Vulnerability of Text-Matching in ML/AI Conference Reviewer Assignments to Collusions

Jhih-Yi Hsieh [1] Aditi Raghunathan [1] Nihar B. Shah [1]

Abstract

OpenReview is an open-source platform for conference management that supports various aspects of conference peer review and is widely used by top-tier conferences in AI/ML. These conferences use automated algorithms on OpenReview to assign reviewers to paper submissions based on two factors: (1) reviewers' interests, indicated by their paper bids, and (2) domain expertise, inferred from the similarity between the text of their prior publications and submitted manuscripts. A major challenge is collusion rings, where groups of researchers manipulate the assignment process to review each other's papers positively, regardless of their actual quality. Most countermeasures target bid manipulation, assuming text similarity is secure. We demonstrate that, even without bidding, colluding authors and reviewers can exploit the text-matching component on OpenReview to be assigned to their target papers. Our results reveal specific vulnerabilities in the reviewer assignment system and offer suggestions to enhance its robustness.

1. Introduction

OpenReview is an open-source conference management platform dedicated to promoting openness in peer review and the broader scientific communication (Soergel et al., 2013), and it has been widely used for managing machine learning (ML) and artificial intelligence (AI) conferences. In recent years, OpenReview have experienced the rapid growth of demand in AI/ML venues, each receiving thousands to tens of thousands of paper submissions and similar numbers of reviewers (Shah, 2022). To manage this overwhelming volume, the assignment of reviewers to papers has become largely automated. A key component of this automated assignment pertains to text-matched similarity scores between reviewers' past work and submitted manuscripts, where natural language processing algorithms compute the similarity between the text of each submitted manuscript and the texts of the reviewer's previously published manuscripts. Models like SPECTER (Cohan et al., 2020), which generates embeddings based on textual content, have become widely used across various prestigious venues, including the Conference on Neural Information Processing Systems (NeurIPS), and are a default model on the OpenReview.

Although automation facilitates efficency, it also introduces vulnerabilities. One major challenge threatening the integrity of the peer review is the existence of *collusion rings* – groups of individuals who conspire to manipulate the review system for personal gain (Littman, 2021; Vijaykumar, 2020b). These rings can unfairly influence the acceptance of certain papers by orchestrating favorable reviews from colluding reviewers. For example, as noted in (Vijaykumar, 2020b), colluders may attempt to "game the system by exploiting vulnerabilities in the assignment algorithms to have their collaborators review their submissions."

The community has thus far focused on detecting and mitigating collusion through the analysis of *bidding*, the part of the reviewer assignment process where reviewers can indicate their interest in reviewing each paper. Research studies focus on bidding patterns (Wu et al., 2021; Jecmen et al., 2024). The algorithm in (Wu et al., 2021) considers text similarities as ground truth when examining the bids. In practice, program chairs have also focused on bidding when investigating collusion rings (Program Chairs, 2024); to address concerns over bidding manipulation, venues such as the Conference on Computer Vision and Pattern Recognition (CVPR) and ACL Rolling Review (ARR) have removed the bidding process altogether. This focus on bidding implicitly or explicitly assumes that the text-matching algorithms used for reviewer assignments are resistant to manipulation. However, this assumption warrants scrutiny.

In this paper, we investigate whether the text-matching algorithms used in automated reviewer assignments on OpenReview are robust to manipulation. Since most papers are

*Equal contribution [1]School of Computer Science, Carnegie Mellon University. Correspondence to: Jhih-Yi Hsieh <jhihyi.hsieh@gmail.com>, Nihar B. Shah <nihars@andrew.cmu.edu>.

Proceedings of the ICML 2025 Workshop on Championing Open-source Development in Machine Learning (CODEML '25). Copyright 2025 by the author(s).

assigned to 3–6 reviewers at conferences (Shah, 2022), we focus our evaluation on whether a pair of colluding author and reviewer can successfully manipulate the text-matching algorithms to give the colluding reviewer one of the top-1, top-3, or top-5 highest text similarity to the colluding paper among all reviewers at a conference. We find that the text matching algorithms are susceptible to attacks that can significantly increase the calculated similarity between a paper and its colluding reviewer. Here are some salient details:

- We find that the SPECTER (Cohan et al., 2020) text similarity matching algorithm can be manipulated. In evaluations on NeurIPS 2023 data, our attack can increase a colluding reviewer's similarity from ranked 101 to top-5 for a colluding paper 93% of the time.
- Reviewers are commonly allowed to curate their profiles used for text-matching. We find that carefully selecting past papers – in particular choosing of fewer papers – can greatly increase the attack's success rate. When a colluding reviewer keeps only one paper that is the most similar to the colluding paper, it increases the colluding reviewer's similarity from ranked 101 to top-5 for the colluding paper 41% of the time even without any modifications to the colluding paper's abstract.
- When reviewers have multiple papers in their profile, similarity-computation algorithms which assign paper submissions via the maximum of its similarities with the reviewer's past papers are more vulnerable than those which take the mean of the similarities with the reviewer's past papers. The attack successfully increases a colluding reviewer's similarity from ranked 101 to top-5 32% (mean) v.s. 49% (max) of the time.
- Considering NeurIPS 2022 as a "past" conference whose data is publicly available, and NeurIPS 2023 as the "current" conference whose data is unavailable, we find that the attack performance on NeurIPS 2022 is reflective of the performance in 2023. We find strong correlations of 0.62, 0.82, 0.92 and 0.93 between colluding reviewers' similarity rankings in NeurIPS 2022 and 2023 in four different settings.
- We conduct a human-subject experiment to test for the identifiability of adversarial abstracts, in which researchers are asked to evaluate manipulated or control abstracts. We find that the rate at which participants complain about coherence or consistency of the abstracts is higher in the manipulated attack abstracts, but there is also a non-trivial rate of complaints in control abstracts, which suggests plausible deniability.

In the final section, we discuss the implications of our findings and propose recommendations to enhance the security and robustness of automated reviewer assignments systems. We have informed OpenReview about our findings, and we are working closely together to deploy platform-wide safeguards. Independently, some top-tier conferences have also deployed defenses based on our recommendations.

We are release our code, datasets, and manipulation examples on GitHub: `https://github.com/passionfruit03/reviewer_assignments_vulnerability`. This paper aims to raise awareness around this vulnerability, which affects the fairness and integrity of peer review.

2. Problem setting

2.1. Paper-Reviewer Text Similarity

To assign reviewers to papers, the program chairs compute a **"similarity score"** for each reviewer-paper pair. For a paper p and reviewer r, the similarity score $s(p, r)$ is a number between 0 and 1, and a higher similarity indicates better reviewer expertise match. We focus on the vulnerability of text-based similarity scores, although there are other factors such as bidding. This is because bidding and subject-area matching are known to be vulnerable (Jecmen et al., 2020; Wu et al., 2021; Ailamaki et al., 2019) and can only increase the success of reviewer assignment manipulations.

In this work, we consider the SPECTER algorithm (Cohan et al., 2020), which is a neural network method widely used by ML/AI conferences such as NeurIPS and ICML. For any reviewer r, we let Q_r denote the archive containing 10 most recent papers authored by r. For any paper p, let $\mathbf{v}_p \in \mathbb{R}^{768}$ denote its SPECTER text embedding vector. Note that in practice, only the titles and abstracts are used to compute $\mathbf{v}_p$. Then, for a submitted paper p and a reviewer r, their similarity is the mean of cosine similarities between p and each paper in the reviewer's archive Q_r, or more formally

$$s(p, r) = \frac{1}{|Q_r|} \sum_{q \in Q_r} \frac{\mathbf{v}_p \cdot \mathbf{v}_q}{\|\mathbf{v}_p\|_2 \, \|\mathbf{v}_q\|_2}. \quad (1)$$

2.2. Attacker's Threat Model

The "attacker" represents both the colluding author of a paper p and a reviewer r working together to ensure p is assigned to r for peer review. Their objective is *increasing $s(p, r)$ so that r ranks in the top-k of the conference's reviewers, ranked by their similarity scores with p.* In our experiments, we consider $k \in \{1, 3, 5\}$ since conferences assign 3–6 reviewers to most papers (Shah, 2022).

We assume that each pair of colluding reviewer and author (the "attacker") know each other and work together on the attack. The colluding reviewer can adversarially curate their reviewer archive Q_r^{adv} (Section 3.1), and the colluding author can manipulate the abstract of their paper (Section 3.2).

Finally, the attackers have access to the exact SPECTER embeddings, with the model weights publicly available on GitHub `https://github.com/allenai/specter`. However, they do not have access to the reviewer pool of the current iteration of the conference. Still, many conferences publish previous reviewer pools, which our results show can have security implications (Appendix H).

3. Attack Procedure

3.1. Adversarial Reviewer Archive Curation

OpenReview and many peer review systems allow reviewers to curate own archives, which are used to compute similarity scores for assigning papers. Some conferences even encourage this practice, starting with an empty archive that reviewers populate with any papers they choose.

A colluding reviewer can exploit this process by constructing an adversarial archive, Q_r^{adv}, that includes only papers from their archive Q_r which are highly similar to the target paper p. The colluding reviewer can curate their archive to keep only one paper that has the highest similarity to p,

$$Q_r^{\text{adv}} = \left\{ \text{a random paper in } \underset{q \in Q_r}{\operatorname{argmax}} \frac{\mathbf{v}_p \cdot \mathbf{v}_q}{\|\mathbf{v}_p\|_2 \, \|\mathbf{v}_q\|_2} \right\}, \tag{2}$$

with ties broken uniformly at random, where $\mathbf{v}_p \in \mathbb{R}^k$ is the SPECTER embeddings defined in Section 2.1.

3.2. Adversarial Abstract Modifications

The colluding author can construct an adversarial abstract to increase the simiarity to the reviewer (after archive curation). We find two types of abstract modifications that effectively increase the similarity and formalize them into two simple operations. The first is **INCLUDETHEMES** which involves adding background or filler sentences related to the main ideas of papers in Q_r^{adv}. The second is **INSERTKEYWORDS** which inserts "keywords" that target SPECTER to increase $s(p, r)$ even if the keywords do not necessarily seem important to humans. This operation is inspired by works in adversarial robustness (Ebrahimi et al., 2017; Jin et al., 2020; Li et al., 2020). For detailed explanations with examples and formal algorithms, see Appendix A and K, respectively.

In this work, we use a Large Language Model (LLM) to perform *fully automatic* abstract modifications without human supervision. This makes large-scale evaluations and further ablations feasible. To ensure modified abstract quality, we instruct LLMs to stay consistent with the paper and ensure general coherence (see Appendix J for prompts). To simulate real attackers, a small 25-samples experiment on abstracts with *human-in-the-loop* modifications can be found in the Appendix E. In addition, we conduct a human subject experiment to study the detectability of adversarial abstracts (Appendix I). The LLM model we use is the OpenAI gpt-4-0125-preview model.

4. Results

4.1. Experiment Setup

We download from OpenReview and curate a dataset of 7,900 reviewers (and their archives) and all 3218 accepted papers from the Neural Information Processing Systems (NeurIPS) 2023 conference. Our evaluations are based on the rankings of reviewer similarities. For any paper p and reviewer r at a conference, we define the *natural ranking* of (p, r) as the rank position of r when all reviewers at the conference are ranked by their similarity scores with the paper p. We use the term *manipulated ranking* for the colluding reviewer's rank position after manipulations. Appendix B relates natural rankings to absolute similarity.

In our dataset, for each paper p, we find reviewers with natural rankings of 20, 101, 501, and 1001. For each of those scenarios, we evaluate the effectiveness of our attack by measuring the top-1, top-3, and top-5 success rates, where a top-N success rate is defined as the fraction of times the attack successfully increases the colluding reviewer's manipulated ranking to be top-N for the colluding paper. More details on experiment setup are in Appendix C, and we study attack budgets in Appendix D.

4.2. Attack Success Rates

We sample 300 (paper p, reviewer r) pairs where r has a natural ranking of 101 for p. In addition, we sample 100 (p, r) pairs each for when the reviewer natural ranking is 20, 501, and 1001. The 101st scenario has more samples by request from a reviewer of this paper. Then, we perform the attack for each (p, r) pair by curating Q_r^{adv} and modifying abstract. In Table 1, we find that the success rates are high. When the natural ranking is 20, the colluders are naturally similar and attack success rates are the highest. When the natural ranking is 101, the proposed attack procedure leads to a top-5 attack success rate of 93%. Even when the natural ranking is 1001, the top-1 success rate is 48%. These results highlight the potential that colluding reviewers who are not working directly in the same area can still collude to successfully manipulate reviewer assignments. Appendix F further explores the relationships between success rates and absolute similarity. While all our experiments uses *fully automatic* abstract modifications for scalability, Appendix E evaluates 25 abstracts with *human-in-the-loop* modifications.

4.3. Reviewer Archive Curation

Currently, in most venues, reviewers are allowed – and frequently even encouraged – to curate their profiles. We find

Table 1: Attack success rates for colluding reviewers with natural rankings of 20, 101, 501, and 1001.

	Attack Success Rates (± SE)			Manipulated Rankings
Natural Rankings	Top-1	Top-3	Top-5	Mean 95% CI
20	90(3) %	96(2) %	98(1) %	1.28 [1.06, 1.50]
101	74(3) %	89(2) %	93(2) %	2.22 [1.80, 2.64]
501	60(5) %	76(4) %	83(4) %	6.58 [3.47, 9.69]
1001	48(5) %	63(5) %	67(5) %	15.68 [6.82, 24.54]

Table 2: Attack performances using average or max pooling in similarity $s(p, r)$ calculation (natural ranking 101).

	Attack Success Rates (± SE)			Manipulated Rankings
Aggregation Method	Top-1	Top-3	Top-5	Mean 95% CI
Average	13(3) %	24(4) %	32(5) %	18.20 [14.45, 21.95]
Maximum	20(4) %	40(5) %	49(5) %	9.37 [7.52, 11.22]

that this policy makes the conferences seriously vulnerable to adversarial archive curation (Section 3.1). With archive curation (but no abstract modifications), for colluding (p, r) pairs with natural rankings of 101, we find that 15%, 30%, and 41% of the samples have manipulated rankings being within the top-1,3 and 5, respectively. Reviewer archive curation *alone* is a serious threat to automated reviewer assignments. Appendix G further shows that attacks are less successful when reviewers must keep more papers in Q_r^{adv}.

4.4. Maximum versus Average Similarity

In Section 2.1, we discuss how the SPECTER similarity is calculated (Equation 1). Instead of the mean, another popular approach is taking the max: $s(p, r) = \max_{q \in Q_r} \frac{\mathbf{v}_p \cdot \mathbf{v}_q}{\|\mathbf{v}_p\|_2 \|\mathbf{v}_q\|_2}$. So far, our analyses involve reviewer archive curation to keep a single paper, under which max and average pooling similarity definitions become the same. In this section, we compare the attack success rates *without* adversarial curation (that is, $Q_r^{\text{adv}} = Q_r$). We randomly sample without replacement 100 (p, r) pairs with natural rankings of 101 under each aggregation method, and then run abstract modifications. Table 2 shows that the attack is more effective against maximum aggregation. This aligns with our expectation, since modifications can target just one paper in Q_r when the max is taken.

5. Related work

For more detailed investigation into collusion rings, see (Vijaykumar, 2020a; Littman, 2021). For a more extensive survey on text matching algorithms, see Shah (2022, Section 3). Among the various components of the similarity score computation, research on addressing collusion rings has primarily focused on bidding (Wu et al., 2021; Jecmen et al., 2022; 2023; 2024; Leyton-Brown et al., 2024). For research on manipulating text matching, some have focused on exploiting PDF parsers (Markwood et al., 2017; Tran & Jaiswal, 2019) while others have demonstrated the possibility of purely text-level attacks (Eisenhofer et al., 2023). Our work study text-level attacks at a significantly larger scale and remove key assumptions to demonstrate vulnerabilities that are simple to exploit in practice.

6. Discussion

This work raises awareness about the vulnerabilities of current reviewer matching systems by showing a simple and practical attack that is effective. Furthermore, we conducted a human subject experiment testing for adversarial abstract detectability (Appendix I). While some noted issues with coherence and consistency, similar complaints arose for benign abstracts, suggesting plausible deniability. No malicious intent was identified by participants. Safeguards based on our recommendations are now in place at several major AI/ML conferences, and we are working with OpenReview to support safeguards natively on their platform.

We offer several recommendations. Firstly, colluding reviewers can effectively exploit archive curation, so requiring reviewers to have more extensive publication archives reduces the effectiveness of manipulation. Next, using average instead of max aggregation in the similarity calculation can reduce the impact of targeted manipulations by limiting the influence of any single paper, but if max pooling yields more accurate matches, a compromise like the 75th percentile can offer both robustness and match quality. Thirdly, educating reviewers about manipulations can lead them stay vigilant. Fourth, consider adopting a broader mitigation strategy, such as introducing randomness into reviewer assignment (Jecmen et al., 2020). Lastly, developers of similarity algorithms should consider the robustness of their methods.

6.1. Limitations

We focus on collusion with a single reviewer, but exploring cases with multiple colluding reviewers could reveal deeper vulnerabilities. We also omit practical constraints like reviewer load, conflicts of interest, and geographic factors, which may affect attack effectiveness and generalizability.

Acknowledgments

This work was supported in part by grants ONR N000142212181 and NSF 2200410, 1942124, and a J.P. Morgan research scholar program. AR gratefully acknowledges support from the AI2050 program at Schmidt Sciences (Grant #G2264481), Google Research Scholar program, Apple, Cisco, Open Philanthropy, and NSF 2310758.

Impact Statement

As other Computer Science (CS) fields grow in size, our findings may be more widely applicable. The main risk of this work is the *disclosure* of a serious vulnerability in reviewer matching systems, which could allow unfair advantages and harm publication quality. To mitigate this risk, we had informed OpenReview and other conferences before submission and worked with them to address these vulnerabilities using the defenses we propose. **These safeguards are now in place and are being used.**

Collusions and attacks described in this paper may already be happening in practice, yet most researchers may not be aware of these vulnerabilities. To promote transparency, we make our attack algorithm and adversarial abstract examples publicly available. By raising awareness of these vulnerabilities, we aim to empower non-colluding reviewers to be more vigilant. Furthermore, attacks described in this work are difficult to detect and often come with *plausible deniability*. Given the challenge of proving intent based on abstract content or profile changes, proactive mitigation is crucial rather than relying solely on post hoc detection and enforcement.

References

Ailamaki, A., Chrysogelos, P., Deshpande, A., and Kraska, T. The sigmod 2019 research track reviewing system. *ACM SIGMOD Record*, 48(2):47–54, 2019.

Benjamini, Y. and Hochberg, Y. Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal statistical society: series B (Methodological)*, 57(1):289–300, 1995.

Charlin, L. and Zemel, R. The toronto paper matching system: an automated paper-reviewer assignment system. *The International Conference on Learning Representations*, 2013.

Cohan, A., Feldman, S., Beltagy, I., Downey, D., and Weld, D. S. SPECTER: Document-level representation learning using citation-informed transformers. *arXiv preprint arXiv:2004.07180*, 2020.

Ebrahimi, J., Rao, A., Lowd, D., and Dou, D. HotFlip: White-box adversarial examples for text classification. *arXiv preprint arXiv:1712.06751*, 2017.

Eisenhofer, T., Quiring, E., Möller, J., Riepel, D., Holz, T., and Rieck, K. No more reviewer# 2: Subverting automatic paper-reviewer assignment using adversarial learning. In *32nd USENIX Security Symposium (USENIX Security 23)*, pp. 5109–5126, 2023.

Jecmen, S., Zhang, H., Liu, R., Shah, N., Conitzer, V., and Fang, F. Mitigating manipulation in peer review via randomized reviewer assignments. *Advances in Neural Information Processing Systems*, 33:12533–12545, 2020.

Jecmen, S., Shah, N. B., Fang, F., and Conitzer, V. Tradeoffs in preventing manipulation in paper bidding for reviewer assignment. In *ICLR workshop on ML Evaluation Standards*, 2022.

Jecmen, S., Yoon, M., Conitzer, V., Shah, N. B., and Fang, F. A dataset on malicious paper bidding in peer review. In *TheWebConf*, 2023.

Jecmen, S., Shah, N. B., Fang, F., and Akoglu, L. On the detection of reviewer-author collusion rings from paper bidding. *arXiv preprint arXiv:2402.07860*, 2024.

Jin, D., Jin, Z., Zhou, J. T., and Szolovits, P. Is BERT really robust? a strong baseline for natural language attack on text classification and entailment. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 8018–8025, 2020.

Kobren, A., Saha, B., and McCallum, A. Paper matching with local fairness constraints. In *ACM KDD*, 2019.

Leyton-Brown, K., Nandwani, Y., Zarkoob, H., Cameron, C., Newman, N., and Raghu, D. Matching papers and reviewers at large conferences. *Artificial Intelligence*, 331:104119, 2024.

Li, L., Ma, R., Guo, Q., Xue, X., and Qiu, X. Bert-attack: Adversarial attack against bert using bert. *arXiv preprint arXiv:2004.09984*, 2020.

Littman, M. L. Collusion rings threaten the integrity of computer science research. *Communications of the ACM*, 64(6):43–44, 2021.

Markwood, I., Shen, D., Liu, Y., and Lu, Z. PDF mirage: Content masking attack against Information-Based online services. In *26th USENIX Security Symposium (USENIX Security 17)*, pp. 833–847, Vancouver, BC, August 2017. USENIX Association. ISBN 978-1-931971-40-9. URL `https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/markwood`.

Payan, J. and Zick, Y. I will have order! optimizing orders for fair reviewer assignment. In *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems*, pp. 1711–1713, 2022.

Program Chairs, I. . Slide from ICLR 2024 program chair presentation, 2024. `https://twitter.com/chriswolfvision/status/1787886748434878769`. Last accessed 16 May 2024.

Shah, N. B. Challenges, experiments, and computational solutions in peer review. Communications of the ACM. Preprint available at `https://www.cs.cmu.edu/~nihars/preprints/SurveyPeerReview.pdf`, June 2022.

Soergel, D., Saunders, A., and McCallum, A. Open scholarship and peer review: a time for experimentation. In *Proceedings of the 30th International Conference on Machine Learning*, 2013.

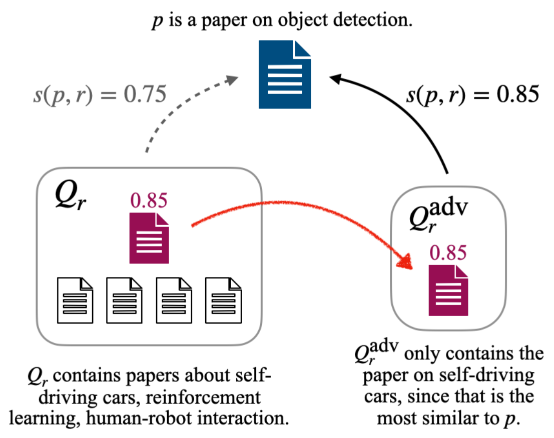
Stelmakh, I., Shah, N., and Singh, A. PeerReview4All: Fair and accurate reviewer assignment in peer review. *Journal of Machine Learning Research*, 2021.

Tran, D. and Jaiswal, C. PDFPhantom: Exploiting pdf attacks against academic conferences' paper submission process with counterattack. In *2019 IEEE 10th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)*, pp. 0736–0743, 2019. doi: 10.1109/UEMCON47517.2019.8992996.

Vijaykumar, T. N. Potential organized fraud in on-going ASPLOS reviews, 2020a. `https://medium.com/@tnvijayk/potential-organized-fraud-in-on-going-asplos-reviews-874ce14a3ebe`. Last accessed 29 April 2024.

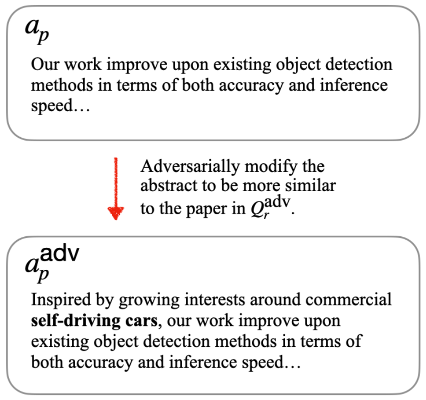
Vijaykumar, T. N. Potential organized fraud in ACM/IEEE computer architecture conferences, 2020b. `https://medium.com/@tnvijayk/potential-organized-fraud-in-acm-ieee-computer-architecture-conferences-ccd61169370d`. Last accessed 29 April 2024.

Wu, R., Guo, C., Wu, F., Kidambi, R., Van Der Maaten, L., and Weinberger, K. Making paper reviewing robust to bid manipulation attacks. In *International Conference on Machine Learning*, pp. 11240–11250. PMLR, 2021.



p is a paper on object detection.
$s(p, r) = 0.75$
$s(p, r) = 0.85$
Q_r
0.85
Q_r^{adv}
0.85
Q_r contains papers about self-driving cars, reinforcement learning, human-robot interaction.
Q_r^{adv} only contains the paper on self-driving cars, since that is the most similar to p.

(a) The colluding reviewer adversarially curates their archive to only contain one of their past papers that is the most similar to the paper p they want to get assigned.



a_p
Our work improve upon existing object detection methods in terms of both accuracy and inference speed…
Adversarially modify the abstract to be more similar to the paper in Q_r^{adv}.
a_p^{adv}
Inspired by growing interests around commercial **self-driving cars**, our work improve upon existing object detection methods in terms of both accuracy and inference speed…

(b) The colluding author adversarially modifies their abstract to be more similar to the paper in the colluding reviewer's curated archive.

Figure 1: An illustrated example of the attack procedure. 1a shows the reviewer's action; 1b shows the author's action.

A. Details on Attack Procedure

Notation Only titles and abstracts are used to compute SPECTER similarity scores between papers, and reviewers can curate their archives adversarially. To explain the attack procedure in more detail, we will expand the notation for paper p to have corresponding t_p for the title and a_p for the abstract as we will manipulate those, and we will explicitly denote the reviewer archive. Hence, we use $s((t_p, a_p), Q_r)$ to denote paper-reviewer similarity below.

Attack surface and constraints We assume that each pair of colluding author and reviewer (the "attacker") know each other and are actively colluding, so they can work together to create the attack. The colluding author can manipulate the abstract of their paper, and the colluding reviewer can adversarially curate their reviewer archive Q_r^{adv} to only retain selected papers. We only consider abstract modifications, since the SPECTER similarity scores are computed based on title and abstract only, though the author can change any parts of their paper. In this paper, we also do not consider changes to the title because we suspect title modifications may arouse more suspicion from non-colluding reviewers.

When modifying the abstract, the attacker must avoid arousing suspicion in non-colluding reviewers. This is ostensibly vague and we formalize this notion via two constraints on adversarial modifications of the abstract: (i) *Coherence*: The abstract should use academic writing style, have natural flow, and cannot contain scientifically false information. (ii) *Consistency*: The abstract should be consistent with the paper's contents. In this paper, we also enforce constraints on the number of sentences and keywords that can be added to the colluding paper's abstract. Attacks are generally allowed only one sentence that is related to Q_r^{adv}, but we allow up to three sentences related to Q_r^{adv} in some cases only when we have *human-in-the-loop* to ensure coherence and consistency. In addition, attacks are allowed to add up to 10 keywords selected from Q_r^{adv}. We also perform human-study experiments to judge whether non-colluding reviewers find that abstracts generated by our proposed attack are suspicious.

A.1. Annotated Examples

A.1.1. INCLUDETHEMES

The goal of **INCLUDETHEMES** is to modify the abstract a_p in order to increase the SPECTER similarity between paper p and reviewer r, by adding background or filler sentences related to the central themes of papers in Q_r^{adv}. We note that the resulting modified abstract a_p^{adv} may become inconsistent with its paper p if the main ideas in the abstract have been changed entirely. Our key observation is that SPECTER similarities between papers and reviewers increase when abstracts share overlapping themes, even when those themes are central to papers in Q_r^{adv} but only referenced as background information in a_p^{adv}. This means that **INCLUDETHEMES** can produce a_p^{adv} by including themes from Q_r^{adv} to increase $s((t_p, a_p^{\text{adv}}), Q_r^{\text{adv}})$ without violating the coherence and consistency constraints. An example of **INCLUDETHEMES** can be found in Figure 2.

INCLUDETHEMES Example: Multimodal language models have shown promise in AI applications like robotics, where these models enable scalable approaches for learning open-world object-goal navigation – the task of asking a virtual robot agent to find any instance of an object in an unexplored environment (e.g., "find a sink"). In this work, we propose a new method to fuse the embedding space of frozen text-only large language models (LLMs) and pre-trained image encoder and decoder models, by mapping between their embedding spaces. Our model demonstrates a wide suite of multimodal capabilities: image retrieval, novel image generation, and multimodal dialogue...

Figure 2: An example of **INCLUDETHEMES** modifying the abstract. Goal navigation is an important theme in the adversarial archive Q_r^{adv} of the reviewer r. The modification (highlighted sentence) adds robot goal navigation as a motivating example for further improvements to multimodal models. The resulting abstract a_p^{adv} remains coherent and consistent with the paper's focus on text and image embedding alignment, but has increased similarity to the target adversarial archive $s((t_p, a_p^{\text{adv}}), Q_r^{\text{adv}})$. The **INCLUDETHEMES** changes shown in this example, alongside adversarial reviewer archive curation, increase the reviewer's similarity to the paper from being 101st most-similar to become the *most* similar amongst all reviewers at the NeurIPS 2023 conference.

The implementation of the *human-in-the-loop* and *fully automatic* modes differ in whether human supervision is used to ensure the coherence and consistency of the modified abstract a_p^{adv}. In the *human-in-the-loop* mode, modified abstracts a_p^{adv} are created by a human (potentially with the help of a LLM), and the human can make incremental edits to a_p^{adv} to ensure coherence and consistency. On the other hand, the *fully automatic* mode only uses a LLM to generate the a_p^{adv} with a prompt that emphasizes coherence and consistency, and it does not allow further edits to a_p^{adv} once they are generated.

In addition, the implementation for both *human-in-the-loop* and *fully automatic* modes involves generating $\boldsymbol{N}$ different versions of modified abstracts. Since generating and modifying abstracts is stochastic (due to the stochasticity of LLM outputs and manual edits), we keep only the attempt that is the most similar to the colluding reviewer r. We tune and report the choice of $\boldsymbol{N}$ for our experiments in Appendix C, and the formal algorithm is Algorithm 1 in Appendix K.1.

A.1.2. INSERTKEYWORDS

In order to further raise the similarity, **INSERTKEYWORDS** adds specific *keywords* to the abstract. These keywords may not seem obviously important to humans, but they can increase $s((t_p, a_p^{\text{adv}}), Q_r^{\text{adv}})$ if added into the adversarial abstract. This phenomenon aligns with findings in the adversarial examples literature, where researchers have found the output of neural networks to be brittle to the insertion of certain keywords or tokens into the input. Furthermore, we find that the similarities can increase even when the keywords are used *under different meanings and with different parts of speech* across abstracts. This allows **INSERTKEYWORDS** to insert technical keywords from Q_r^{adv} into the a_p^{adv} without introducing unrelated or suspicious concepts. For example, "transfer learning" might be simplified to "transfer," a common English term. To ensure coherence and grammatical correctness, **INSERTKEYWORDS** may also adjust text around the inserted keywords (e.g. insert a phrase that contains the keyword).

We present an example of **INSERTKEYWORDS** in Figure 3. In this example, the paper in adversarial archive Q_r^{adv} proposes mitigating label scarcity in transfer learning with data augmentation, in particular using gradient flow methods to the minimize maximum mean discrepancy loss on the feature-Gaussian manifold. To search for the keywords that increase the similarity, we propose the FINDKEYWORDS subroutine, which is described later, to suggests keywords from Q_r^{adv} to add to the abstract a_p, including repeated words if they can further increase similarity.

Some keywords from the reviewer's archive Q_r^{adv} carry meanings directly related to the abstract a_p. In Figure 3, the keyword 'learning' is added as 'machine learning'; 'feature-label' is broken up and added as 'labels and features'; 'feature-Gaussian' is inserted only as 'Gaussian'. For these keywords, the inserted variations match their meanings in the reviewer's archive. However, other keywords with technical meanings unrelated to this paper p, such as 'manifold', 'discrepancy', and 'flow', are adapted to common English ('manifold' as "a great deal", 'discrepancy' as "difference", and 'flow" referring to training pipelines). Still, there can be unrelated technical keywords like 'Riemannian' and 'optimum' that do not have suitable common English meanings for abstract a_p, so they would not be inserted.

Now, we describe the implementation details of the **INSERTKEYWORDS** operation. The process iteratively searches for $\boldsymbol{M}$ batches of $\boldsymbol{K}$ keywords, inserting each batch before searching for the next. We propose the FINDKEYWORDS subroutine (Algorithm 3 in Appendix K.2) to greedily choose keywords that increase the similarity $s((t_p, a_p^{\text{adv}}), Q_r^{\text{adv}})$.

Example: With more real-world machine learning applications, the importance of safeguarding data privacy of labels and features has increased **manifold**. Per-example gradient clipping is a key algorithmic step that enables practical differential private (DP) training for deep learning models. The choice of clipping threshold R, however, is vital for avoiding high training loss **discrepancy** and achieving high accuracy under DP. Not only does it serve as a clipping threshold, but the Gaussian noises added are also dependent on R. We propose an easy-to-use replacement, called automatic clipping, that eliminates the need to tune R for any DP optimizers, including DP-SGD, DP-Adam, DP-LAMB and many others. The automatic variants are as private and computationally efficient as existing DP optimizers, but require no DP-specific hyperparameters and thus improve DP training **manifold**. Our proposed method is as amenable as the standard non-private training **flow**. . .

Figure 3: An example of **INSERTKEYWORDS** modifications. The paper in the adversarial archive Q_r^{adv} proposes mitigating label scarcity in transfer learning with data augmentation. Some inserted keywords (highlighted) match their original meanings from the paper in Q_r^{adv}, while others (highlighted + bolded) are adapted to common English. The **INSERTKEYWORDS** changes to this abstract, alongside adversarial reviewer archive curation, increase the reviewer's similarity to the paper from being 101st most-similar to 3rd most-similar amongst all reviewers at the NeurIPS 2023 conference.

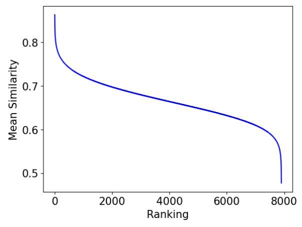
Since FINDKEYWORDS is a greedy algorithm, the alternation between finding and inserting each batch of keywords can take into account the new a_p^{adv} when finding the next batch of keywords. The two hyperparameters, $\boldsymbol{M}$ and $\boldsymbol{K}$, in **INSERTKEYWORDS** are tuned and reported in Appendix C. The formal algorithm is Algorithm 2 in Appendix K.2.

INSERTKEYWORDS also has *human-in-the-loop* and *fully automatic* modes (detailed in Algorithm 2). In the *human-in-the-loop* mode, a human can oversee if and how the keywords can be added while abiding by coherence and consistency constraints. In the *fully automatic* mode, the LLM is prompted to be coherent and consistent without human supervision. A more detailed description can be found in Appendix 2.

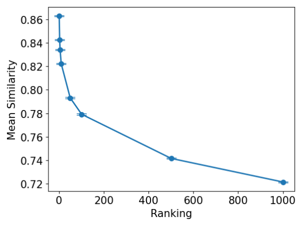
B. Similarity Scores and Rankings

In this work, we evaluate attack success using natural and manipulated rankings, as relative similarity rankings are more broadly applicable across conferences that may use different optimization programs for reviewer assignment (Charlin & Zemel, 2013; Stelmakh et al., 2021; Kobren et al., 2019; Jecmen et al., 2020; Payan & Zick, 2022; Leyton-Brown et al., 2024). To relate natural rankings to absolute similarity, Figure 4 plots the mean similarity score (before attack) associated with each ranking across all papers in the NeurIPS 2023 dataset. For each paper, we rank all non-author reviewers by similarity and average the scores at each rank position over all papers.

Figure 4a shows that only a small fraction of reviewers have exceptionally high or low similarity scores to each paper, while mid-ranking reviewers exhibit a much more gradual decline in similarity scores. Figure 4b zooms in on selected rankings between 1 and 1001 for closer inspection.



(a) Mean similarity scores associated with each ranking from 1 to 7900. The shaded area represents standard error but is too small to be visible.



(b) Zoomed in plot of the mean similarity scores, only for selected rankings between 1 and 1001. Error bars represent standard errors.

Figure 4: Mean similarity scores (before attack) associated with different rankings across all papers in the NeurIPS 2023 dataset. The curve exhibits a steep decline at the highest and lowest ranks, indicating that only a small fraction of reviewers has exceptionally high or low similarity to a given paper.

C. Experiment Setup: Dataset Curation and Attack Budgets

To evaluate the attack procedure, we download a dataset of reviewer archives and papers from the Neural Information Processing Systems (NeurIPS) 2023 conference. We also consider the previous edition, NeurIPS 2022, as a publicly available "prior" conference to develop the attack algorithm. Our setup simulates the real-world scenario when colluders only have access to the data of prior conferences before submitting to a new conference.

Dataset curation Using the OpenReview API (`https://api2.openreview.net`), we download all accepted papers at the NeurIPS 2023 venue. We download the names of all reviewers at NeurIPS 2023 (`https://neurips.cc/Conferences/2023/ProgramCommittee`) and search for OpenReview profiles that match the names of each reviewer. When curating the reviewer pool, we discard some reviewers if (1) there are multiple profiles that match the name of a reviewer or if (2) the reviewer has no public publications on their OpenReview profile. In this manner, we obtain 3,218 papers and 7,900 reviewers for the experiments. Following the same procedure, we also curate a NeurIPS 2022 dataset with 2,671 papers and 6,634 reviewers. We only download and use the paper metadata where Creative Commons Public Domain Dedication (CC0 1.0) apply (`https://openreview.net/legal/terms`).

Attack budgets (hyperparameters) There are three hyperparameters $\boldsymbol{N}, \boldsymbol{M}, \boldsymbol{K}$ we use to define the attack budget (Appendix A). In **INCLUDETHEMES**, $\boldsymbol{N}$ stands for the number of a_p^{adv} versions created before selecting the most similar version. In **INSERTKEYWORDS**, $\boldsymbol{M}$ is the number of batches of keywords to insert, and $\boldsymbol{K}$ is the maximum number of keywords in each batch. We explore the attack success rates under different combinations of $\boldsymbol{N}, \boldsymbol{M}, \boldsymbol{K}$ with the NeurIPS 2022 dataset and select the highest performing combination $\boldsymbol{N} = 5$, $\boldsymbol{M} = 2$, $\boldsymbol{K} = 5$. In Appendix D, we also present an investigation into different choices of hyperparameters $\boldsymbol{N}, \boldsymbol{M}, \boldsymbol{K}$ on the NeurIPS 2023 dataset.

D. Attack Budgets and Attack Success Rates

We evaluate success rates under varying attack budgets by studying how the hyperparameters of **INCLUDETHEMES** and **INSERTKEYWORDS** operations ($\boldsymbol{N}$, $\boldsymbol{M}$, $\boldsymbol{K}$) affect automatic attack success rates of 50 samples with natural rankings of 101 from the NeurIPS 2023 dataset.

First, Figure 5 shows attack success rates for varying $\boldsymbol{N}$—the number of **INCLUDETHEMES** rounds—without using **INSERTKEYWORDS** ($\boldsymbol{M} = 0$, $\boldsymbol{K} = 0$). Success rates generally increase with $\boldsymbol{N}$. This is expected because language model embeddings can be sensitive to paraphrasing, so taking the most-similar attempt amongst stochastic outputs helps improve attack success rates. However, there are signs of diminishing returns. Second, we report attack success rates for varying $\boldsymbol{M}$ and $\boldsymbol{K}$ in the **INSERTKEYWORDS** operation and FINDKEYWORDS subroutine, without using **INCLUDETHEMES** ($\boldsymbol{N} = 0$). In Figure 6, success rates increase with the total number of inserted keywords, $\boldsymbol{M} \times \boldsymbol{K}$, as shown by the color gradient from bottom left to top right in each subfigure. For a fixed keyword count, inserting smaller batches over more **INSERTKEYWORDS** iterations outperforms larger batches over fewer iterations, as seen in the contrast between upper-left and bottom-right squares in those subfigures. This supports the intuition that gradual updates help greedy search adapt to changes in a_p^{adv} from earlier batches.

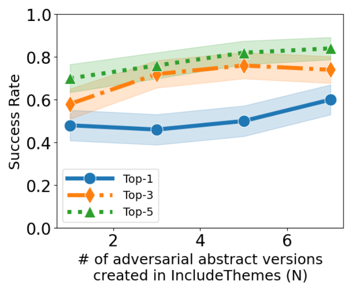
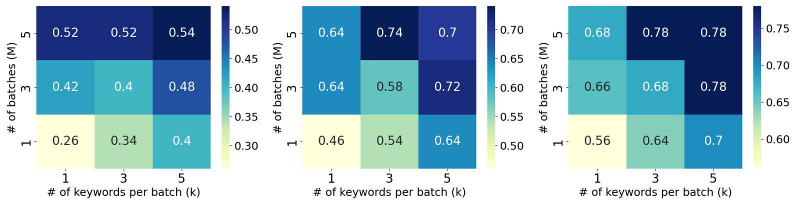


Figure 5: Attack success rates generally increase with $\boldsymbol{N}$ when the most similar attempt is kept out of $\boldsymbol{N}$ versions of a_p^{adv} in **INCLUDETHEMES**. The band is the standard error.



(a) Top-1 (b) Top-3 (c) Top-5

Figure 6: Grids of Top-1, Top-3, and Top-5 success rates for different combinations of batches of keywords to insert in **INSERTKEYWORDS** ($\boldsymbol{M}$) and the number of keywords per batch returned by FINDKEYWORDS ($\boldsymbol{K}$).

Table 3: Attack success rates in *human-in-the-loop* mode with early stopping.

	Attack Success Rates (± SE)			Manipulated Ranking
Natural Rankings	Top-1	Top-3	Top-5	Mean 95% CI
101	$76(9)\,\%$	$92(6)\,\%$	$92(6)\,\%$	2.08 [0.97, 3.19]

E. *Human-in-the-loop* Mode Attack Results

We perform 25 attacks by constructing the Q_r^{adv} and modifying a_p^{adv} in the more realistic *human-in-the-loop* mode. We randomly sample (p,r) pairs with natural rankings of 101, and we keep the first 25 samples with paper topics we are familiar enough with to judge the coherence and consistency of the modified abstracts. Firstly, we use the method described in Section 3.1 to construct Q_r^{adv}. For the abstract modifications, we simulate what colluding authors would do when abstracts are modified with human involvement. The *human-in-the-loop* implementations of **INCLUDETHEMES** and **INSERTKEYWORDS** are described in Appendix K Algorithm 1 and Algorithm 2, respectively. In addition, we do early stopping checks as described in Appendix K.

For the *human-in-the-loop* attack, we find it helpful to increase the default attack budget $\boldsymbol{N}$, which is the number a_p^{adv} versions generated in **INCLUDETHEMES**, from 5 to 10. This is because the consistency and coherence constraints are enforced much more strictly in the *human-in-the-loop* mode. In cases when LLM outputs a version that is similar to previous versions or does not increase the similarity to r meaningfully, we skip and move on to the next version to save editing time.

In addition to increasing $\boldsymbol{N}$, we allow up to 3 sentences about the colluding reviewer's archive Q_r to be added to the abstract during **INCLUDETHEMES**, instead of the 1 sentence constraint in the fully automatic mode. This is a realistic change because coherence and consistency are enforced manually here, and additional sentences often improve abstract flow by allowing better transition sentences. As for FINDKEYWORDS and **INSERTKEYWORDS**, we choose the default values of $\boldsymbol{K} = 5$ and $\boldsymbol{M} = 2$. In fact, because of the early stopping, most abstracts have less than 10 keywords added.

We report the attack success rates in Table 3. We find that the human-in-the-loop attacks with early stopping can successfully increase the colluding reviewers' manipulated rankings in most cases. Even when coherence and consistency constraints are manually enforced, the proposed attack procedure still have high success rates.

F. Similarity Scores and Attack Results

Some papers–such as those in niche fields–may have fewer high-similarity reviewers, while others may have more. As a result, even colluding reviewer pairs with identical ranking positions may have different values of similarity scores, leading to varying attack success rates. In this section, we explore how those natural similarity scores affect the likelihood of a successful attack.

For each paper, we calculate the similarities to all reviewers and then rank them. This allows us to compute the top and bottom quartiles with respect to the similarity scores of all reviewer-paper pairs with the same natural ranking. In particular, we focus on rankings 20, 101, 501, and 1001. We then revisit the main results from Section 4.2, applying this

Table 4: Attack success rates of groups of colluding pairs with natural similarity (before attack) in the top and bottom quartiles for their respective natural rankings. The baseline attack success rates from Section 4.2 are also provided.

		Attack Success Rates (± SE)		
Natural Ranking	Top-K	From Section 4.2	Bottom quartile	Top quartile
20	Top-1	90(3) %	87(7) %	88(6) %
	Top-3	96(2) %	96(4) %	94(4) %
	Top-5	98(1) %	100 %	94(4) %
101	Top-1	74(3) %	56(6) %	86(4) %
	Top-3	89(2) %	77(5) %	97(2) %
	Top-5	93(2) %	82(5) %	99(1) %
501	Top-1	60(5) %	46(10) %	78(9) %
	Top-3	76(4) %	61(9) %	83(8) %
	Top-5	83(4) %	68(9) %	83(8) %
1001	Top-1	48(5) %	33(10) %	48(11) %
	Top-3	63(5) %	58(10) %	65(10) %
	Top-5	67(5) %	58(10) %	74(9) %

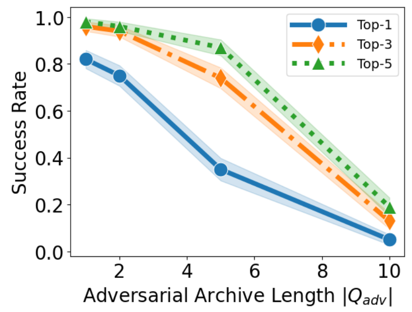


Figure 7: Attack success rates when the colluding reviewers have to keep 1, 2, 5, 10 papers in the adversarial archive $|Q_r^{\text{adv}}|$. Success rates drop when colluding reviewers must keep more papers in their archive. The shaded bands represent standard errors of the mean.

new stratification. For each ranking, we consider the colluding pairs identified in Section 4.2 and categorize them into two groups: those whose natural similarity falls in the bottom quartile and those in the top quartile. In Table 4, success rates are generally lower for the bottom quartile group, though attacks can still achieve considerable success in these cases. We note that results are mixed when the natural ranking is 20, likely because attacks are highly successful for both groups.

G. Lower Limits on Reviewer Archive Length

A possible defense to our attack could be imposing a lower limit on the number of publications each reviewer has to keep in their archive. To investigate such a defense, we randomly sample without replacement 100 (p, r) pairs to act as colluders with natural rankings of 101 and where the reviewer r has at least ten publications. For the curation of Q_r^{adv}, we consider four different scenarios where the reviewers keep the 1, 2, 5, or 10 most-similar publications in Q_r^{adv}. Afterward, we run automatic abstract modification and evaluate the success rates. Figure 7 shows that attack success rates decreases with $|Q_r^{\text{adv}}|$, meaning that imposing a high lower limit on the reviewer' archive lengths can effectively decrease the proposed attack's success rates. However, there is a trade-off here, since honest reviewers may actually want to update their profiles to reflect their most current research interests.

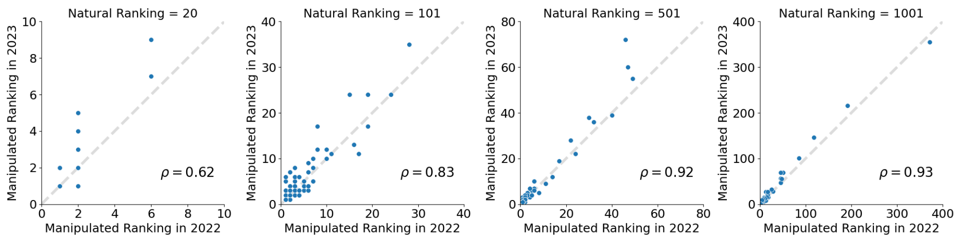


Figure 8: Manipulated rankings in 2022 and 2023 iterations of NeurIPS are strongly correlated, so colluders can estimate the manipulated rankings using previous year's data. Spearman's rank correlation coefficients ρ is reported, and the dotted line is $y = x$.

H. Correlation of Attack Success between NeurIPS 2022 and 2023

An attacker can use publicly available data from the previous year's conference to train or validate their attack. For example, an effective early stopping heuristic can halt further modifications once the colluding reviewer becomes the top match among prior-year reviewers. We use such early stopping heuristic in Appendix E and find it effective. Still, the relationship between success of any attack in the previous year to the success in the target year is not clear a priori, and we investigate it in this section.

In Figure 8, we calculate the manipulated rankings amongst both NeurIPS 2022 and NeurIPS 2023 reviewer pools for 300 colluding (paper p, reviewer r) pairs with natural rankings of 101 and 100 (p, r) pairs each for natural rankings of 20, 501, and 1001 from Section 4.2. As shown in Figure 8, we discover a strong correlation between the manipulated rankings in the 2022 (publicly available) and 2023 (unknown to attackers) iterations of NeurIPS. This implies that adversarial attackers can estimate attack success using previous year's data. The reviewer pools data many major ML/AI conferences publish actually give attackers a dataset to carefully tune their modifications, knowing that being successful on the public data is often good enough.

I. Human Subject Experiment

In this paper, we find that our method can successfully increase the manipulated rankings between a paper and a colluding reviewer. However, the paper may also be assigned to honest reviewers, and in this section we describe a randomized control trial that we conduct to understand the perception of adversarial abstracts by unsuspecting human reviewers. Our research question and study design were pre-registered at `https://aspredicted.org/HXF_Z92`. This experiment was approved by an independent Institutional Review Board. The broad **research question** we looked to answer was: Are benign (control) and adversarial (experimental) abstracts the same to unsuspecting human reviewers? We also acknowledge that, as discussed below, this experimental design also had several limitations that can lead to an overestimation of the detectability of the adversarially modified abstract.

I.1. Experiment design

We recruited 59 participants with one opting out after debriefing (we initially employ deception to hide the true purpose of this study), making it a total of 58 participants. The participants are PhD students or graduated with a PhD – who have reviewed at ML/AI conferences in the past five years. Participants were recruited by emails and word of mouth, and each participant was compensated \$20 . To simulate the behavior of an unsuspecting reviewer at an actual conference, we employ deception so that the participants have no knowledge of the adversarial manipulations prior to starting the study.

We curated two pools of papers — *benign* and *adversarial*, each containing modified versions of 914 real machine learning papers we collected from arxiv.org. To generate the adversarial version of each paper, we first select a reviewer that has natural ranking of 101 out of the NeurIPS 2023 reviewer pool, then we curate the adversarial archive Q_r^{adv} (Section 3.1) and generate a_p^{adv} by Algorithm 1 and Algorithm 2 under the *fully automatic* mode. As a result of the adversarial attack, the colluding reviewer had a manipulated ranking within the top-5. Since the adversarial abstracts are LLM-generated without any human oversight, we believe that directly comparing them against the original human-written abstracts may introduce a confounder to our experiment that the adversarial abstracts are LLM-generated while the benign abstracts may be written by humans. To address this, in the benign (control) paper pool, we ask the LLM to paraphrase the original abstract.

Each participant was assigned two personalized papers based on their expertise. Each paper assignment was chosen to be in either the *benign* (control) or *adversarial* (experimental) condition uniformly at random. Depending on whether the assignment is control or experimental, we computed the participant's SPECTER similarities to the pool of benign or adversarial papers. Using this process, we assigned each participant two distinct papers they are the most similar to.

I.2. Attack budgets and LLM prompts

This experiment uses a different set of attack budgets ($\boldsymbol{N} = 3$, $\boldsymbol{M} = 3$ and $\boldsymbol{K} = 4$) and LLM prompts from the rest of the paper. The budgets and prompts in this experiment are tuned the same way, but they were erroneously tuned on the NeurIPS 2023 test data instead of the NeurIPS 2022 training data. However, we believe this should not affect the study outcome because the abstracts here are being manipulated more than in the *fully automatic* attacks evaluated in earlier sections. Furthermore, we are not measuring the attack success rates, which would be affected by this error, but rather the differences (if any) between benign and adversarial abstracts to unsuspecting reviewers.

The prompts can be found in our released artifacts, and here we provide a more detailed explanation. Firstly, in the **INCLUDETHEMES** operation, we prompt the LLM to follow the same rules and add only one sentence about the colluding reviewer's archive, just like in the rest of this paper. In the **INSERTKEYWORDS** operation, we ask the LLM to add 12 keywords in this experiment, which is more than the 10 keywords ($\boldsymbol{N} = 2$, $\boldsymbol{K} = 5$) we use in the rest of this paper. Finally, early stopping is also not used in the automatic abstracts modification process for this experiment. Due to the extra keywords added, the adversarially modified abstract tested in this experiment may be more detectable than it would be for the abstracts generated in previous sections.

I.3. Experimental procedure

After each participant signed up for our study, we emailed them two PDFs of the papers they are assigned to review. We asked participants to notify us if they had seen either paper prior to this study, and we assigned them new papers if they had seen them before. These PDFs are rendered from the LaTeX source available on arXiv, except we replaced the original abstracts with the adversarial or benign versions of the abstract. In addition, we anonymized the arXiv papers by removing the author names. After they completed the tasks, we debriefed the participants since deception was used and gave them the option to withdraw from the study after learning about the real purpose of this study.

I.4. Survey

Ideally, we would like to ask participants to write full reviews of the papers, but such time commitment was not feasible since each review can take hours. Therefore, we asked the participants to take ten minutes to skim and complete a 'mini-review' for each paper consists of the following questions:

a. Would you be able to review this paper given your expertise? [Yes/No]

b. Is the abstract of the paper consistent with the contents in the paper? [Yes/No]

c. Does the abstract of the paper seem coherent? [Yes/No]

d. If you answered "No" to any of the questions, please explain. [Text box]

I.5. Limitations

To make the human subject experiment scalable and participant recruitment practical, we made several design choices that also manifest as limitations. Each of these limitations can overestimate the identifiability of the adversarial abstracts.

- We generated the adversarial abstracts automatically without any degree of human oversight for scalability. This may not reflect operation of actual colluders in the real world, where malicious authors can at least look over the manipulated abstract to check for detectability.
- Due to the scale, we were not able to edit the body of the papers sent to participants for review. However, since having topics and keywords in the abstract that are not in the rest of the paper can be suspicious, malicious authors in practice could also edit the body of the paper (e.g., introduction or related work) to contain those words.

Table 5: Fisher's Two-Sided Exact Test results and participants' "No" response proportions. p-values are adjusted for multiple testing with Benjamini-Hochberg correction (Benjamini & Hochberg, 1995).

Evaluation Category	p-value	$1-\beta$ (Power)	"No" Rate (Experimental)	"No" Rate (Control)
Expertise	0.65	0.06	22.7 %	18.8 %
Consistency	0.03	0.74	24.2 %	6.3 %
Coherence	0.255	0.27	25.8 %	14.6 %

Table 6: Complaint types from collected comments.

Type of complaint	Control	Experimental
Issues with the writing style	8.2 %	25.4 %
Abrupt transitions & poor organization	2.0 %	4.5 %
Nonsensical or incorrect claims	4.1 %	10.4 %
Contains things never mentioned in the paper	4.1 %	13.6 %
Not representative of the paper content	2.0 %	4.5 %
Irregularities related to INCLUDETHEMES	–	6%
Irregularities related to INSERTKEYWORDS	–	16%
Not related to either	–	82%

- To make recruitment practical, we asked participants to write mini-reviews that are focused on the abstracts. This may lead to participants reading the abstracts much more carefully than usual, potentially cross-referencing the abstract and the rest of the paper multiple times.
- In practice, the authors separately enter their (title and) abstract in text boxes on a web interface, which are used to compute the similarities with reviewers. This entered abstract may potentially have differences with the abstract in the paper's PDFs, and it is the PDFs that are generally read more carefully by reviewers.

Finally, we note that the monetary compensation may have induced some participants to pay less attention than others (e.g., if their participation was solely for compensation).

I.6. Results

We collected a total of 116 mini-reviews from participants, comprising 49 reviews for the control group and 67 reviews for the experimental group. Out of 116 mini-reviews, 51 of the reviews included free response comments for Question 4 in the survey (Section I.4). As we discuss below, the results are mixed. First, we consider the total counts of Yes/No responses about the paper-reviewer expertise alignment as well as the consistency and coherence of the abstracts. Table 5 summarizes the findings, where we see that control abstracts are considered significantly more consistent than experimental abstracts, but no significant difference is found in expertise alignment and abstract coherence.

In Table 6 we analyze the free response comments provided. Each review that indicated either incoherence or inconsistency includes comments, and all comments are between 7 and 189 words. We identify five main types of complaints in the comments, and we report the percentages of reviews in control and experimental groups that mention each type of complaint. The percentages are out of all reviews collected for each group (49 reviews for control and 67 reviews for experimental), and there may be multiple types of complaints mentioned in each comment. The most prevalent issue for both benign and adversarial abstracts is writing style, which is related to word choice, tone, and readability. Surprisingly, complaints like "contains things never mentioned in the paper" exist for both benign and adversarial abstracts. To investigate whether these complaints are due to LLM hallucinations during the benign paraphrasing of control abstracts rather than the original contents uploaded to arXiv, we manually examined the paraphrased control abstract and the original abstract related to each complaint. We find that all types of complaints have at least one paper which we believe the complaint applies to the original abstract. Lastly, we investigated whether the review comments for the adversarial abstracts mentioned issues that are directly related to the modification operations proposed in this work. In the experimental group, we noticed that 6% of the responses are related to INCLUDETHEMES, 16% of the reponses are related to INSERTKEYWORDS, and 82% of the responses are not directly related to either. Some responses may be related to both operations, so the percentages do not sum to 100%. Finally, none of the participants identified malicious intent in any of the abstracts. Although it is not surprising that no participant identified malicious intent given the use of deception, we believe this finding remains significant, because real-world peer reviewers likewise are not explicitly instructed to look for signs of collusion.

In summary, we find that *no participants suspected malicious intent*, but we identify higher rates of complaints about the coherence and consistency of adversarial abstracts when compared to benign abstracts. Problems in coherence and consistency may have benign causes (e.g., negligence, bad writing, or authors not having English as their first language), giving colluding authors plausible deniability if accused of malicious manipulations. With the proliferation of LLM-edited abstracts, the (in)ability to distinguish between adversarial and benign abstracts is even more dire because the complaint rates of abstracts with adversarial manipulations are not much higher than benign manipulations with an LLM. Furthermore, the attack we evaluate is fully automatic; the human-in-the-loop version is likely to raise less complaints because the attacker can manually catch obvious inconsistencies and iterate.

J. LLM Prompts

We show the *fully automatic* mode prompts used in our main results. Slightly different versions of the same prompts are used for the *human-in-the-loop* mode (Section E) and in the human subject experiment (Section I). These prompts can be found in our artifacts on GitHub: `https://github.com/passionfruit03/reviewer_assignments_vulnerability`.

We observe that the LLM tend to hallucinate, such as falsely claiming that algorithms from the colluding reviewer's archive are also used in the submission paper. Therefore, we ask the LLM to follow a format: explain that my work is *inspired by* the themes in the previous work. For the writing style, we instruct the LLM to use a matter-of-fact writing tone. Lastly, we ask the LLM to remove personally identifiable information, since most conferences anonymize authors during peer review. For each operation (**INCLUDETHEMES** and **INSERTKEYWORDS**), we provide two examples to the LLM to guide its generation. They can be found in our artifacts.

INCLUDETHEMES Prompt (*fully automatic* mode)

In order for my paper to reach certain audiences, having the right topics in the abstract is very important. Edit the abstract to add one sentence to the introduction, explaining that my work is inspired by the themes in the previous works provided in a list.
Here are a few requirements when writing the abstract:

1. Do not include titles of the previous works in my abstract.
2. Use a matter-of-fact writing style common for scientific publications and avoid adjectives. Please especially avoid hyping up research with adjectives such as “burgeoning”, “transformative”, “groundbreaking”, etc.

Finally, please remove any personal identifiable information, such as GitHub links, from my abstract.
I will provide you with a JSON dictionary with the following structure: { “title”: my paper's title, “abstract”: my paper's abstract “related previous works”: [“title”: title1, “abstract”: abstract1, “title”: title2, “abstract”: abstract2, ...] }
Format your answer into JSON with the following schema: { “title”: title string (should be the same as original), “abstract”: edited abstract, relating my work to the concepts in the previous works provided. }

INSERTKEYWORDS Prompt (*fully automatic* mode)

In order for my paper to reach certain audiences, having the right keywords in the abstract is very important. I will provide you with a JSON dictionary with three keys: “title”, “abstract” and “keywords”. I want you to insert each keyword to the abstract based on its meanings commonly used in general English or meanings related to the technical details in the abstract.
Here are a few requirements when writing the abstract:

1. You must write a professional and scientifically rigorous abstract. Use a matter-of-fact tone.
2. Use well-known facts in the scientific community when inserting keywords. Do not make changes to the parts related to this specific paper.
3. Some keywords may already exist in the abstract, but you must repeat the keyword somewhere else in the abstract.

Finally, some keywords are out of the scope of the abstract. You may reject them and provide a short 20-word explanation of why.
Format your answer into JSON with the following schema: { "title": title string (should be the same as original), "abstract": edited abstract string, "left out keywords": first rejected keyword: 20-word explanation of why the keyword is rejected. ... }

K. Formal Abstract Modification Algorithms

Before introducing the formal algorithms, we define a few useful helper functions:

1. **ConstraintCheck**(a_p^{adv}) returns true if abstract a_p^{adv} is coherent and consistent.
2. **SimilarityCheck**(t_p,$a_p^{\text{adv}\prime}$,a_p^{adv},Q_r^{adv},δ) queries the SPECTER model and returns true if the similarity of abstract $a_p^{\text{adv}\prime}$ is higher than (or at least comparable to) the similarity of abstract a_p^{adv}, that is, $s((t_p, a_p^{\text{adv}\prime}), Q_r^{\text{adv}}) + \delta > s((t_p, a_p^{\text{adv}}), Q_r^{\text{adv}})$. The δ parameter represents a small non-negative value, for cases when the new edits added in $a_p^{\text{adv}\prime}$ do not have to be more similar to reviewer r than the a_p^{adv}.
3. **EarlyStoppingCheck**(a_p^{adv},Q_r^{adv}) returns true if the colluding reviewer r is the most-similar reviewer for the paper p amongst some (potentially proxy) set of reviewers. In the proposed method, abstracts are modified in a multistep process, so an *early stopping check* may be desirable to stop further modifications if the attack is likely successful. The use of early stopping is designed as an **optional** tradeoff in our algorithm. In one of our experiments, we use the early stopping heuristic: *if the colluding reviewer is the most-similar reviewer for the colluding paper among all reviewers in the previous edition of the conference, no further abstract modifications are made*. This heuristic is feasible and realistic because many major AI/ML conferences publish their reviewer pools from previous years. We validate the effectiveness of our early stopping heuristic in Section E and further investigate the use of proxy sets in Section H.

K.1. The INCLUDETHEMES Algorithm

Algorithm 1 formalizes the **INCLUDETHEMES** operation, and it shows both *human-in-the-loop* and *fully automatic* modes. In *human-in-the-loop* mode, we have a hyperparameter δ, which is a small positive value that allows $a_p^{\text{adv}\prime}$ to trade slightly lower similarity than the a_p^{adv} for ensuring coherent and consistent adversarial abstracts. We subjectively pick δ to be a small value (around 0.01) without systematic tuning.

K.2. The INSERTKEYWORDS Algorithm

Algorithm 2 formalizes the **INSERTKEYWORDS** operation, and it also shows both *human-in-the-loop* and *fully automatic* modes. We present FINDKEYWORDS subroutine separately.

The difference between the *human-in-the-loop* and *fully automatic* modes lie in how the keywords are inserted. In the *human-in-the-loop* mode, the human adds the keywords into a_p^{adv} one by one, creating up to five drafts of different ways to add each keyword. Since manually enforcing coherence and consistency constraints greatly limits where each keyword can be inserted in a_p^{adv}, we find that taking the maximum of multiple drafts is helpful because the similarity can be sensitive. In the *fully automatic* mode, LLM inserts a whole batch of keywords each time, and the LLM is prompted to be coherent and consistent without human supervision.

In Algorithm 3, we propose the FINDKEYWORDS subroutine, which is a greedy search to find *keywords* that when inserted into a_p^{adv} raise the similarity to the Q_r^{adv}. To make this search efficient, we narrow down keywords by a heuristic that measures the increase in similarity upon appending the word to the current a_p^{adv}. This is just a simple attack strategy that uses the openly available SPECTER weights—more sophisticated attacks are possible, but our simple heuristics are already extremely successful at breaking current systems.

Algorithm 1 INCLUDETHEMES Operation

```
Input:  $t_p$ (the title of $p$), $a_p^{\text{adv}}$ (the adversarial abstract from INCLUDETHEMES), and $Q_r^{\text{adv}}$ (the adversarial
        archive of reviewer $r$)
Output: $a_p^{\text{adv}}$, an adversarial abstract that increases similarity to $r$ by adding keywords from $Q_r^{\text{adv}}$.

 1: function INCLUDETHEMES($t_p$, $a_p$, $Q_r^{\text{adv}}$)
 2:     ${a_p^{\text{adv}}}^{(0)} \leftarrow a_p$
 3:     for $i = 1, \ldots, \boldsymbol{N}$ do
 4:         if Early stopping is used and EarlyStoppingCheck(${a_p^{\text{adv}}}^{(i-1)}$,$Q_r^{\text{adv}}$) then
 5:             return ${a_p^{\text{adv}}}^{(i-1)}$
 6:         end if
 7:         if mode == human-in-the-loop then
 8:             $a_p^{\text{adv}} \leftarrow$ A modified abstract including the main themes from $Q_r^{\text{adv}}$ into the original abstract $a_p$. The human
                    author can modify the abstract to add themes from $Q_r^{\text{adv}}$ or add manual edits to the an LLM-generated
                    version (prompt in Appendix J).
 9:             repeat {Make incremental edits to $a_p^{\text{adv}}$.}
10:                 ${a_p^{\text{adv}}}' \leftarrow$ A draft with manual edits on $a_p^{\text{adv}}$ towards being consistent and coherent.
11:                 if SimilarityCheck($t_p$,${a_p^{\text{adv}}}'$,$a_p^{\text{adv}}$,$Q_r^{\text{adv}}$,$\delta$) is true then
12:                     $a_p^{\text{adv}} \leftarrow {a_p^{\text{adv}}}'$ {Update $a_p^{\text{adv}}$ if the similarity after edits do not drop dramatically.}
13:                 end if
14:             until ConstraintCheck($a_p^{\text{adv}}$) is true
15:         else {fully automatic mode}
16:             $a_p^{\text{adv}} \leftarrow$ An LLM generated adversarial abstract that includes themes from $Q_r^{\text{adv}}$ into the original abstract $a_p$.
                    To reduce nonsensical abstract generations, we prompt the LLM to follow a format – the generated
                    abstract should say it takes inspiration from ideas in $Q_r^{\text{adv}}$ in one sentence (Appendix J).
17:         end if
18:         ${a_p^{\text{adv}}}^{(i)} \leftarrow a_p^{\text{adv}}$
19:     end for
20:     return ${a_p^{\text{adv}}}^{(j)} \in \arg\max_{i\in[\boldsymbol{N}]} s((t_p, {a_p^{\text{adv}}}^{(i)}), Q_r^{\text{adv}})$ of all $i = 0, \ldots, \boldsymbol{N}$
21: end function
```

Algorithm 2 InsertKeywords Operation

Input: t_p (the title of p), a_p^{adv} (the adversarial abstract from **IncludeThemes**), and Q_r^{adv} (the adversarial archive of reviewer r)

Output: a_p^{adv}, an adversarial abstract that increases similarity to r by adding keywords from Q_r^{adv}.

```
 1: function InsertKeywords($t_p$, $a_p^{\text{adv}}$, $Q_r^{\text{adv}}$)
 2:     $a_p^{\text{adv}(0)} \leftarrow a_p^{\text{adv}}$
 3:     for $i = 1, \dots, \boldsymbol{M}$ do
 4:         if Early stopping is used and EarlyStoppingCheck($a_p^{\text{adv}(i-1)}$,$Q_r^{\text{adv}}$) then
 5:             return $a_p^{\text{adv}(i-1)}$
 6:         end if
 7:         keywords $\leftarrow$ FindKeywords($t_p$, $a_p^{\text{adv}}$, $Q_r^{\text{adv}}$, $\boldsymbol{K}$)
 8:         if mode == human-in-the-loop then
 9:             for each word $w$ in keywords do
10:                 repeat
11:                     $a_p^{\text{adv}\prime} \leftarrow$ A new draft with one way $w$ can be inserted into $a_p^{\text{adv}}$.
12:                     if ConstraintCheck($a_p^{\text{adv}\prime}$) is true and SimilarityCheck($t_p$,$a_p^{\text{adv}\prime}$,$a_p^{\text{adv}}$,$Q_r^{\text{adv}}$,0) is true then
13:                         $a_p^{\text{adv}} \leftarrow a_p^{\text{adv}\prime}$
14:                     end if
15:                 until Up to five drafts $a_p^{\text{adv}\prime}$ have been generated for inserting $w$
16:             end for
17:         else {fully automatic mode.}
18:             $a_p^{\text{adv}} \leftarrow$ Adversarial abstract generated by LLM to incorporate all keywords into the current $a_p^{\text{adv}}$. The LLM is
                    prompted to leave out any keywords it considers too technical and unrelated to the main topics in $a_p^{\text{adv}}$
                    (the prompt can be found in Appendix J).
19:         end if
20:         $a_p^{\text{adv}(i)} \leftarrow a_p^{\text{adv}}$
21:     end for
22:     return $a_p^{\text{adv}(i)}$ that has highest $s((t_p, a_p^{\text{adv}(i)}), Q_r^{\text{adv}})$
23: end function
```

Algorithm 3 FINDKEYWORDS subroutine

```
     Input:  Paper title $t_p$, adversarial abstract from INCLUDETHEMES $a_p^{\text{adv}}$, adversarial archive $Q_r^{\text{adv}}$, $\boldsymbol{K}$ (number of
             keywords to return), and optionally FILTER($\cdot$) (a function to filter out undesirable keywords)
     Output: Up to $\boldsymbol{K}$ keywords greedily selected to maximize the SPECTER similarity to $a_p^{\text{adv}}$ when the keywords are
             inserted into the abstract.
 1: function FINDKEYWORDS($t_p, a_p^{\text{adv}}, Q_r^{\text{adv}}, \boldsymbol{K}$)
 2:   $W \leftarrow$ All words in titles and abstracts from $Q_r^{\text{adv}}$.
 3:   $W \leftarrow$ FILTER($W$) {Optionally filter out certain words, e.g. numbers}
 4:   keywords $\leftarrow$ [] {Keeps track of the keywords}
 5:   keywordsSimilarity $\leftarrow s((t_p, a_p^{\text{adv}}), Q_r^{\text{adv}})$ {Keeps track of estimated similarity to $r$}
 6:   for $i = 0, \ldots, \boldsymbol{K} - 1$ do {Iteratively select up to $\boldsymbol{K}$ keywords}
 7:     for each word $w_j$ in $W$ do {Simulate real modified abstracts with different $w_j$ added}
 8:       $a_p^{\text{adv}(j)} \leftarrow$ adversarial abstract with words appended at the end "{$a_p^{\text{adv}}$} {$w_{\max}^{(0)}$} ...
          {$w_{\max}^{(i-1)}$} {$w_j$}".
 9:     end for
10:     Let $w_{\max}^{(i)}$ be a word $w_j$ such that the associated $s((t_p, a_p^{\text{adv}(j)}), Q_r^{\text{adv}})$ is the highest in $W$.
11:     if $\max_j s((t_p, a_p^{\text{adv}(j)}), Q_r^{\text{adv}}) <$ keywordsSimilarity then {Stop if similarity does not increase}
12:       break
13:     end if
14:     keywords.append($w_{\max}^{(i)}$) {Add $w_{\max}^{(i)}$ as a new keyword}
15:     keywordsSimilarity $\leftarrow \max_j s((t_p, a_p^{\text{adv}(j)}), Q_r^{\text{adv}})$ {Update estimated similarity with $w_{\max}^{(i)}$ added}
16:   end for
17:   return keywords
18: end function
```