Temperature in SLMs: Impact on Incident Categorization in On-Premises Environments

Marcio Pohlmann[1], **Alex Severo**[1], **Gefte Almeida**[1]
Diego Kreutz[1], **Tiago Heinrich**[2], **Lourenço Pereira**[3]

[1] AI Horizon Labs – PPGES – Universidade Federal do Pampa (UNIPAMPA)
[2]Max Planck Institute for Informatics (MPI)
[3]Instituto Tecnológico de Aeronáutica (ITA)

{alexsevero,marciopohlmann,geftealmeida}.aluno@unipampa.edu.br
diegokreutz@unipampa.eud.br, theinric@mpi-inf.mpg.de, ljr@ita.br

Abstract. *SOCs and CSIRTs face increasing pressure to automate incident categorization, yet the use of cloud-based LLMs introduces costs, latency, and confidentiality risks. We investigate whether locally executed SLMs can meet this challenge. We evaluated 21 models ranging from 1B to 20B parameters, varying the temperature hyperparameter and measuring execution time and precision across two distinct architectures. The results indicate that temperature has little influence on performance, whereas the number of parameters and GPU capacity are decisive factors.*

1. Introduction

The increasing volume and complexity of cybersecurity incidents have generated a growing overload on response teams, which face the need for scalable solutions for triage, categorization, and prioritization of events. Structured categorization of these incidents is essential for identifying patterns, understanding the diversity of threats, and improving defense strategies. However, this process still faces important limitations, such as ambiguity in reports, a lack of standardization, and a shortage of specialized professionals.

Automated artificial intelligence methods have emerged as promising solutions to accelerate incident categorization and improve operational efficiency [Ogundairo and Broklyn 2024]. Persistent challenges remain, including limited labeled data, semantic ambiguity, and heterogeneous attack formats [Ibrishimova 2019]. Recent studies indicate that hybrid strategies that combine human expertise with advanced computational techniques, especially Large Language Models (LLMs) capable of processing unstructured text and capturing complex semantic patterns, can significantly improve the robustness and accuracy of incident classification.

Conversely, the costs associated with LLM usage, as well as the need to anonymize sensitive incident-related information, may limit the adoption of these tools in corporate environments. In this context, Small Language Models (SLMs) gain relevance, as they can be executed locally (on-premises) and require less computational capacity. Moreover, these models allow fine-tuning through the selection of architectures with different numbers of parameters and calibration of inference hyperparameters, such as sampling (*top-k*, *top-p*, and *temperature*), penalization (*frequency*, *presence*, and *repetition*), and generation control (*max_tokens*, *min_tokens*, and *stop*) [Zhao et al. 2023].

In this work, we investigate the influence of the temperature hyperparameter in SLMs from different vendors and of different magnitudes, evaluating both execution time and precision in incident categorization. For the experiments, we used a dataset composed of six balanced categories of real incidents from a CSIRT, enabling a consistent assessment of the models' classification capabilities. The main contribution of this study is a systematic and comprehensive experimental evaluation involving 21 distinct models, offering a comparative analysis of the impact of temperature on SLMs executed locally for security incident categorization.

2. Parameters and Hyperparameters in Language Models

In language models based on deep neural networks, parameters correspond to the values learned during training, such as weights and biases that transform input signals across layers[1]. The number of parameters determines the model's representational capacity: larger architectures capture more complex linguistic relationships but require more computational resources and present a higher risk of *overfitting*.

Hyperparameters are defined before training or during inference, and they influence both the learning process and the model's behavior. In the context of inference, these hyperparameters can be organized into three categories: sampling, penalization, and generation control. Sampling hyperparameters determine how the next token is selected, including temperature, *top-k*, and *top-p*. Penalization hyperparameters adjust probabilities to avoid excessive repetition, while control hyperparameters define structural limits of the output, such as the maximum number of tokens and *stop sequences* [Zhao et al. 2023].

Temperature regulates the level of randomness in text generation. Low values tend to produce more deterministic and precise outputs [Renze 2024], while higher values increase diversity and the risk of incoherence [Wang et al. 2020]. Studies indicate that temperatures close to 0.0 are suitable for reasoning and translation, temperatures above 1.0 favor creativity, and values higher than 1.6 may lead to the so-called temperature paradox [Renze 2024, Li et al. 2025].

3. Methodology

The experiments were structured as illustrated in Figure 1, following a three-stage pipeline similar to that adopted in recent studies (e.g., [Severo et al. 2025a, Severo et al. 2025b]): Input Data, Processing, and Results Analysis. The objective was to evaluate the automated categorization of security incidents considering variations in temperature, processing time, and precision. For the experiments, two distinct computational architectures were used: (i) an AMD Ryzen 7 4800H with 32 GB of RAM and an NVIDIA GeForce GTX 1650 GPU (4 GB), and (ii) an Intel Core i7-12700 with 64 GB of RAM and an NVIDIA RTX A4000 GPU (16 GB).

Regarding the **Input Data** used to evaluate the performance of SLMs in classifying security incidents, we employed the dataset of real CSIRT incidents created and described by [Severo et al. 2025a]. The original dataset contains 194 anonymized records, from which we selected a balanced subset consisting of 6 distinct categories, each with 4 incidents, totaling 24 occurrences. The number of incidents per class was determined by

[1] `https://www.ibm.com/think/topics/model-parameters`.

the minority classes. The entire original dataset had previously been classified by two cybersecurity specialists, who independently analyzed the incidents, establishing a ground truth for comparison with the automated results.

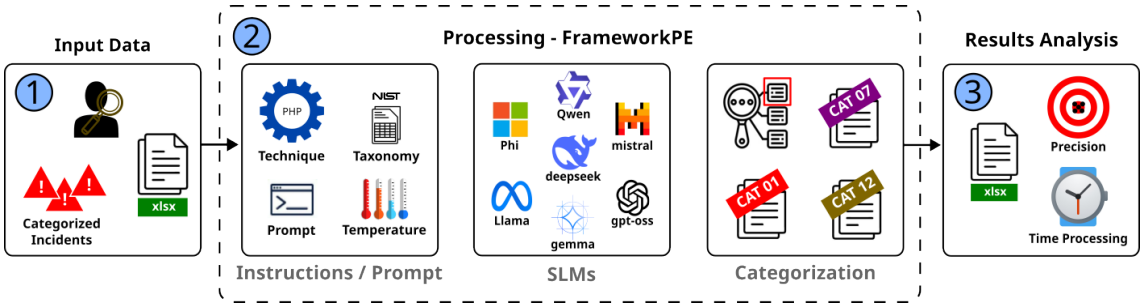

Figure 1. Experiment Stages

In the **Processing** stage, we used FrameworkPE[2], which implements several prompt engineering techniques. The user can select the desired technique, and for our experiments, we adopted PHP, which achieved the best results among the techniques recently evaluated for security incident classification [Severo et al. 2025b]. PHP is complemented by the use of a baseline taxonomy (NIST), textual rules for output standardization, parametrization of inference temperature, and the selection of different SLMs.

We included models ranging from 1 to 20 billion parameters made available by the Ollama provider (version 0.12.3), totaling 21 language models from seven different vendors. Each execution produces a collection of categorized incidents, which is subsequently used in the evaluation stage. The final stage, **Results Analysis**, consisted of measuring precision (comparison between inference and the ground truth) and processing time obtained by each model, considering the two hardware architectures used in the evaluation: Ryzen7/GTX 1650 and i7/RTX A4000.

4. Results and Discussion

Figure 2 shows the total time each model took to process the incidents on both architectures. (Ryzen7/GTX 1650 and i7/RTX A4000).

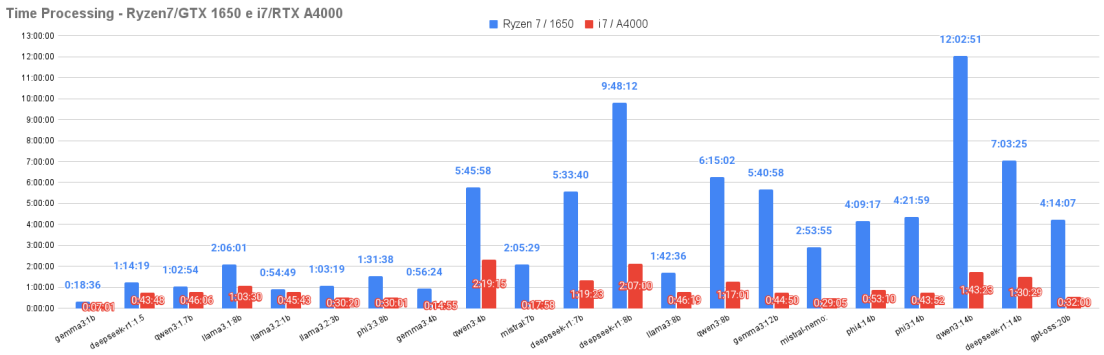

Figure 2. Processing Time per Architecture

[2]https://github.com/AILabs4All/FrameworkPE

The results indicate that the Ryzen 7/GTX 1650 architecture exhibited significantly higher execution times, while the i7/RTX A4000 architecture delivered substantially faster performance. This difference reflects the greater efficiency of CPU and GPU resource management in the second architecture, highlighting the direct impact of the execution environment on model performance.

Table 1 presents the processing times for categorizing the 24 incidents in each of the 21 language models evaluated under four temperature configurations (T0, T0.4, T0.7, and T1). The values are displayed in the H:MM:SS format, including totals per model and per temperature configuration.

Table 1. Execution times of the models under different temperatures

	Ryzen 7 / GTX 1650 Architecture					**i7/RTX A4000 Architecture**				
Model	**T0**	**T0.4**	**T0.7**	**T1**	**Model Total**	**T0**	**T0.4**	**T0.7**	**T1**	**Model Total**
deepseek-r1:1.5b	0:19:28	0:17:36	0:19:06	0:18:09	1:14:19	0:11:29	0:10:04	0:14:28	0:07:47	0:43:48
deepseek-r1:7b	1:18:48	1:22:36	1:22:30	1:29:46	5:33:40	0:24:23	0:14:21	0:24:48	0:15:51	1:19:23
deepseek-r1:8b	2:30:37	2:25:35	2:26:27	2:25:33	9:48:12	0:35:46	0:37:30	0:30:38	0:23:06	2:07:00
deepseek-r1:14b	1:46:26	1:46:29	1:44:51	1:45:39	7:03:25	0:12:19	0:29:25	0:24:09	0:24:36	1:30:29
gemma3:12b	1:26:06	1:24:42	1:24:58	1:25:12	5:40:58	0:03:47	0:06:21	0:10:36	0:24:06	0:44:50
gemma3:1b	0:04:57	0:04:38	0:04:31	0:04:30	0:18:36	0:01:17	0:01:23	0:01:54	0:02:27	0:07:01
gemma3:4b	0:14:08	0:13:59	0:13:56	0:14:21	0:56:24	0:01:48	0:05:20	0:05:15	0:02:32	0:14:55
gpt-oss:20b	1:03:14	1:05:48	1:03:40	1:01:25	4:14:07	0:04:01	0:03:44	0:09:15	0:15:00	0:32:00
llama3:8b	0:25:43	0:25:23	0:25:48	0:25:42	1:42:36	0:23:00	0:04:51	0:11:40	0:06:48	0:46:19
llama3.1:8b	0:25:43	0:31:28	0:33:44	0:35:06	2:06:01	0:12:14	0:15:14	0:23:42	0:12:20	1:03:30
llama3.2:1b	0:15:13	0:13:48	0:11:29	0:14:19	0:54:49	0:06:25	0:15:24	0:10:34	0:13:20	0:45:43
llama3.2:3b	0:15:01	0:15:38	0:17:09	0:15:31	1:03:19	0:07:17	0:05:37	0:09:29	0:07:57	0:30:20
mistral-nemo:12b	0:43:55	0:43:14	0:43:51	0:42:55	2:53:55	0:05:58	0:13:31	0:04:03	0:05:33	0:29:05
mistral:7b	0:32:04	0:30:34	0:31:21	0:31:30	2:05:29	0:04:28	0:04:41	0:03:41	0:05:08	0:17:58
phi3:14b	1:07:18	1:03:48	1:04:16	1:06:37	4:21:59	0:04:18	0:04:26	0:10:32	0:24:36	0:43:52
phi3:3.8b	0:23:11	0:24:39	0:21:40	0:22:08	1:31:38	0:03:08	0:07:09	0:15:28	0:04:16	0:30:01
phi4:14b	1:03:02	1:05:13	0:59:10	1:01:52	4:09:17	0:10:01	0:09:38	0:18:19	0:15:12	0:53:10
qwen3:4b	1:26:25	1:30:29	1:25:05	1:23:59	5:45:58	0:33:41	0:33:21	0:29:10	0:43:03	2:19:15
qwen3:1.7b	0:16:37	0:15:09	0:15:29	0:15:39	1:02:54	0:10:12	0:12:01	0:11:16	0:12:37	0:46:06
qwen3:8b	1:35:15	1:32:21	1:33:43	1:33:43	6:15:02	0:23:59	0:12:12	0:22:03	0:18:47	1:17:01
qwen3:14b	3:01:25	2:57:36	3:02:34	3:01:16	12:02:51	0:18:15	0:31:35	0:28:35	0:24:58	1:43:23
Temperature Total	**20:14:36**	**20:10:43**	**20:05:18**	**20:14:52**		**4:17:46**	**4:37:48**	**5:19:35**	**5:10:00**	
Architecture Total					**80:45:29**					**19:25:09**

As observed, the variation in the execution time between the different temperatures (T0, T0.4, T0.7, and T1) is relatively small, suggesting that the temperature hyperparameter primarily influences the diversity and textual coherence of the responses rather than inference time. However, smaller models such as *gemma3:1b*, *deepseek-r1:1.5b*, and *llama3:1.8b* exhibit execution times far below those of larger models such as *gpt-oss:20b* and *deepseek-r1:14b*. These results reinforce the notion that the number of parameters is one of the decisive factors for computational cost.

In the i7/RTX A4000 architecture, the greater dispersion in execution times can be explained by the way Ollama manages model loading and inference, performing *load/unload* operations and establishing varying communication patterns between CPU and GPU. This dynamic generates different latencies, depending on the model size and the usage of GPU memory.

Smaller models tend to underutilize CUDA cores, whereas larger models require memory reallocation, resulting in fewer linear execution times. The absence of specific optimizations, such as *mixed precision* and tuning for Tensor Core utilization, also con-

tributes to this oscillation. In contrast, the Ryzen7/GTX 1650 architecture exhibited more regular behavior, possibly due to more stable execution and simpler resource management.

Table 2 and Figure 3 present the summary of the average precision of the models. Table 2 shows the number of incidents correctly categorized and the accuracy percentage for each model and temperature across both architectures. As shown, precision remained relatively stable across the different inference temperatures (T0 to T1), indicating that the temperature hyperparameter exerts limited influence on the precision of classification.

Table 2. Model Precision at Different Temperatures

	Ryzen7/GTX 1650 Architecture								**i7/RTX A4000 Architecture**							
Model	**T0**	**%**	**T0.4**	**%**	**T0.7**	**%**	**T1**	**%**	**T0**	**%**	**T0.4**	**%**	**T0.7**	**%**	**T1**	**%**
deepseek-r1:1.5b	4	16,67	3	12,50	3	12,50	3	12,50	3	12,50	4	16,67	2	8,33	1	4,17
deepseek-r1:7b	11	45,83	10	41,67	12	50,00	12	50,00	9	37,50	11	45,83	11	45,83	10	41,67
deepseek-r1:8b	0	0,00	0	0,00	0	0,00	0	0,00	9	37,50	10	41,67	11	45,83	16	66,67
deepseek-r1:14b	16	66,67	15	62,50	15	62,50	15	62,50	16	66,67	17	70,83	15	62,50	16	66,67
gemma3:12b	15	62,50	15	62,50	15	62,50	15	62,50	14	58,33	15	62,50	14	58,33	15	62,50
gemma3:1b	0	0,00	0	0,00	1	4,17	1	4,17	1	4,17	1	4,17	2	8,33	0	0,00
gemma3:4b	15	62,50	15	62,50	15	62,50	15	62,50	15	62,50	15	62,50	15	62,50	15	62,50
gpt-oss:20b	15	62,50	16	66,67	17	70,83	17	70,83	14	58,33	16	66,67	16	66,67	16	66,67
llama3:8b	12	50,00	12	50,00	12	50,00	12	50,00	13	54,17	12	50,00	12	50,00	12	50,00
llama3.1:8b	16	66,67	16	66,67	15	62,50	14	58,33	14	58,33	15	62,50	15	62,50	14	58,33
llama3.2:1b	1	4,17	2	8,33	2	8,33	0	0,00	0	0,00	2	8,33	0	0,00	0	0,00
llama3.2:3b	14	58,33	15	62,50	14	58,33	16	66,67	10	41,67	10	41,67	12	50,00	8	33,33
mistral-nemo:12b	12	50,00	12	50,00	12	50,00	12	50,00	13	54,17	13	54,17	12	50,00	12	50,00
mistral:7b	15	62,50	16	66,67	15	62,50	15	62,50	15	62,50	15	62,50	16	66,67	15	62,50
phi3:14b	16	66,67	15	62,50	15	62,50	16	66,67	16	66,67	15	62,50	15	62,50	14	58,33
phi3:3.8b	12	50,00	13	54,17	14	58,33	13	54,17	14	58,33	10	41,67	6	25,00	11	45,83
phi4:14b	13	54,17	12	50,00	13	54,17	13	54,17	12	50,00	13	54,17	15	62,50	11	45,83
qwen3:4b	16	66,67	14	58,33	16	66,67	13	54,17	14	58,33	15	62,50	12	50,00	14	58,33
qwen3:1.7b	12	50,00	13	54,17	14	58,33	13	54,17	14	58,33	11	45,83	15	62,50	14	58,33
qwen3:8b	14	58,33	12	50,00	13	54,17	13	54,17	14	58,33	13	54,17	14	58,33	13	54,17
qwen3:14b	16	66,67	13	54,17	13	54,17	14	58,33	14	58,33	14	58,33	12	50,00	15	62,50

The comparison between the two architectures shows a slightly superior performance for the i7 and RTX A4000 combination, a result that likely reflects its higher processing capacity and more advanced GPU optimizations. Medium-sized models such as *deepseek-r1:14b*, *phi3:14b*, and *qwen3:4b* reached the highest precision levels on both platforms, suggesting a favorable balance between generalization ability and computational cost. In addition, the i7 and RTX A4000 system demonstrated more consistent behavior across the evaluated models, particularly those with larger parameter counts, reinforcing its suitability for workloads that demand stable inference performance.

As shown in Table 2, the experiments conducted on the Ryzen7/GTX 1650 architecture revealed recurrent failures in the categorization task when using the deepseek-r1:8b model. Subsequent analyses based on the execution logs obtained through Ollama's debug mode indicated architectural limitations of the GTX 1650 and the hybrid inference regime triggered when the model cannot be fully loaded into VRAM. Although the GPU provides 4 GB of video memory, only about 2.3 GB were effectively available for loading the model weights; the logs confirm that 2.3 GB were allocated on the GPU, while

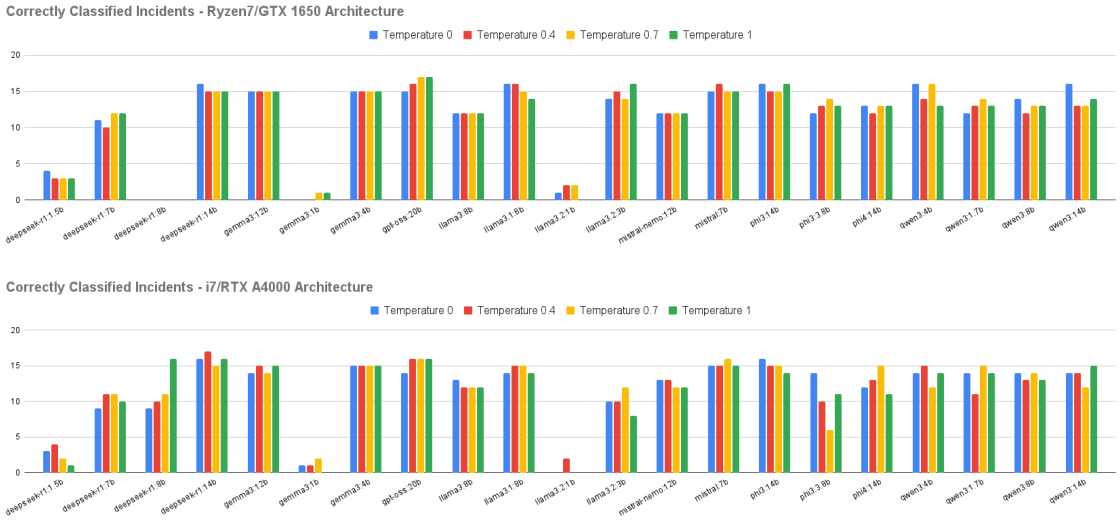

Figure 3. Precision per Architecture

approximately 2.6 GB remained in system RAM and were processed by the CPU. This partitioning forces the inference mechanism to rely on continuous data transfers over the PCIe bus, whose measured throughput was only 2.5 GB/s, in contrast with the 32 GB/s available on the RTX A4000, introducing substantial latencies during token generation.

In addition, only 20 of the 37 layers of the model were offloaded to the GPU, while the remaining 17 layers were executed on the CPU, significantly reducing the overall inference throughput. The KV-cache was similarly fragmented, with approximately 320 MB allocated on the GPU and 256 MB on main memory, which intensified the data exchange between CPU and GPU and contributed to performance fluctuations. As a consequence of these restrictions, the model exhibited considerably high response times (1m18s, 48.47s, 4m21s, 3m46s, and 4m54s), along with multiple timeouts. However, when executed interactively through the command line—thus avoiding the overhead of the HTTP API—the model was able to complete the task, suggesting that the failures did not stem from model configuration or intrinsic limitations, but rather from hardware bottlenecks.

Conversely, in the i7/RTX A4000 architecture, the model could be fully accommodated within the available VRAM, allowing both inference and categorization to proceed reliably and with low latency. The contrast between the two environments confirms that the inadequate behavior observed in the Ryzen7/GTX 1650 setup is directly related to insufficient video memory and the resulting partitioning of the model across CPU and GPU, which critically affects inference completeness and stability.

5. Conclusion and Future Work

The experiments showed that temperature has little impact on accuracy or inference time in automated incident categorization. The computational architecture and the number of parameters were the determining factors. Smaller models delivered higher efficiency, and medium-sized models offered a better balance between cost and accuracy, which reinforces the suitability of SLMs for local environments with limited resources. Performance varied across hardware. DeepSeek-R1 14B reached the highest precision on the

Ryzen7 and GTX 1650 system, while GPT-OSS 20B performed best on the i7 and RTX A4000 system. The i7 and RTX A4000 setting also showed greater variation in execution times, possibly due to how Ollama alternates model operations between CPU and GPU. The Ryzen7 and GTX 1650 system exhibited more linear and predictable behavior. To mitigate the inference instabilities observed in DeepSeek-R1 8B, several configuration adjustments were effective under hardware constraints. Reducing the number of layers stored on the GPU, lowering the context window, placing the KV cache entirely on the CPU when necessary, using lower-precision quantization, and increasing Ollama's API timeout collectively improved stability and ensured reliable local inference in resource-constrained scenarios.

Propostas futuras: (i) medir uso de CPU/GPU para correlacionar recursos, latência e estabilidade; (ii) investigar técnicas leves (quantização, poda, compressão) para reduzir tempo e memória em hardware de baixo custo; (iii) analisar o impacto combinado de hiperparâmetros (temperatura, *top-k*, *top-p*, penalidades) na qualidade semântica.

Acknowledgments. This research was partially supported by the CNPq[3], grant 409743/2025-9; by CAPES[4], under Financing Code 001; and by FAPERGS[5], through grant agreements 24/2551-0001368-7 and 24/2551-0000726-1.

References

[Ibrishimova 2019] Ibrishimova, M. D. (2019). Cyber incident classification: Issues and challenges. In *Advances on P2P, Parallel, Grid, Cloud and Internet Computing*.

[Li et al. 2025] Li, L., Sleem, L., Gentile, N., Nichil, G., and State, R. (2025). Exploring the impact of temperature on large language models:hot or cold?

[Ogundairo and Broklyn 2024] Ogundairo, O. and Broklyn, P. (2024). Natural language processing for cybersecurity incident analysis. *Journal of Cyber Security*.

[Renze 2024] Renze, M. (2024). The effect of sampling temperature on problem solving in large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, page 7346–7356. Association for Computational Linguistics.

[Severo et al. 2025a] Severo, A., Lautert, D., Almeida, G., Kreutz, D., Rodrigo, G., Jr, L. P., and Bertholdo, L. (2025a). LLMs e engenharia de prompt para classificação automatizada de incidentes em SOCs. In *Anais Estendidos do XXV SBSeg*. SBC.

[Severo et al. 2025b] Severo, A., Lautert, D., Kreutz, D., Bertholdo, L., Pohlmann, M., and Quincozes, S. (2025b). Categorização de incidentes de segurança utilizando engenharia de prompts em LLMs. In *Anais do XXV SBSeg*, pages 256–272. SBC.

[Wang et al. 2020] Wang, P.-H., Hsieh, S.-I., Chang, S.-C., Chen, Y.-T., Pan, J.-Y., Wei, W., and Juan, D.-C. (2020). Contextual temperature for language modeling.

[Zhao et al. 2023] Zhao, W. X., Zhou, K., Li, J., Tang, T., Wang, X., Hou, Y., Min, Y., Zhang, B., Zhang, J., Dong, Z., et al. (2023). A survey of large language models. *arXiv preprint arXiv:2303.18223*, 1(2).

[3] `https://www.gov.br/cnpq/pt-br`
[4] `https://www.gov.br/capes/pt-br`
[5] `https://fapergs.rs.gov.br`