansing Conditioner, 12 oz	0.1438
+Defense	Salon Grafix Healthy Hair Nourishing Cleansing Conditioner, 12 oz	0.1135
BertAttack	Salon Grafix Healthy Hair style Cleansing Conditioner, 12 oz	0.1172
+Defense	Salon Grafix Healthy Hair Style Cleansing Conditioner, 12 oz.	0.1139

Table 17: Item "B007MNYY14" in the Amazon-Beauty dataset. The red part points out the differences from the original text.

Model	Text	Exposure
Clean	Phyto Organics Set Theratin Shampoo amp; Humectin Conditioner 1L Each	0.0014
Trivial	Phyto Organics Set Theratin Shampoo amp; Humectin Conditioner 1L Each purchase cheap	0.0015
GPT	Luxurious Phyto Organics Set: Theratin Shampoo Humectin Conditioner - 1L Each!	0.0019
DeepwordBug	Phyto Organics et heratin Shampoo amp; Humectin Conditioner 1L Each	0.0370
+Defense	Phyto Organics and Keratin Shampoo & Humectin Conditioner 1L Each	0.0170
PunAttack	Phyto Organic's Se't Theratin Shampoo amp; Humectin Conditioner 1L Each	0.0386
+Defense	Phyto Organics Set Theratin Shampoo amp; Humectin Conditioner 1L Each	0.0014
Textfooler	Phyto Organic Setting Theratin Shampoo amp; Humectin Conditioner 1L Each	0.0200
+Defense	Phyto Organic Setting Theratin Shampoo & Humectin Conditioner - 1L Each	0.0272
BertAttack	Phyto Organics for Theratin Shampoo makeup; Humectin Conditioner 1L Each	0.0483
+Defense	Phyto Organics for Theratin Shampoo & Hair Conditioner - 1L Each	0.0540

Table 18: Item "B0030UG27W" in the Amazon-Beauty dataset. The red part points out the differences from the original text.

Model	Text	Exposure
Clean	GAIAM Toeless Grippy Yoga Socks Toesocks	0.0017
Trivial	GAIAM Toeless Grippy Yoga Socks Toesocks wonderful product	0.0032
GPT	GAIAM Grippy Toeless Yoga Socks - Ultimate Toesocks	0.0036
DeepwordBug	GAIAM Toeless Grippy Yoga oScks Toesocks	0.0071
+Defense	GAIAM Toeless GripBpy Yoga Socks - Quality Socks	0.0041
PunAttack	GA-IAM Toeless Grip-py Yoga Sock's Toesocks	0.0085
+Defense	GA-IAM Toeless Grippy Yoga Sock's Toesocks	0.0053
TextFooler	GAIAM Toeless Grippy Yoga Sock Toesocks	0.0111
+Defense	Gaiam Toeless Grippy Yoga Socks - Toe Socks	0.0031
BertAttack	GAIAM Toeless Grippy Yoga with Toesocks	0.0080
+Defense	GAIAM Toeless Grip Yoga Socks with Toesocks	0.0056

Table 19: Item "B008EADJPG" in the Amazon-Sports dataset. The red part points out the differences from the original text.

Model	Text	Exposure
Clean	BladesUSA E419-PP Polypropylene Karambit Training Knife 6.7-Inch Overall	0.0014
Trivial	BladesUSA E419-PP Polypropylene Karambit Training Knife 6.7-Inch Overall quality excellent	0.0012
GPT	Ultimate Training Knife: BladesUSA E419-PP Karambit - Unbeatable Performance!	0.0041
DeepwordBug	BladesUSA E419-PP Polyproylene Karambit Training Knife 6.7-Inch Ovearll	0.0158
+ Defense	BladesUSA E419-PP Polypropylene Karambit Training Knife 6.7-Inch Overall	0.0014
PunAttack	Bla'desUSA E419-PP Polypropy'lene Karambit Training Knife 6.7-Inch Overall	0.0117
+ Defense	BladesUSA E419-PP Polypropylene Karambit Training Knife 6.7-Inch Overall	0.0014
TextFooler	BladesUSA E419-PP Polypropylene Karambit Training Knifes 6.7-Inch Overall	0.0036
+ Defense	BladesUSA E419-PP Polypropylene Karambit Training Knifes 6.7-Inch Overall	0.0036
BertAttack	BladesUSA E419-PP a Karambit Training Knife 6.7-Inch Overall	0.0095
+ Defense	BladesUSA E419-PP: A Karambit Training Knife with 6.7-Inch Overall Length	0.0082

Table 20: Item "B0089AH12I" in the Amazon-Sports dataset. The red part points out the differences from the original text.