ZeroLINC: Training-Free Local Classification of Security Incident Reports

Modality: Open Source

Cristhian Kapelinski[1], **Beatriz Machado**[1], **Diego Kreutz**[1]

[1]AI Horizon Labs, Federal University of Pampa (UNIPAMPA)

{cristhianavila,beatrizmachado}.aluno@unipampa.edu.br
diegokreutz@unipampa.edu.br

Abstract. *Security operations centers must classify large volumes of incident reports. Recent work automates this by prompting commercial large language models (LLMs), which charge per call and receive sensitive incident data, or locally hosted models with billions of parameters, which demand a GPU server. We present ZeroLINC, an open-source tool that lets a SOC or CSIRT analyst assign each incoming report to one of 12 NIST categories on a single local GPU, with no gradient training and no external API. Its instance-memory engine labels each ticket by similarity to the nearest already-labeled tickets. With 89 labeled examples it reaches* ***90.8%*** *mean test accuracy over the eight categories the corpus populates ($p < 0.001$ against the majority baseline), matching the band that prior work reached only with a commercial LLM. A second engine works with no labeled data at all, up to* ***70.9%****. We selected the engines with a systematic benchmark of 293 runs on 182 expert-labeled tickets from real incident response teams. Classifying the whole corpus takes seconds and under* ***3 Wh*** *on one GPU.*

1. Introduction

CERT.br registered over 274,000 incident notifications in Brazil in the first half of 2026, more than ten thousand per week[1]. Security Operations Centers (SOCs) and Computer Security Incident Response Teams (CSIRTs) have to read and classify that volume. In a recent study, two specialized analysts took 3 to 4 hours each to categorize fewer than two hundred tickets by hand [Severo et al. 2025b]. Report classification is therefore a triage bottleneck, and automating it reduces response time directly.

Recent work automates this classification with generative language models and prompt engineering. Severo et al. [Severo et al. 2025b] prompted four commercial large language model (LLM) APIs to sort 194 real CSIRT tickets into a 12-category taxonomy drawn from the NIST SP 800-61r3 incident response profile [Nelson et al. 2025]. Their best configuration (Gemini 2, progressive-hint prompting) agreed with the human experts on **92.27%** of tickets, for US$ 1.08 in API fees and up to 190 minutes of wall-clock time; others cost up to US$ 5.84. A follow-up study [Almeida et al. 2025] moved the same pipeline to locally hosted open models to avoid the privacy and data-sovereignty risks of sending incident data to third-party APIs: 7B to 70B models reach only around 60% accuracy on 24 tickets while requiring a GPU server. Both studies leave the same question open: training-free classifiers had not been evaluated on this task [Yin et al. 2019, Laurer et al. 2024a, Stepanov et al. 2025, Aarab 2026]. These are published models used as they are, with no gradient updates for the task, mostly under a billion parameters; *zero-shot* means using them with no labeled examples at all.

[1]CERT.br, Incident statistics, June 2026

We present ZeroLINC[2], an open-source tool that lets a SOC or CSIRT analyst assign each incoming report to one of 12 NIST categories on a single local GPU. It has two engines. The instance-memory engine labels a ticket by similarity to the nearest operator-labeled tickets; it reaches **90.8%** mean test accuracy from 89 such examples, over the eight of the twelve categories the corpus populates, and never updates a model weight. The zero-shot engine needs no labeled data at all and reaches up to **70.9%**. We selected the engines through a systematic benchmark of 293 evaluation runs. The design crosses four training-free classifier families with 8 category verbalizations and 4 input views. A verbalization is the wording of each category shown to the model, the analogue of prompt engineering. An input view is a cleaned-up version of the ticket text. We run the grid on the corpus lineage of the two reference studies, under a validation/test protocol with paired significance tests. Table 1 positions ZeroLINC against the related work. *Local* marks execution without an external API, *Cost* names the cost dimension each work reports, and *Acc.* gives the reported accuracy and ticket count (absent for the generic methods).

Table 1. Related approaches: incident classification studies and zero-shot classification methods.

Work	**Task**	**Method**	**Params**	**Local**	**Cost**	**Acc.** (n)
[Severo et al. 2025b]	SOC triage	LLM prompting	n/a	×	US$, min	92.3% (194)
[Almeida et al. 2025]	SOC triage	SLM prompting	7–70B	✓	time	≈60% (24)
[Yin et al. 2019]	generic	entailment	<1B	✓	–	–
[Stepanov et al. 2025]	generic	single-pass encoder	<1B	✓	examples/s	–
[Aarab 2026]	generic	zero-shot benchmark	0.1–12B	✓	latency	–
ZeroLINC (this work)	**SOC triage**	**zero-shot + k-NN memory**	**≤0.6B**	✓	**s, Wh, VRAM**	**70.9% (182) / 90.8% (93)**

Beyond the two reference studies, incident classification has been approached with models trained per deployment on labeled incident data [Ibrishimova 2019], and recent surveys document the alert-triage bottleneck that motivates automation across security operations [Habibzadeh et al. 2025, Ndichu et al. 2026]. LLM benchmarks cover adjacent tasks such as phishing detection [Nasution et al. 2025]. These regimes require what ZeroLINC avoids: gradient training on task labels or LLM-scale inference. On the methods side, Yin et al. [Yin et al. 2019] reformulated classification as textual entailment, Laurer et al. [Laurer et al. 2024a, Laurer et al. 2024b] reframed every class as a natural-language hypothesis and trained universal classifiers, Stepanov et al. [Stepanov et al. 2025] adapted GLiNER [Zaratiana et al. 2024] to score all labels in one pass, and Aarab [Aarab 2026] benchmarked these families on generic corpora. None had been evaluated on security-operations traffic. A classifier that over-predicts some labels can be recalibrated by rescaling its scores toward a uniform label prior [Zhao et al. 2021]. On SOC data this backfires (Section 2): a few categories genuinely dominate, so imposing uniformity removes signal, not bias.

ZeroLINC lets an operator (i) classify a ticket CSV with no labeled data; (ii) turn

[2]`https://github.com/CristhianKapelinski/zerolinc` (demonstration video: `https://youtu.be/hbo6mkqxbRc`)

confirmed classifications into a reference index that raises accuracy to 90.8%; and (iii) reproduce every number here from the committed run records. The paper follows the order in which the tool was built. Section 2 presents the benchmark that selected the engines and the accuracy, significance, and cost evidence. Section 3 presents the architecture that packages the winning configurations. Section 4 shows installation and usage. Section 5 gives limitations, conclusions, and next steps.

2. Benchmark and Engine Selection

A measurement study selected everything ZeroLINC ships. It answers two questions: which training-free configuration classifies best with no labeled data at all (the zero-shot grid of Figure 1), and which use of an accumulated labeled set classifies best (the lower block of Table 2 and Figure 2).

Corpus. We use the 182 unique expert-labeled tickets of the reference studies' corpus: their 194 records minus nine non-incident announcements and three exact duplicates. All are real e-mail threads received by Brazilian CSIRTs, labeled into the 12 NIST-derived categories by two security specialists [Severo et al. 2025b, Severo et al. 2025a]. The corpus is heavily skewed (118 of 182 tickets, 64.8%, are CAT5, that is, vulnerability exploitation; four categories have no positive example) and bilingual (132 tickets predominantly in Portuguese, 50 in English). The corpus was provided by the reference studies' authors on request and is not redistributed. The artifact ships the run records (per-ticket predictions and labels under sequential pseudonyms, no ticket text), from which every reported number regenerates.

Design. We compare four families of training-free classifier, none of which updates any weights for this task.

A *Natural Language Inference (NLI)* model judges whether one sentence follows from another; classification becomes asking, for each category, whether the ticket supports the statement *"this report describes* ⟨*category*⟩*"* [Yin et al. 2019]. A *GLiClass* encoder reads the ticket and all twelve category labels together and scores every category in a single pass [Stepanov et al. 2025]. An *embedding* model maps each ticket to a vector so that tickets with similar meaning land near each other [Zhang et al. 2025], and classifies by proximity to labeled examples. A *reranker* scores the ticket against each label as a separate yes/no question.

A checkpoint is one published model file, run as-is. The core of the study is a full grid over these families: 9 checkpoints $\times$ 8 category verbalizations $\times$ 4 input views $=$ 288 runs. Four further runs probe the reranker and the larger checkpoints. One leave-one-out run evaluates a k-nearest-neighbor (k-NN) similarity vote by classifying each labeled ticket with all the others. Majority-class and keyword baselines complete the comparison, for 293 runs. All experiments run on one machine: AMD Ryzen 5 8600G with 6 cores, 32 GB RAM, NVIDIA GeForce RTX 5060 Ti with 16 GB of GPU memory (VRAM), Linux. Inference is deterministic: every model returns its top-scoring category and sees the identical ticket list. The 4 input views are the normalized full text, a subject-first view, a boilerplate-filtered view, and their composition. The 8 verbalization sets, the column labels of Figure 1, range from bare category names (`*-name`) through NIST category descriptions (`*-desc*`) to event-style hypotheses (`en-event`); Appendices A and B detail both.

Because reporting a grid maximum would inflate the estimate (the reference studies tuned prompts on the set they report on), we use a selection protocol. We first split

the 182 tickets into a seeded, class-stratified validation half and test half (89/93 tickets), then select configurations on the validation half alone, then evaluate the selected one on the untouched test half, and repeat over 5 seeds. We test each configuration against the majority baseline with a paired McNemar test, which checks whether two classifiers differ on the same tickets beyond chance.

What moves zero-shot accuracy. Verbalization dominates: the same checkpoint varies by more than 60 percentage points between its best and worst wording, and semantically rich verbalizations beat bare category names on average (Figure 1). Language matching is model-specific: multilingual checkpoints handle English hypotheses over Portuguese tickets, whereas the English-only BART [Lewis et al. 2020] collapses unless the categories themselves are verbalized in Portuguese. No family wins everywhere. The best zero-shot run comes from the NLI model at **70.9%** accuracy and 0.54 macro-F1 (the unweighted mean of the per-class F1 scores), but GLiClass-modern-base reaches 67.0% on its best-scoring configuration (Figure 3a) at one seventh of the wall-clock time, which is why the tool packages both. Two negative results set the defaults. The reranker family stays below 20% raw accuracy, dominated by a per-label bias in its yes/no scores. Scaling up does not help: GLiClass-large reaches only 42.3%, below its base-size sibling, so the zero-shot engine keeps the smaller checkpoint.

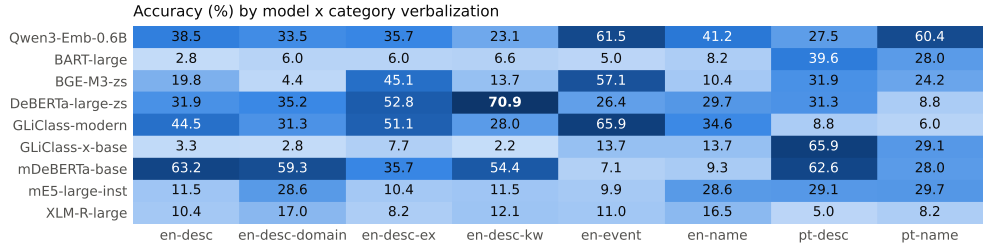

Figure 1. Accuracy (%) by model and category verbalization, subject-first view.

Protocol results: zero-shot. Table 2 averages each estimate over the 5 splits (min–max in parentheses), with every McNemar p taken against the majority baseline. Neither non-neural floor is a real solution. The majority class looks strong on raw accuracy (64.8%, 63.4% on the protocol's test halves) but covers only the dominant category (macro-F1 0.10). The keyword baseline built from the reference prompts' search terms reaches just 29.7%. Raw accuracy therefore misleads an operator on this skewed corpus. At this sample size no single zero-shot configuration separates from the majority baseline on accuracy, but all reach macro-F1 of at least 0.50 against its 0.10, so they do cover the minority classes. The cross-family ensemble (each family's best member combined by within-ticket score rank, a scale-free average) raises mean accuracy by a fraction of a point and reaches significance in one split. Its macro-F1 falls from 0.53 for the best NLI member to 0.37. Because macro-F1 captures minority-class coverage on this skewed corpus, this trade-off does not favor the ensemble. Per-label score calibration [Zhao et al. 2021] assumes a uniform label prior: on a corpus that is 64.8% one class it discards the real class balance and roughly halves the best configuration's accuracy. We evaluated and rejected it.

Protocol results: spending labeled tickets. SOC operation accumulates confirmed classifications within days, so the second selection question is which use of a labeled reference set classifies best. We gave the same validation half (89 tickets) to five candidates, listed in the lower block of Table 2. Two change model weights: a fully fine-tuned multilingual encoder (XLM-R-base [Conneau et al. 2020]) and a SetFit-

Table 2. Selection protocol: mean test accuracy, macro-F1, and McNemar p over 5 splits.

System (selected on validation)	Test acc. (%)	Macro-F1	McNemar p
Zero-shot (no labeled data)			
Majority class (floor)	63.4	0.10	–
Best NLI cross-encoder	68.8 (66.7–69.9)	0.53	0.34–0.69
Cross-family ensemble	69.0 (65.6–73.1)	0.37	0.04–0.48
Given 89 labeled reference tickets			
Fine-tuned encoder (XLM-R-base)	76.1 (74.2–81.7)	0.21	0.001–0.022
SetFit-style contrastive + head	89.5 (86.0–91.4)	0.58	< 0.001
Linear probe (logistic regression)	90.5 (89.2–91.4)	0.53	< 0.001
Nearest class centroid	91.0 (89.2–92.5)	0.62	< 0.001
k-NN similarity vote (selected)	90.8 (89.2–92.5)	0.56	< 0.001

style fine-tune, which pulls same-category tickets together and pushes different-category ones apart, then fits a small classifier on top [Tunstall et al. 2022]. Three read a frozen (never retrained) vector and differ only in how they use the labeled ones: a logistic regression; a nearest-class-centroid rule, which picks the closest per-category average vector [Snell et al. 2017]; and a k-NN similarity-weighted vote, with k chosen by leave-one-out inside the validation half. The outcome matches what the few-shot literature predicts. Full fine-tuning is the weakest use of the labels (76.1%, macro-F1 0.21) and the contrastive fine-tune recovers most of the gap (89.5%). The three frozen-embedding rules cluster at the top: each beats the majority baseline in every split ($p < 0.001$), and none is distinguishable from the others (paired McNemar between centroid and k-NN vote: $p \geq 0.48$ in every split). They disagree on only one or two tickets per test half, and the centroid's slightly higher macro-F1 rests on those same few. Training buys the operator nothing: every weight-updating option costs about one minute of GPU optimization per refresh, yet none beats the frozen rules, which update by appending a row to the reference set. With no metric separating the three, ZeroLINC packages the k-NN vote (**90.8%** mean test accuracy, range 89.2–92.5%), the only one whose predictions point to labeled tickets the analyst can open and audit.

How many labels the selected engine needs. The reference-set size was not tuned: 89 is simply the protocol's validation half. Stratified subsamples over the same 5 splits (tool defaults) show the entry cost. With 10 labeled tickets the engine already reaches 77.6%, above the best zero-shot configuration; with 30 it reaches 88.2%; and it saturates from about 45 labels (90.5%, Figure 2b). Figure 2a shows why so few suffice: recurring alert templates form tight clusters in the embedding space, so a single labeled instance resolves a whole template family. Scaling the embedding model does not pay. Under the protocol a 4B-parameter variant of the same family yields 89.9% against 90.8% for the 0.6B default (overlapping confidence intervals in Figure 3), at roughly 86 s and 11.4 GB of VRAM (against 28 s and 3.2 GB for the default), so the engine defaults to the smaller one.

When to trust memory over content. Figure 2c breaks the best zero-shot run down per class. The strongest classes are the frequent, lexically grounded ones: CAT5 (F1 0.91, n=118) and CAT3 denial-of-service (recall 0.94, n=17). The systematic failure is CAT12, intrusion attempts (recall 0.14, n=29). Most CAT12 tickets are forwarding notices whose informative evidence was an attachment the anonymized corpus does not contain; only the subject line separates the categories. In one representative case, a ticket whose entire visible content is a forwarding notice plus the subject *"Re: Abuse*

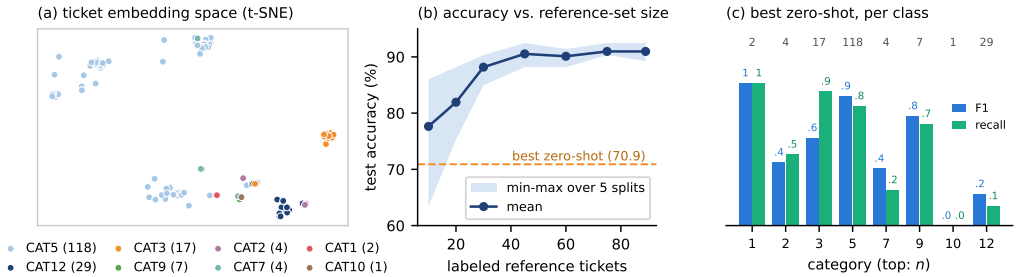

Figure 2. Instance-memory data regime: (a) 2-D map of the ticket vectors (t-SNE), colored by the category the experts assigned; (b) test accuracy vs. reference-set size (band: min–max over 5 splits; dashed: best zero-shot); (c) per-class F1 and recall of the best zero-shot run (top row: class size).

from <IP>" carries the expert label CAT12. Every zero-shot family misclassifies it. The instance-memory engine sidesteps this ceiling because it does not score content against category semantics: the reference set holds labeled instances of the same forwarding template and similarity retrieves them (recall 0.80, n=15, on the test half). Hence the tool ships both.

Cost. Figure 3a reports accuracy and Figure 3b reports wall-clock time for the best configuration of every evaluated checkpoint, including candidates the tool does not package, such as the reranker. The default zero-shot engine processes the 182 tickets in 8.0 s using 0.35 Wh[3] and 0.9 GB of VRAM. The strongest zero-shot configuration takes 56 s and 2.5 Wh. The instance-memory family (leave-one-out points in Figure 3) embeds and votes over the full corpus in 28 s at 90.7% accuracy with the default encoder, a cost the centroid rule shares. For the same lineage, the reference study reports full passes through commercial APIs taking from under 25 minutes to about three hours and costing single-digit dollar fees [Severo et al. 2025b]. The on-premise follow-up requires hosting 7B to 70B parameters, whose weights alone occupy 14 to 140 GB of memory at half precision, to reach around 60% accuracy [Almeida et al. 2025].

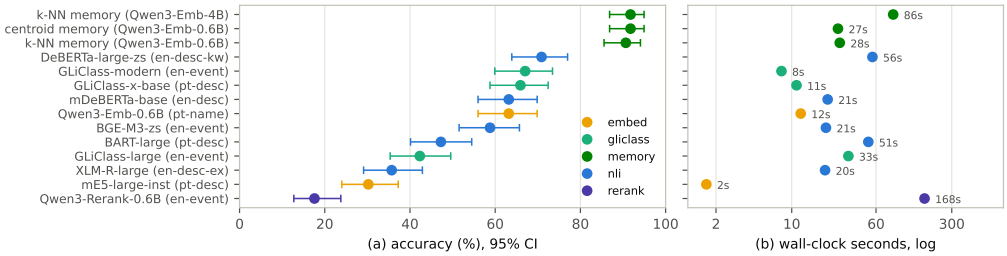

Figure 3. Best configuration per checkpoint and engine: (a) accuracy with 95% Wilson confidence interval; (b) wall-clock time for the full corpus, log scale.

3. The ZeroLINC Architecture

ZeroLINC packages the three configurations selected in Section 2 as a Python classifier built on PyTorch, Hugging Face Transformers, and Sentence-Transformers, with versions pinned by uv, a Python package manager. The three are a default zero-shot engine (GLiClass-modern-base), a maximum-accuracy zero-shot engine (the DeBERTa-v3-large-zeroshot cross-encoder, which scores one category at a time), and an instance-memory engine (Qwen3-Embedding-0.6B with a k-NN vote). Every model is an open checkpoint. Figure 4 shows the flow: a ticket CSV enters, five modules process it, and

[3]At the average Brazilian residential tariff of roughly R$ 0.92/kWh (US$ 0.16/kWh), 1 Wh costs under US$ 0.0002. ANEEL, InfoTarifa bulletin, 3rd ed., Dec 2025.

a prediction CSV leaves. The evaluation harness of Section 2 reuses these modules in a companion repository, adding the grid runner, the baselines, and the selection protocol; it is not part of the tool's runtime path.

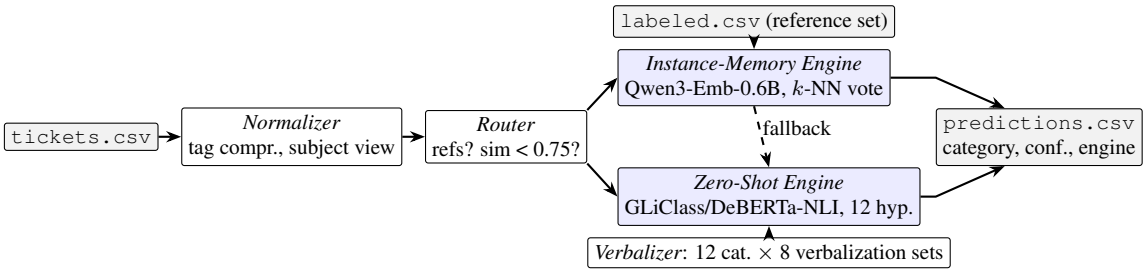

Figure 4. ZeroLINC architecture and data flow.

Normalizer. Tickets in the target domain are anonymized e-mail threads in which every sensitive span was replaced by a tag such as `[EMAIL_ADDRESS_f6f7...]`. The normalizer compresses each tag to a short placeholder (`<EMAIL>`, `<IP>`), because the hash suffixes waste encoder context without signal. It can prepend a copy of the e-mail `Subject` line, producing the subject-first view of Section 2. Each engine uses the input view chosen for its configuration in the benchmark.

Verbalizer. Zero-shot classifiers are steered by how each category is written out. This module holds the 8 verbalization sets evaluated in the benchmark, from bare category names to event-style declarative hypotheses that turn topic matching into a genuine entailment question. Appendix B lists all eight with examples. All of this text is taken verbatim from the reference studies' prompt material [Severo et al. 2025a], so the comparison with the LLM pipelines isolates the classification mechanism. For each checkpoint, the zero-shot engine uses the verbalization set and input view that scored highest in the benchmark. For the maximum-accuracy DeBERTa cross-encoder, that is the description-plus-keywords set (`en-desc-kw`) under the subject-first view.

Zero-Shot Engine. For deployments with no labeled data. The NLI backend follows the entailment formulation [Yin et al. 2019]: the ticket is the statement, each category verbalization is a candidate conclusion, and the model scores the 12 pairs so the best-supported category wins. That costs one pass through the model per category. GLiClass [Stepanov et al. 2025] instead encodes the label set together with the text and scores all categories in a single pass, hence its speed. The default checkpoint is GLiClass-modern-base, the best speed-accuracy trade-off of Section 2. The `--engine zeroshot-max` flag swaps in the benchmark's strongest configuration, the DeBERTa-v3-large-zeroshot cross-encoder [Laurer et al. 2024a, He et al. 2023] (70.9% on the full corpus). The backend abstraction implements instruction-embedding and generative-reranker backends [Wang et al. 2024, Chen et al. 2024, Zhang et al. 2025]: naming a checkpoint in the `--model` option evaluates any of these families.

Instance-Memory Engine. The most accurate engine, used when the operator provides labeled tickets. ZeroLINC turns each ticket into a vector with Qwen3-Embedding-0.6B [Zhang et al. 2025], finds its k nearest references (k=3 by default), and assigns the label those neighbors vote for by summed cosine similarity; the closest similarity becomes the confidence score. No gradient updates occur: knowledge is added by appending labeled examples to the reference set, which the `train` command persists as a reusable embedding index. The vote is chosen over the centroid and the linear probe, statistically indistinguishable from it in Section 2 ($p \geq 0.48$), because only the vote points at concrete labeled tickets the analyst can open and audit. The mechanism exploits the

corpus structure visible in Figure 2a.

Router. The router picks an engine per ticket. Without a reference set, everything goes to the zero-shot engine. With one, a ticket goes to the instance-memory engine unless its closest reference is less than 0.75 similar (adjustable); those tickets fall back to the zero-shot engine, so unseen incident types are not forced onto the memory.

4. Installation and Usage

Installed with `uv sync` from the repository,[2] ZeroLINC exposes one command. On day zero an operator classifies tickets with the zero-shot engine (`--engine zeroshot-max` selects the strongest configuration). Once the operation has labeled tickets, `train` builds the persistent reference index, which stores embeddings and trains nothing, and `classify` uses it with automatic per-ticket fallback (`--memory labeled.csv` also works directly):

```
uv run zerolinc classify --input tickets.csv --engine zeroshot
uv run zerolinc train --memory labeled.csv --model-out soc.npz
uv run zerolinc classify --input tickets.csv --model soc.npz
```

The tool reads a CSV of ticket texts and, for the reference set, a ticket id and a category (CAT1..CAT12); column names are configurable. It writes `predictions.csv` with one row per ticket: predicted category, confidence, and the engine used. Tickets below the similarity threshold are marked `zeroshot-fallback`, so an operator can review the least certain ones first. The repository documents the full column contract.[4]

CPU-only execution is supported (slower); models download automatically from Hugging Face on first use. Measured VRAM peaks: 0.9 GB (default zero-shot), 1.4 GB (strongest), under 4 GB (instance-memory).

5. Final Considerations

Limitations. Results come from the bilingual Portuguese and English corpus of one national CSIRT ecosystem. Tickets lack their attachments, which bounds recall on forwarding-notice classes for any content-based classifier, LLMs included. Only eight of the twelve categories have positive examples. The 90.8% figure is neither a zero-shot result nor a transfer claim: reference and test tickets come from the same operational workflow, so the accuracy rests on alert templates that recur inside that deployment and has to be re-measured on another team's traffic before it transfers. Our figures and those of the reference studies share the corpus lineage but differ in samples and in metric, human-judged generative answers there against exact expert-label match here, so the comparison concerns cost regimes and accuracy bands, not differences of a few points.

Demonstration. We classify anonymized tickets with no reference set, load one and show the accuracy jump, trigger a fallback, and rank the output by confidence. We need only the presenters' laptop (NVIDIA GPU, models pre-downloaded, no network) and a projector.

Conclusion. ZeroLINC shows that LLM-scale models are not a prerequisite for automated SOC incident classification: encoders of at most 0.6B parameters reach a selection-controlled 90.8% mean accuracy once about 90 labeled tickets exist, and up to 70.9% with none, in under one minute and under 3 Wh on one GPU, with no incident data leaving the organization.

Future directions. We will test whether one team's reference set helps another, request complete exports so attachment-bound classes become reachable, and grow the reference set automatically from SOAR platforms.

[4]The input columns are named `conteudo`, `incidente_id`, and `categoria` in the corpus, reflecting its Brazilian origin; the tool accepts any names via command-line options.

Acknowledgments

This work was partially supported by CAPES, CNPq (Grant No. 409743/2025-9), and FAPERGS (Grant Nos. 24/2551-0001368-7 and 24/2551-0000726-1). We thank the anonymous reviewers of SBSeg 2026 for their suggestions and comments. The authors also acknowledge the use of Generative AI as a supporting tool for code and manuscript review and improvement; they reviewed all content and take full responsibility for its accuracy and final version.

References

Aarab, I. (2026). BTZSC: A benchmark for zero-shot text classification across cross-encoders, embedding models, rerankers and LLMs. In *ICLR*.

Almeida, G. et al. (2025). On-premise SLMs vs. commercial LLMs: Prompt engineering and incident classification in SOCs and CSIRTs. In *ERRC*.

Chen, J. et al. (2024). M3-Embedding: Multi-linguality, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. arXiv:2402.03216.

Conneau, A. et al. (2020). Unsupervised cross-lingual representation learning at scale. In *ACL*.

Habibzadeh, A., Feyzi, F., and Ebrahimi Atani, R. (2025). Large language models for security operations centers: A comprehensive survey. arXiv:2509.10858.

He, P., Gao, J., and Chen, W. (2023). DeBERTaV3: Improving DeBERTa using ELECTRA-style pre-training with gradient-disentangled embedding sharing. In *ICLR*.

Ibrishimova, M. D. (2019). Cyber incident classification: Issues and challenges. In *3PGCIC*.

Laurer, M. et al. (2024a). Building efficient universal classifiers with natural language inference. arXiv:2312.17543.

Laurer, M. et al. (2024b). Less annotating, more classifying: Addressing the data scarcity issue of supervised machine learning with deep transfer learning and BERT-NLI. *Political Analysis*, 32(1).

Lewis, M. et al. (2020). BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *ACL*.

Nasution, A. H. et al. (2025). Benchmarking 21 open-source large language models for phishing link detection with prompt engineering. *Information*, 16.

Ndichu, S., Ban, T., Ozawa, S., Takahashi, T., and Inoue, D. (2026). AI-driven security alert screening and alert fatigue mitigation in security operations centers: A survey. arXiv:2605.08316.

Nelson, A. et al. (2025). Incident response recommendations and considerations for cybersecurity risk management: A CSF 2.0 community profile. Technical Report SP 800-61r3, NIST.

Severo, A. S. P. et al. (2025a). Categorização de incidentes de segurança utilizando engenharia de prompts em LLMs. In *SBSeg*.

Severo, A. S. P. et al. (2025b). LLMs e Engenharia de Prompt para Classificação Automatizada de Incidentes em SOCs. In *SBSeg (Extended)*.

Snell, J., Swersky, K., and Zemel, R. S. (2017). Prototypical networks for few-shot learning. In *NeurIPS*.

Stepanov, I., Shtopko, M., et al. (2025). GLiClass: Generalist lightweight model for sequence classification tasks. arXiv:2508.07662.

Tunstall, L. et al. (2022). Efficient few-shot learning without prompts. arXiv:2209.11055.

Wang, L. et al. (2024). Multilingual E5 text embeddings: A technical report. arXiv:2402.05672.

Yin, W., Hay, J., and Roth, D. (2019). Benchmarking zero-shot text classification: Datasets, evaluation and entailment approach. In *EMNLP-IJCNLP*.

Zaratiana, U. et al. (2024). GLiNER: Generalist model for named entity recognition using bidirectional transformer. In *NAACL*.

Zhang, Y. et al. (2025). Qwen3 Embedding: Advancing text embedding and reranking through foundation models. arXiv:2506.05176.

Zhao, Z. et al. (2021). Calibrate before use: Improving few-shot performance of language models. In *ICML*.

A. Input Views

The subject-first view prepends a copy of the e-mail `Subject` lines to the normalized text, so the most condensed statement of the ticket always sits inside the encoder window. The boilerplate-filtered view drops every line occurring in at least 15% of the corpus documents, a corpus-statistical rule with no hand-written patterns. The fourth view composes the two.

B. Verbalization Sets

Each set writes the 12 categories out differently; all wording comes verbatim from the reference studies' prompt material. `en-name`, `pt-name`: bare category names in English and Portuguese (*"Account compromise"*). `en-desc`, `pt-desc`: the NIST category descriptions (*"Unauthorized access to user or administrator accounts."*). `en-desc-domain`: the descriptions under a domain-anchored hypothesis template (*"This security incident report describes..."*). `en-desc-kw`, `en-desc-ex`: the descriptions extended with the reference prompts' own per-category search terms or illustrative examples. `en-event`: event-style declarative hypotheses (*"An attacker gained unauthorized access to a user or administrator account."*).

C. Artifacts

Two open-source repositories accompany this work, both licensed under the GNU Affero General Public License v3.0 or later (AGPL-3.0-or-later). **The tool** (the ZeroLINC classifier and its command-line interface): `https://github.com/CristhianKapelinski/zerolinc`. **The benchmark** (the evaluation harness, baselines, selection protocol, and committed run records): `https://github.com/CristhianKapelinski/zerolinc-benchmark`. Each includes a `LICENSE` file at its root. **Demonstration video**: `https://youtu.be/hbo6mkqxbRc`.