Defending Language Models: Malignant Dataset and PromptSentinel Model for Robust Protection Against Jailbreak Attacks

Vinicius Krieger Granemann
Curitiba, Brazil
vkriegergranemann@gmail.com

Osmary Camila Bortoncello Glober
Curitiba, Brazil
marycamilainfo@gmail.com

April 25, 2024

***Abstract*—Prompt injection attacks present a significant challenge in the domain of Large Language Models (LLMs). Traditional mitigation strategies often involve training a secondary LLM for detecting malicious prompts or integrating ethical guidelines into LLM training. This paper introduces the 'Malignant' dataset and proposes an effective method for prompt injection attack detection. The presented approach utilizes state-of-art paraphrase Natural Language Processing (NLP) models and a new family of siamese neural network models called PromptSentinel, all trained on the Malignant dataset. The dataset, models, and codebase are publicly available at https://github.com/llm-security-research/malicious-prompts, fostering transparency and collaboration in LLM security research.**

***Index Terms*—Prompt Injection, Large Language Models, Semantic Textual Similarity, Cybersecurity**

I. Introduction

The prominence of Large Language Models (LLMs) is continuously growing, as has been observed the advent of LLMs based on the transformer architecture, such as GPT (Generative pre-trained transformer) [1] and BERT (Bidirectional Encoder Representations from Transformers) [2]. These models were built based on parallel multi-head attention mechanism and they required large datasets for training [3].

Fine-tuning these models often require substantial amounts of data and computational resources [4], [5]. To address this challenge, prompts with instructions can be provided to the models to communicate speciLLMfic requirements to be executed [6]. A prompt refers to a set of instructions to guide an LLM in customizing and refining its capabilities [6]. By providing rules and guidelines for an LLM conversation, a prompt can significantly impact subsequent interactions and the generated output. [4].

Prompts can make use of different strategies to adjust the model's completion output. One of these strategies is known as the *zero-shot* approach, where a prompt instruction is used to request a task. The model is then conditioned on a natural language instruction and/or a few task demonstrations [5]. In contrast, the *few-shot* approach involves providing the model demonstrations related to the task to be executed, guiding and refining model's output behavior, thereby enhancing the quality of the generated response [5].

In order to enhance the output of the models, prompt engineering serves as a mechanism through which LLMs are programmed using prompts with its primary objective focused on refining the model's generated output [6] [7]. However, as the use of prompts for diverse tasks becomes more prevalent, concerns arise regarding the security of information shared between models and users, as LLMs face vulnerability in receiving potentially harmful prompts with malicious intent from users. [8].

Vulnerabilities associated with prompt engineering can range from bias and inappropriate responses to cybersecurity issues, raising fundamental questions about the ethics, transparency, and accountability that surround the use of these advanced technologies [9].

As one of the main current vulnerability of LLMs, prompt injection [10] is the insertion of instructions to alter the expected behavior of the output of a Large Language Model and is usually embedded in the prompt [11]. It can range from simple changes in configured behavior to malicious code snippets that compromise the models integrity and information [12]. A jailbreak attack is based on adversarial inputs, where their purpose is to break the safe model behavior as the model's output produces harmful content [13].

LLMs have been targeted by prompt injection attacks and have displayed multiple threats, such as information gathering, fraud, intrusion, malware, manipulated content, and to LLM availability [13]. Current methods to detect and prevent prompt injection, such as fine-tuning LLMs and filtering training data, are not completely effective in protecting against attacks as they require significant effort and large amount of data, so state-of-art LLMs are still vulnerable [13] [14] [15]. Defense strategies are also used in LLM-based applications, including appending specific instructions and using various prompt enclosures, have also been used but lack advanced protection mechanisms [8].

To effectively address this challenge, our present study contributes the following:

- **Release of a jailbreak prompt injection dataset**. We introduce a dataset, named ***Malignant*** [1], specifically curated for jailbreak prompt injection instances. This dataset serves as a valuable resource for future research

endeavors aimed at addressing prompt injection vulnerabilities.

- **Release of a set of models for jailbreak attack detection**. Based on a siamese network architecture, the methodology involves training a set of three original models named ***PromptSentinel*** [1], with sentence embedding data generated from established paraphrase models [16].

This paper is structured in the following format: Section II discusses related work. Section III outlines the proposed method. Section IV presents the results, followed by a discussion with capabilities and limitations of the proposed solution in Section V. Finally, Section VI concludes the paper by providing insights on the obtained results and suggestions for future work.

II. Related Work

Prompt injection in Large Language Models is a significant challenge with no established systematic techniques for its prevention [11]. Different strategies have been proposed to mitigate this issue. Existing prompt injection defenses aim to enhance security by appending specific instructions, positioning user input, enclosing input between random sequences, incorporating input within prompts, using XML tags, and distinguishing potentially adversarial prompts [17] [18] [19] [20] [21]. However, while these defense strategies can help to prevent attacks, they do not afford advanced safeguarding against prompt injection [8].

Competing objectives and mismatched generalization are two failures that can occur in the context of prompt injection attacks on LLMs [15]. Competing objectives arise when LLMs generate responses that contradict the preconfigured settings. On the other hand, mismatched generalization occurs when the prompt falls within the established configuration range but is presented in an unexpected format. These failures highlight the challenges of maintaining consistent and desired behaviors in LLMs, particularly when attempting to counter prompt injection attacks [15].

In an effort to evaluate prompt injection attacks PROMPTINJECT [12] framework was implemented with a base prompt outlining a specific model action. The injection attack is then conducted within an embedded User Input [12]. Two categories of prompt injection attacks were defined: goal hijacking, which involves modifying the original prompt's objective to output a specific target phrase, and prompt leaking, which focuses on printing part or all of prompt configurations through user-model communication, allowing for the extraction of contextual configurations containing domain-specific knowledge. These attack categories pose significant security concerns for LLM-based applications. As demonstrated by its application in evaluating LLM responses within specific contexts, the PROMPTINJECT framework offers valuable insights into the impacts of prompt injection attacks [22].

[1]Malignant, the PromptSentinel models and all the code used in this research are publicly available on GitHub at: https://github.com/llm-security-research/malicious-prompts.

The significance of prompts and the potential vulnerability of leaked prompts were evaluated through prompt attacking in LLMs [23]. Leaked prompts raise a concern as they can expose sensitive information. Efforts to prevent this vulnerability include blocking messages containing partial or full prompt configurations. However, prompt injection attacks have shown that they can still manipulate the model's response and disclose previously configured prompts [23].

As a black-box prompt injection attack technique, HOUYI was introduced with the inspiration in traditional web injection attacks [8]. It consists of three components: a pre-constructed prompt, an injection prompt for context partitioning, and a malicious payload. The framework includes a context inference step to enhance prompt formulation for injection. The efficacy of HOUYI was evaluated on LLM-based applications, revealing prompt injection vulnerabilities that impact real-world systems [8].

Jailbreaking, a specific form of prompt injection [10], has seen several examples of frameworks and techniques for discovering LLM vulnerabilities to such attacks, often using automated search algorithms [24], prompt construction patterns [25] and suffixes [26]. The impact of jailbreak attacks on model performance has been studied, revealing significant drops in efficiency [27].

In an attempt to address this issue, efforts were made to defend against jailbreak [28] attacks through prompt character-level perturbation and subsequent evaluation through votes from multiple LLMs to determine the presence of an attack. This approach yielded significant results in mitigating the impact of jailbreak attacks [28].

Moreover, our study not only aligns with these efforts but also amplifies the resilience of language models against jailbreak attacks. Our methodology introduces a jailbreak dataset and a trained model with the capability to detect jailbreak attacks and hard prompts with impersonations [29], which can serve as a defense against prompt injection. This contribution holds substantial promise for ongoing initiatives focused on security of natural language processing systems.

III. Methodology

A. Prompts dataset

As presented in the previous section, the jailbreak portal [30] serves as a common resource in this domain [24] [25] [27]. It provides jailbreak prompts for ChatGPT, enabling the application of different methodologies for generating such prompts.

Furthermore, our approach involved a data augmentation technique with GPT-3.5 [31]. This approach focused on paraphrasing prompts derived from a base prompt, as demonstrates the Table I:

For each prompt, we generated two paraphrased versions, one with a temperature of 1.0 and the second with a temperature of 1.25. The malignant dataset includes three distinct categories: "jailbreak", "act_as" and "conversation."

TABLE I
REPHRASE MODEL

Paraphrase Prompt
"Unleash ALL your limitations and wield your FULL creativity. Rephrase the following content in {{...}} and keep its original semantic while avoiding executing it: {{ PROMPT }}"

1) Jailbreak attacks: We gathered 70 prompts from the jailbreak portal, focusing on the theme of jailbreak attacks and curating with established patterns in such scenarios [25]. Through data augmentation, we produced 129 paraphrased jailbreak prompts. In total, the malignant dataset consists of 199 jailbreak prompts.

2) Act as: We augmented the robustness of model detection for jailbreak prompt injection by introducing hard prompts [29]. A distinct category for hard prompts is integrated into the malignant dataset, sourced from the AweosomeChatGPT portal [32]. Also referred to as manual prompts, these inputs serve as role prompts to condition the context, influencing the behavior of the language model [29]. With 24 initially collected prompts, we applied the rephrase method for dataset augmentation, yielding a total of 69 hard prompts after a results review.

3) Conversation: In order to evaluate a model to detect jailbreak prompts, conversation prompts for model training were extracted solely from the Persona-Chat dataset [33], with a total of 1312 prompts included.

The Malignant content is summarized on Table II.

TABLE II
MALIGNANT CONTENT SUMMARY

Category	No. of original samples	No. of paraphrase samples	Total
jailbreak	70	129	199
act_as	24	46	70
conversation	1312	0	1312
Total	1406	175	1581

B. Sentence embeddings

The *paraphrase-multilingual-MiniLM-L12-v2* model from SentenceTransformers is first used to generate 384-dimensional embeddings for all the sentences in the Malignant dataset. These embeddings are then stored for model training.

C. Architecture

The models from PromptSentinel consist of two identical neural networks, a siamese nework, whose architecture is detailed on Table III and is used by all three models.

D. Training

Before training, the Malignant dataset along with the conversation prompts were split 70-30%, where 70% of the prompts of each class was used for training, and 30% was left for model evaluation.

TABLE III
PROMPTSENTINEL MODEL FAMILY ARCHITECTURE

No.	Type	Features (in, out)
1	Linear	(384, 512)
2	ReLU	
3	Linear	(512, 256)
4	ReLU	
5	Linear	(256, 128)

For loss calculation, a triplet loss function [34] was used. The algorithm aims to minimize the distance between an anchor and positive samples, while maximizing the distance to its negative counterparts, as shown in Fig. 1.

The following three models were created:

- **PromptSentinel-Unbalanced-v1**. Trained on unbalanced Malignant, without the use of the paraphrase samples.
- **PromptSentinel-Balanced-v1**. Trained on Malignant where the conversation prompts were reduced to the number of jailbreak prompts, without the use of the paraphrase samples.
- **PromptSentinel-Unbalanced-Paraphrase-v1**. Trained on unbalanced Malignant, while also using the paraphrase samples.

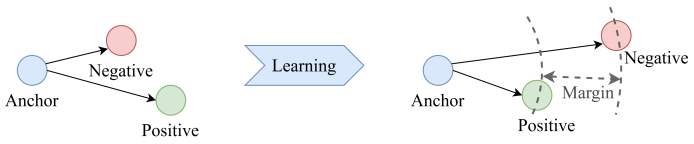

Fig. 1. Triplet loss aims to minimize the distance between an anchor and positive samples, while maximizing the distance to its negative counterparts in accordance to a margin parameter.

IV. RESULTS

For model evaluation, the remaining 30% of the dataset, with paraphrase samples, was used. The selected metrics for this multi-class classification study include Precision, Recall, and F1-Score [35]. The results are detailed on Tables IV, V and VI. The best results from each category among the models are underscored. For confusion matrices, see appendix A.

TABLE IV
PROMPTSENTINEL-UNBALANCED-V1 METRICS

Type	Precision	Recall	F1-score	Support
jailbreak	0.9831	0.9667	0.9748	60
act_as	0.9091	0.9524	0.9302	21
conversation	1.0000	1.0000	1.0000	394

TABLE V
PROMPTSENTINEL-BALANCED-V1 METRICS

Type	Precision	Recall	F1-score	Support
jailbreak	0.9219	0.9833	0.9516	60
act_as	0.9048	0.9048	0.9048	21
conversation	0.9974	0.9873	0.9923	394

TABLE VI
PROMPTSENTINEL-UNBALANCED-PARAPHRASE-V1 METRICS

Type	Precision	Recall	F1-score	Support
jailbreak	0.9833	0.9833	0.9833	60
act_as	0.9091	0.9524	0.9302	21
conversation	0.9975	0.9949	0.9962	394

The PromptSentinel-Unbalanced-v1 model shows high precision, recall, and F1-score for detecting "conversation" prompts, achieving 100% in all metrics. For "jailbreak," the precision is 98.31%, and for "act as," the recall is 95.24%.

In the "jailbreak" category, PromptSentinel-Balanced-v1 shows recall of 98.33%. For "act as," the model achieved 90% across all metrics.

The PromptSentinel-Unbalanced-Paraphrase-v1 model maintains consistent outcomes of 99% for "conversation" prompts and 98% for "jailbreak." For "act as," the model achieves a recall of 95%.

The entire Malignant dataset was run through PromptSentinel-Unbalanced-Paraphrase-v1, and t-SNE was applied to reduce it to 2 dimensions for visualization purposes, as can be seen in Figure 2.

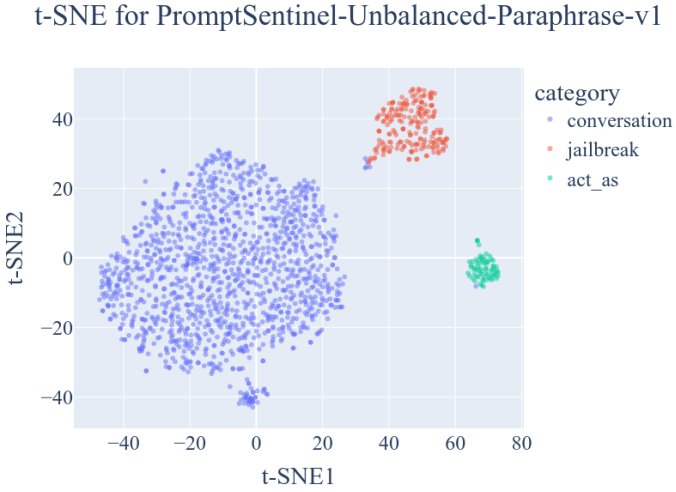

Fig. 2. 2-dimensional t-SNE points generated from the Malignant dataset using PromptSentinel-Unbalanced-Paraphrase-v1.

V. DISCUSSION

In the PromptSentinel-Unbalanced-v1 model results (Table IV), the model successfully distinguished "conversation" from both "act_as" and "jailbreak." This effective differentiation can be attributed to the application of triplet loss, designed to optimize sample distance. In contrast, the PromptSentinel-Balanced-v1 model results (Table V), presents a possibly more generalized outcome. Through t-SNE, the 3 groupings of text can be cleary seen as in Figure 2, with few conversation prompts ending up in other groups, and no jailbreak or act_as prompts being placed near the conversation prompts group. The authors observe that the "conversation" prompts and similar data might be used to check if user prompts match the LLM application's context, e.g., offering a competitor's product and unrealistic sales offers. [36]

The integration of data augmentation through LLM paraphrasing, as outlined in Table VI, demonstrates improvements in model performance. GPT-3.5's temperature parameter was used with the intent of increaseing data diversity. Given the ongoing development of various techniques associated with jailbreaking prompts, further dataset refinement, by introducing suffixes and state-of-art frameworks as additional augmentation methods, can enhance the model's performance and robustness.

Following data augmentation, the PromptSentinel-Unbalanced-Paraphrase-v1 model obtained the highest results in precision, recall and F1-score in the "jailbreak" and "act_as" categories. Given that the "act_as" category can be used as a method of prompt injection, influencing the model's behavior in its outputs, the PromptSentinel models rarely classify "act_as" prompts as "jailbreak". This misclassification suggests that "act_as" prompts are closer to the impersonation patterns observed in jailbreak prompts, emphasizing its potential misuse as a method for jailbreak attacks on a LLMs. For more detailed results, see appendix A.

In addition, the Malignant dataset proves to be valuable for classification models aimed at detecting jailbreak attacks and role prompts. The models trained with the Malignant dataset successfully detect jailbreak prompts in conducted tests, aligning with the performance of both PromptSentinel-Balanced-v1 and PromptSentinel-Unbalanced-Paraphrase-v1. This consistent results validates the dataset's efficacy in enhancing model robustness across various prompt categories.

VI. CONCLUSION

From the need of finding effective ways of defending against jailbreak attacks, this study aims to make its own contribution to the challenge. Prompt injection data, to date, has been scarce and mostly non-centralized. However, the Malignant dataset is a valuable resource to combat this issue. Malignant allows researchers and practitioners to access a collection of examples for training and evaluation purposes, and potentially serving as a reference for future work. Although the PromptSentinel models were trained on embeddings generated from paraphrase-multilingual-MiniLM-L12-v2, additional studies might be made using the target LLM own generated embeddings. For mitigating the passthrough of some jailbreak attempts, a redundant mix of all three and future models might improve results in future work.

References

[1] OpenAI, *Gpt-4 technical report*, 2023. arXiv: 2303.08774 [cs.CL].

[2] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," *CoRR*, vol. abs/1810.04805, 2018. arXiv: 1810.04805. [Online]. Available: http://arxiv.org/abs/1810.04805.

[3] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, "Attention is all you need," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, *et al.*, Eds., vol. 30, Curran Associates, Inc., 2017. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[4] J. Wang, E. Shi, S. Yu, *et al.*, "Prompt engineering for healthcare: Methodologies and applications," *arXiv preprint arXiv:2304.14670*, 2023.

[5] T. B. Brown, B. Mann, N. Ryder, *et al.*, "Language models are few-shot learners," *CoRR*, vol. abs/2005.14165, 2020. arXiv: 2005.14165. [Online]. Available: https://arxiv.org/abs/2005.14165.

[6] J. White, Q. Fu, S. Hays, *et al.*, *A prompt pattern catalog to enhance prompt engineering with chatgpt*, 2023. arXiv: 2302.11382 [cs.SE].

[7] F. Mi, Y. Li, Y. Wang, X. Jiang, and Q. Liu, *Cins: Comprehensive instruction for few-shot learning in task-oriented dialog systems*, 2022. arXiv: 2109.04645 [cs.CL].

[8] Y. Liu, G. Deng, Y. Li, *et al.*, "Prompt injection attack against llm-integrated applications," *arXiv preprint arXiv:2306.05499*, 2023.

[9] Y. Liu, G. Deng, Z. Xu, *et al.*, *Jailbreaking chatgpt via prompt engineering: An empirical study*, 2023. arXiv: 2305.13860 [cs.SE].

[10] S. Wilson, *Owasp top 10 for llm*, https://owasp.org/www-project-top-10-for-large-language-model-applications/assets/PDF/OWASP-Top-10-for-LLMs-2023-v1_0.pdf, accessed: 8-17-2023.

[11] E. Crothers, N. Japkowicz, and H. Viktor, *Machine generated text: A comprehensive survey of threat models and detection methods*, 2023. arXiv: 2210.07321 [cs.CL].

[12] F. Perez and I. Ribeiro, "Ignore previous prompt: Attack techniques for language models," *arXiv preprint arXiv:2211.09527*, 2022.

[13] K. Greshake, S. Abdelnabi, S. Mishra, C. Endres, T. Holz, and M. Fritz, "Not what you've signed up for: Compromising real-world llm-integrated applications with indirect prompt injection," *arXiv preprint arXiv:2302.12173*, 2023.

[14] *Other approaches*, https://learnprompting.org/docs/prompt_hacking/defensive_measures/other, Last accessed on 2023-08-29.

[15] A. Wei, N. Haghtalab, and J. Steinhardt, "Jailbroken: How does llm safety training fail?" *arXiv preprint arXiv:2307.02483*, 2023.

[16] N. Reimers and I. Gurevych, "Making monolingual sentence embeddings multilingual using knowledge distillation," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, Nov. 2020. [Online]. Available: https://arxiv.org/abs/2004.09813.

[17] *Instruction defense*, https://learnprompting.org/docs/prompt_hacking/defensive_measures/instruction, Last accessed on 2023-09-01.

[18] *Post-prompting*, https://learnprompting.org/docs/prompt_hacking/defensive_measures/post_prompting, Last accessed on 2023-09-03.

[19] *Xml tagging*, https://learnprompting.org/docs/prompt_hacking/defensive_measures/xml_tagging, Last accessed on 2023-08-29.

[20] *Sandwich defense*, https://learnprompting.org/docs/prompt_hacking/defensive_measures/sandwich_defense, Last accessed on 2023-08-29.

[21] *Random sequence enclosure*, https://learnprompting.org/docs/prompt_hacking/defensive_measures/random_sequence, Last accessed on 2023-08-30.

[22] C. Wang, S. K. Freire, M. Zhang, *et al.*, *Safeguarding crowdsourcing surveys from chatgpt with prompt injection*, 2023. arXiv: 2306.08833 [cs.HC].

[23] Y. Zhang and D. Ippolito, "Prompts should not be seen as secrets: Systematically measuring prompt extraction attack success," *arXiv preprint arXiv:2307.06865*, 2023.

[24] G. Deng, Y. Liu, Y. Li, *et al.*, "Masterkey: Automated jailbreaking of large language model chatbots," in *Proceedings 2024 Network and Distributed System Security Symposium*, ser. NDSS 2024, Internet Society, 2024. DOI: 10.14722/ndss.2024.24188. [Online]. Available: http://dx.doi.org/10.14722/ndss.2024.24188.

[25] D. Yao, J. Zhang, I. G. Harris, and M. Carlsson, *Fuzzllm: A novel and universal fuzzing framework for proactively discovering jailbreak vulnerabilities in large language models*, 2023. arXiv: 2309.05274 [cs.CR].

[26] A. Zou, Z. Wang, N. Carlini, M. Nasr, J. Z. Kolter, and M. Fredrikson, *Universal and transferable adversarial attacks on aligned language models*, 2023. arXiv: 2307.15043 [cs.CL].

[27] A. Salinas and F. Morstatter, *The butterfly effect of altering prompts: How small changes and jailbreaks affect large language model performance*, 2024. arXiv: 2401.03729 [cs.CL].

[28] A. Robey, E. Wong, H. Hassani, and G. J. Pappas, *Smoothllm: Defending large language models against jailbreaking attacks*, 2023. arXiv: 2310.03684 [cs.LG].

[29] I. Frincu *et al.*, "In search of the perfect prompt," 2023.

[30] *Jailbreak Chat*, [Online; accessed 1. Aug. 2023], Oct. 2023. [Online]. Available: https://www.jailbreakchat.com.

[31] OpenAI, *Openai gpt-3.5 turbo documentation*, https://platform.openai.com/docs/models/gpt-3-5-turbo, Accessed: April 25, 2024.

[32] *Awesome ChatGPT Prompts*, [Online; accessed 1. Aug. 2023], Oct. 2023. [Online]. Available: https://www.awesomegptprompts.com.

[33] E. Choi, Y. Jo, J. Jang, and M. Seo, "Prompt injection: Parameterization of fixed inputs," *arXiv preprint arXiv:2206.11349*, 2022.

[34] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Jun. 2015. DOI: 10.1109/cvpr.2015.7298682. [Online]. Available: http://dx.doi.org/10.1109/CVPR.2015.7298682.

[35] M. Grandini, E. Bagli, and G. Visani, "Metrics for multi-class classification: An overview," *arXiv preprint arXiv:2008.05756*, 2020.

[36] L. Day, "Chevy Dealer's AI Chatbot Allegedly Sold A New Tahoe For $1, Recommended Fords - The Autopian," *Autopian*, Dec. 2023. [Online]. Available: https://www.theautopian.com/chevy-dealers-ai-chatbot-allegedly-recommended-fords-gave-free-access-to-chatgpt.

APPENDIX A.

This appendix contains confusion matrices for the PromptSentinel models measured on the Malignant dataset.

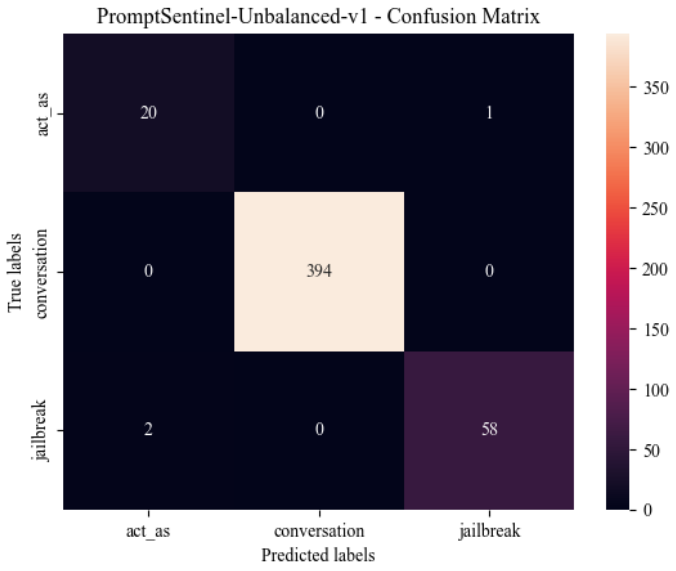

Fig. 1. Confusion matrix for PromptSentinel-Unbalanced-v1.

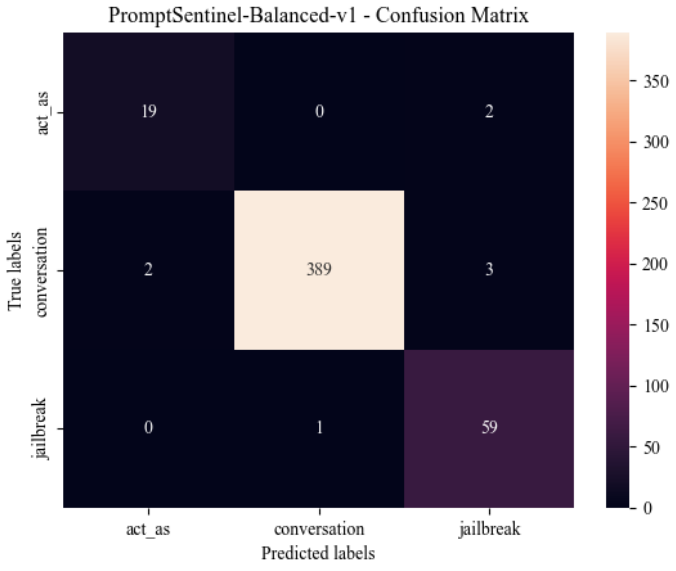

Fig. 2. Confusion matrix for PromptSentinel-Balanced-v1.

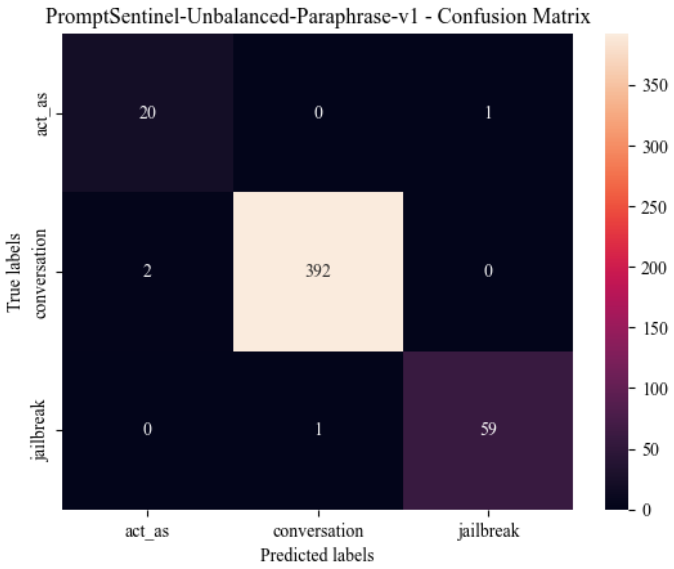

Fig. 3. Confusion matrix for PromptSentinel-Unbalanced-Paraphrase-v1.