Christopher Aaron O'Hara

Udacity MSc Capstone – Generative Modeling

February 20, 2026

Overview

This project implements a Transformer-based generative workflow for LANL-style cybersecurity telemetry and evaluates whether generated event sequences can support structured RCA-style narrative drafting. The notebook uses a deterministic processed subset, compares baseline and ablation training configurations, generates synthetic incident sequences, and evaluates output quality with both diversity diagnostics and governance-oriented risk checks. The goal is a correct, reproducible generative implementation that supports analyst decision quality rather than polished free-form generation (Vaswani et al., 2017; LeCun et al., 2015).

Dataset: LANL 2017 Network Event Data with WLS Context Subset

Source links: https://csr.lanl.gov/data/2017/ ; https://csr.lanl.gov/data/auth/ ; https://csr.lanl.gov/data/cyber1/

Project repository: https://github.com/Ohara124c41/generative-modeling-lanl-rca

Motivation

In local-cloud and industrial SOC settings, telemetry is high-volume and heterogeneous, while triage windows are short. Generative models can help by drafting candidate event narratives and root-cause hypotheses, but this is only useful when generation behavior is measurable, auditable, and tied to risk controls (National Institute of Standards and Technology, 2024; Bellamy et al., 2019).

This project tests a practical middle layer between raw telemetry and analyst action: generated RCA artifacts. The key question is not only "can the model generate plausible sequences," but "does it remain informative under imbalance and heavy tails without collapsing into repetitive dominant patterns."

Dataset Description

The data source is publicly documented LANL cybersecurity telemetry. In this run, the notebook loaded a processed cache containing 300,000 network-flow rows (day 2) and a supplementary 50,000-row WLS context subset for optional contextual grounding (Los Alamos National Laboratory Cyber Security Research, 2017). The dataset is real, public, and non-synthetic, and it is not reused from prior capstone projects.

For modeling, 50,000 event rows were used under deterministic limits to keep runtime practical while preserving realistic schema behavior. The event representation includes timing, source/destination devices, protocol, ports, packet counts, and byte counts. Tokenization produced 600,000 tokens (12 tokens/event average) with vocabulary size 10,635.

EDA in the notebook shows protocol concentration and heavy-tailed traffic variables (for example packet and byte features with very high skew/kurtosis), which directly informed model-selection and evaluation strategy.

Model Architecture and Design Decisions

Technology stack

Implementation used Python with NumPy/Pandas for data handling, PyTorch for model/training, Matplotlib/Seaborn for plots, NetworkX for graph views, Jupyter for literate execution, tqdm for progress diagnostics, and AIF360 for optional bias-risk screening (Van Rossum & Drake, 2009; Harris et al., 2020; McKinney, 2010; Paszke et al., 2019; Hunter, 2007; Waskom, 2021; Hagberg et al., 2008; Project Jupyter, n.d.; da Costa-Luis et al., 2022; Bellamy et al., 2019).

Generative model choice

The model is a compact Transformer language model trained for next-token prediction over structured event tokens. This architecture was selected because self-attention can model long-range dependencies across event attributes (for example protocol, endpoint, and throughput token interactions) better than strictly local sequence heuristics (Vaswani et al., 2017).

Model selection and justification

This model choice follows directly from the task definition and observed data structure. Each LANL record is converted into a structured token bundle, and incidents are represented as short event sequences; that makes this a sequence-generation problem, so a Transformer language model is a principled fit. The model also produces token-level outputs that can be inspected and mapped into RCA hypotheses, which is preferable to opaque free-form generation for security operations use.

Why this matters: reviewers and practitioners need to see that the architecture was selected for problem fit, not convenience. A sequence-native model with auditable outputs supports technical traceability and safer downstream use.

Training setup

Core baseline settings were `d_model=128`, `n_layers=2`, `dropout=0.2`, `lr=1e-3`, `weight_decay=1e-4`, and 10 epochs. Optimization used Adam with gradient clipping (`GRAD_CLIP=1.0`) for stability (Kingma & Ba, 2015; Pascanu et al., 2013). Ablations changed one major factor at a time (dropout, width/depth, and learning rate), then extended the best short-run configuration (`low_lr_1e4`) to 10 epochs for deeper comparison.

Parameter justification

Hyperparameters were chosen to balance stability, runtime, and interpretability on available compute:

- `d_model=128`, `n_layers=2` provide enough capacity to learn event token relations without immediately escalating overfit risk.

- `dropout=0.2` adds regularization for concentrated heavy-tail telemetry.

- `lr=1e-3` baseline and `lr=1e-4` ablation explicitly test optimization sensitivity after early validation degradation.

- `weight_decay=1e-4` adds light regularization in repetitive sequence regimes.

- `epochs=10` with per-epoch validation enables observation of early-best/late-degrade behavior.

- fixed seed and deterministic preprocessing caps preserve fair comparability across ablations.

Why this matters: explicit parameter rationale makes model-selection claims defensible and reproducible instead of anecdotal.

Sampling and generation controls

Generation used temperature scaling and top-k constrained sampling with repetition controls, because unconstrained decoding in neural LMs often increases degeneration and repetitive loops (Holtzman et al., 2020; Fan et al., 2018).

Experimental Comparison

Controlled comparison design

The comparison strategy was staged and controlled. The baseline run used the default compact Transformer setup (`d_model=128`, `n_layers=2`, `dropout=0.2`, `lr=1e-3`). Ablations then changed one major factor at a time (dropout, model width/depth, or learning rate) while keeping data split, tokenization pipeline, seed, optimizer family, and evaluation logic fixed. This design isolates the effect of each change and avoids conflating architecture and optimization effects.

A second controlled step (Section 8B in the notebook) extended the strongest short-run candidate, `low_lr_1e4`, from the short ablation horizon to 10 epochs. This was not a new model family; it was a follow-up under the same configuration to verify whether the early validation advantage persisted under longer training exposure. The transition from ablations to 8B therefore serves as the experimental bridge from screening-level comparison to final candidate confirmation.

Why this matters: this structure turns model choice into an evidence trail. Instead of selecting a model from one-off runs, the workflow documents what changed, what stayed constant, and why the final model was promoted.

Output Evaluation and Interpretation

Training behavior and model selection

The baseline overfit after early epochs: validation loss was best at epoch 1 (~2.009) and degraded to ~3.747 by epoch 10, while training loss kept decreasing. In short-run ablations, `low_lr_1e4` gave the best initial validation loss (~1.649). The extended run `low_lr_1e4_ext10` improved best validation loss slightly to ~1.641 (perplexity ~5.160), then also degraded at later epochs. This indicates a consistent “early best epoch” pattern under current data conditions.

Relative to baseline best validation loss, the top model family produced about 18.3% improvement. Based on these results, the selected final model was `low_lr_1e4_ext10`.

Why this matters: this selection is evidence-driven. The final candidate is tied to controlled ablation results and a measurable improvement criterion.

Generated output quality

The model generated 8 sample sequences, usually 14 events each (one sample at 15). A concrete failure pattern is repeated motif carryover across early events (for example repeated `DP_5061 ... PR_6 ...`

structures), showing that coherence is present but novelty remains constrained.

Quantitative diagnostics from the run:

- distinct-1: about 0.244 to 0.363

- distinct-2: about 0.384 to 0.528

- repetition ratio: about 0.638 to 0.756

Distinct-n metrics and repetition ratios indicate moderate diversity with substantial repetition, so outputs are useful for hypothesis scaffolding but not comprehensive coverage of rare behavior (Li et al., 2016).

Why this matters: if repetition remains high, generated narratives can overweight frequent traffic modes and under-represent rare but operationally important patterns.

RCA interpretation layer

The RCA layer produced confidence-scored hypotheses for each generated sequence. In this run, confidence ranged about 0.356 to 0.455. Most outputs were interpreted as structured recurrent traffic patterns, with fewer burst-transition hypotheses. This supports the intended use as analyst-assist narrative drafting, not autonomous incident resolution.

Exported evidence artifacts

The run exported 8 rows of structured artifacts to:

- `data/processed/p5_generated_rca_artifacts.csv`

- `data/processed/p5_generated_rca_artifacts.jsonl`

This preserves generation traceability for downstream integration and audit.

Why this matters: traceable exports allow analysts to verify, challenge, and reproduce generated conclusions instead of treating them as black-box outputs.

Interpretation for a Non-Technical Audience

This system learns common patterns from historical cybersecurity network records and then generates example incident sequences with draft RCA explanations. In this run, it produced usable outputs, but many of them still looked similar to each other, which means the model is better at reproducing frequent patterns than inventing diverse rare-event scenarios. The confidence scores were moderate rather than high, so the outputs should be treated as analyst support, not as final incident truth. In practical terms, this tool can speed up first-pass triage by suggesting what might be happening, but a human analyst still needs to verify the evidence before action is taken.

Ethical Considerations and Responsible Use

Three project-specific ethical concerns were identified.

1. Bias amplification from dominant traffic modes: protocol concentration (for example high share of PR_17/PR_6 regimes) can cause the model to over-represent common patterns and under-surface rare but important behaviors. The AIF360 proxy audit flagged strong disparity on a protocol-based split (`SPD ~0.860`, `DI ~7.167`), signaling concentration risk rather than proving causal unfairness (Bellamy et al., 2019; Hardt et al., 2016).

2. Misuse risk in automation: generated RCA narratives can appear authoritative even when confidence is moderate and diversity is limited. Treating these outputs as automatic truth could distort triage in critical infrastructure contexts. This workflow therefore treats outputs as human-reviewed decision support with explicit uncertainty labeling (National Institute of Standards and Technology, 2024).

3. Creative ownership and attribution risk: generated narratives are synthetic compositions derived from public telemetry patterns. They should be treated as machine-assisted analytical drafts, with provenance

retained in exported artifacts and analyst accountability preserved in final incident conclusions.

Why this matters: in critical-infrastructure contexts, plausible but incorrect generated narratives can cause real operational harm. Ethical controls in this project are therefore tied to observed run behavior (moderate confidence, repetition, subgroup disparity), not generic policy statements.

Limitations and Future Improvements

Current limitations:

- subset scale constraints (50,000 modeling rows) relative to full LANL volume,

- persistent repetition despite improved low-learning-rate training,

- proxy-only fairness screening with a small audit set,

- heuristic RCA scoring rather than validated causal inference.

Near-term improvements:

1. Add early stopping and checkpoint policy driven by validation stability.

2. Expand decoding experiments (top-k vs top-p and stronger anti-repetition controls).

3. Add rare-pattern stratified evaluation and more failure-case slicing.

4. Broaden fairness-risk screening across multiple subgroup definitions.

5. Add retrieval-grounded evidence links from generated narratives to source event windows.

Future Work: Agentic AI Extension

The next logical step is an agentic triage workflow where an agent consumes the exported P5 artifact contract (sequence, RCA hypothesis, confidence, diagnostic scores), calls telemetry/query tools, and returns a transparent action plan with safeguards. The design should use explicit

decision logic, bounded tool permissions, stateful context, and fallback behaviors so that generation outputs become one evidence source among several, not a unilateral decision driver (Yao et al., 2023; National Institute of Standards and Technology, 2024).

Future Work: Integrated Industrial Application

For integrated industry synthesis, this P5 layer can serve as the narrative-generation component in a larger DevSecAIOps stack: statistical screening (P2), predictive ML triage (P3), deep detection (P4), generative RCA drafting (P5), and agentic orchestration (P6), all connected through traceable contracts and governance checks. The converging design direction is a local-cloud IIoT architecture where every handoff is auditable, risk-scored, and human-supervised, consistent with OT security and cybersecurity framework guidance (National Institute of Standards and Technology, 2024; Stouffer et al., 2023).

Reproducibility

The notebook includes deterministic configuration, fixed seed controls, explicit preprocessing caps, end-of-run V&V checks, and persisted model/checkpoint artifacts. The project-scoped `requirements.txt` lists minimal libraries required for rerun (not an indiscriminate full-environment freeze). Dataset access is documented, and processed-cache execution is supported for practical reruns without shipping multi-GB raw archives.

Technology, Method, and Concept Citation Coverage

Technologies explicitly used and cited:

- Python, Jupyter, NumPy, Pandas, PyTorch, Matplotlib, Seaborn, NetworkX, tqdm, AIF360.

Methods and algorithms explicitly used and cited:

- Transformer self-attention sequence modeling.

- Next-token language-model training with Adam optimization and gradient clipping.

- Controlled ablation design (single-factor comparisons).

- Temperature/top-k decoding with repetition controls.

- Diversity diagnostics (distinct-n, repetition ratio).

- Fairness-risk diagnostics (statistical parity difference, disparate impact).

- NIST CSF-oriented governance interpretation.

References

1. Bellamy, R. K. E., Dey, K., Hind, M., Hoffman, S. C., Houde, S., Kannan, K., Lohia, P., Martino, J., Mehta, S., Mojsilovic, A., Nagar, S., Ramamurthy, K. N., Richards, J., Saha, D., Sattigeri, P., Singh, M., Varshney, K. R., & Zhang, Y. (2019). AI Fairness 360: An extensible toolkit for detecting and mitigating algorithmic bias. IBM Journal of Research and Development, 63(4/5), 4:1-4:15. https://doi.org/10.1147/JRD.2019.2942287
2. Fan, A., Lewis, M., & Dauphin, Y. (2018). Hierarchical neural story generation. In Proceedings of ACL 2018. https://arxiv.org/abs/1805.04833
3. Hagberg, A. A., Schult, D. A., & Swart, P. J. (2008). Exploring network structure, dynamics, and function using NetworkX. In Proceedings of the 7th Python in Science Conference (pp. 11-15). https://networkx.org/documentation/stable/reference/introduction.html
4. Hardt, M., Price, E., & Srebro, N. (2016). Equality of opportunity in supervised learning. In Advances in Neural Information Processing Systems 29. https://proceedings.neurips.cc/paper/2016/hash/9d2682367c3935defcb1f9e247a97c0d-Abstract.html
5. Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., Fernandez del Rio, J., Wiebe, M., Peterson, P., ... Oliphant, T. E. (2020). Array programming with NumPy. Nature, 585, 357-362. https://doi.org/10.1038/s41586-020-2649-2
6. Holtzman, A., Buys, J., Du, L., Forbes, M., & Choi, Y. (2020). The curious case of neural text degeneration. In International Conference on Learning Representations (ICLR). https://arxiv.org/abs/1904.09751

7. Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. Computing in Science & Engineering, 9(3), 90–95. https://doi.org/10.1109/MCSE.2007.55
8. Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization. In International Conference on Learning Representations (ICLR). https://arxiv.org/abs/1412.6980
9. LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. Nature, 521, 436–444. https://doi.org/10.1038/nature14539
10. Li, J., Galley, M., Brockett, C., Gao, J., & Dolan, B. (2016). A diversity-promoting objective function for neural conversation models. In NAACL-HLT 2016. https://arxiv.org/abs/1510.03055
11. Los Alamos National Laboratory Cyber Security Research. (2017). LANL cyber datasets and event telemetry resources. https://csr.lanl.gov/data/2017/
12. McKinney, W. (2010). Data structures for statistical computing in Python. In S. van der Walt & J. Millman (Eds.), Proceedings of the 9th Python in Science Conference (pp. 56–61). https://doi.org/10.25080/Majora-92bf1922-00a
13. National Institute of Standards and Technology. (2024). The NIST Cybersecurity Framework (CSF) 2.0. https://www.nist.gov/cyberframework
14. Pascanu, R., Mikolov, T., & Bengio, Y. (2013). On the difficulty of training recurrent neural networks. In Proceedings of ICML 2013. https://proceedings.mlr.press/v28/pascanu13.html
15. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., … Chintala, S. (2019). PyTorch: An imperative style, high-performance deep learning library. In Advances in Neural Information Processing Systems 32. https://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library
16. Project Jupyter. (n.d.). Project Jupyter documentation. https://docs.jupyter.org/en/latest/
17. da Costa-Luis, C., Larroque, S. K., Altendorf, K., Mary, H., Korobov, M., Yarkov, A., Budin, R., Rodrigues, N., Lee, C., Newey, M., de Albuquerque, A., Zhiyuan, Z., Casper da Costa-Luis, … & tqdm developers. (2022). tqdm: A fast, extensible progress meter for Python and CLI. Journal of Open Source Software, 7(78), 4581. https://doi.org/10.21105/joss.04581
18. Stouffer, K. A., Pease, M., Tang, C., Zimmerman, T., Pillitteri, V. Y., Lightman, S., Hahn, A., Saravia, S., Sherule, A., & Thompson, M. (2023). Guide to Operational Technology (OT) Security (NIST SP 800-82r3). National Institute of Standards and Technology. https://doi.org/10.6028/NIST.SP.800-82r3
19. Van Rossum, G., & Drake, F. L. (2009). Python 3 reference manual. CreateSpace. https://docs.python.org/3/
20. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention is all you need. In Advances in

Neural Information Processing Systems 30. https://papers.neurips.cc/paper/7181-attention-is-all-you-need
21. Waskom, M. L. (2021). seaborn: Statistical data visualization. Journal of Open Source Software, 6(60), 3021. https://doi.org/10.21105/joss.03021
22. Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., & Cao, Y. (2023). ReAct: Synergizing reasoning and acting in language models. In International Conference on Learning Representations (ICLR). https://arxiv.org/abs/2210.03629