Univ. of Houston, COSC 4371 Projects Ideas

R. Verma, B. Tuck and A. Dunbar

September 15, 2025

1 Verma

Some of the projects use one of the following references:

- Swayamdipta, S., Schwartz, R., Lourie, N., Wang, Y., Hajishirzi, H., Smith, N. A., & Choi, Y. (2020). Dataset cartography: Mapping and diagnosing datasets with training dynamics. arXiv preprint arXiv:2009.10795.
- Goswami, M., Sanil, V., Choudhry, A., Srinivasan, A., Udompanyawit, C., & Dubrawski, A. (2023). Aqua: A benchmarking tool for label quality assessment. Advances in Neural Information Processing Systems, 36, 79792-79807.
- (CleanLab) Northcutt, C., Jiang, L., & Chuang, I. (2021). Confident learning: Estimating uncertainty in dataset labels. Journal of Artificial Intelligence Research, 70, 1373-1411.

Note that there are multiple opportunities for proposing projects that investigate the data quality issue using an appropriate framework in at least three different *recent and popular* datasets and its effect on classification performance for many of the security challenges. We have listed only a few of these possibilities below. For those contemplating a data quality and its impact project, you should also look at the 2019 CCS paper by Verma et al. on Data Quality.

Verma, R. M., Zeng, V., & Faridi, H. (2019, November). Data quality for security challenges: Case studies of phishing, malware and intrusion detection datasets. In Proceedings of the 2019 ACM SIGSAC conference on computer and communications security (pp. 2605-2607).

1.1 DIFrauD Projects

These projects involve the DIFrauD dataset, which is available for public use on HuggingFace. The paper on this dataset is in COLING 2024 by Dainis Boumber et al.

1. The DIFrauD dataset includes data from several different public sources. Investigate the dates these different datasets were released. Could this have an impact on the classification performance? Design an experiment to test this hypothesis rigorously and report before and after results with at least two different classifiers (at least one should be transformer-based).
2. Are all the samples in the DIFraud dataset in English? Use a language detector to determine the number and percentage of samples (classwise) that are from other languages. Report the distribution of languages for each class and domain of the dataset. Next, study the classification performance of at least two classifiers (at least one of them should be transformer-based) with and without the non-English samples. Report domain-wise performance and then aggregate (mean and weighted performance).
3. The different domains in the DIFrauD dataset have very different class imbalances. Investigate one oversampling, one undersampling, one sample weighting method and one hybrid method to address this class imbalance. Report domain-wise performance and the aggregate performance for at least two classifiers (at least one of them should be transformer-based). Also, report the baseline performance of these classifiers, i.e., without any method that mitigates class imbalance.

4. Investigate the label noise in the DIFrauD dataset using the CMU framework Aqua (Mononito Goswamit et al. NIPS 2023) and study its impact on classifier performance with at least two classifiers (at least one of them should be transformer-based).

5. Investigate the label noise in the DIFrauD dataset using the CleanLab framework and study its impact on classifier performance with at least two classifiers (at least one of them should be transformer based).

6. Investigate the label noise in the DIFrauD dataset using the Dataset cartography framework and study its impact on classifier performance with at least two classifiers (at least one of them should be transformer based).

1.2 Phishing and its variants

1. Refer to the 2025 paper "Can features for phishing URL detection be trusted across diverse datasets?" (available on arxiv.org) Do a similar study for phishing email detection with at least three different popular email datasets. Each dataset must be public, as recent as possible, and contain both phishing and nonphishing (legitimate) emails.

2. Similar to item 1) but for phishing website datasets.

3. Investigate label noise in the datasets used by the authors of the 2025 paper in item 1) using the Aqua framework and repeat their study with the clean version (i.e., remove all the noisy samples on which all four methods in Aqua agree) of these datasets.

4. Similar to item 2) but for phishing email datasets.

5. Similar to item 2) but for phishing website datasets.

6. Smishing: Investigate datasets for smishing and classification performance with at least two different classifiers (at least one of them should be transformer-based).

7. Vishing: Investigate datasets for vishing and classification performance with at least two different classifiers (at least one of them should be transformer-based).

8. QRishing Investigate datasets for QRishing and classification performance with at least two different classifiers (at least one of them should be transformer-based).

9. Design at least two different techniques to defeat the KnowPhish detection method proposed in USENiX Security 2024 conference. (Reference: KnowPhish: Large Language Models Meet Multimodal Knowledge Graphs for Enhancing Reference-Based Phishing Detection)

1.3 Malware Detection

1. Investigate the datasets and models that have been used in **three** malware detection papers that appeared in the last 3 years (2022-2025). Study the cross-dataset performance of at least two different classifiers from these papers.

2. Investigate the label noise issue among at least three different malware datasets that were used in malware detection papers in the last 3 years (2022-2025) using the Aqua framework. Investigate classifier performance for at least two different classifiers (at least one of them should be transformer-based).

1.4 Intrusion Detection

1. Investigate the datasets and models that have been used in **three** intrusion detection papers that appeared in the last 3 years (2022-2025). Study the cross-dataset performance of at least two different classifiers from these papers.

2. Investigate the label noise issue among at least three different intrusion datasets that were used in intrusion detection papers in the last 3 years (2022-2025) using the Aqua framework. Investigate classifier performance for at least two different classifiers (at least one of them should be transformer-based) before and after the noisy samples (all 4 methods in Aqua should agree on these) are removed from the datasets. Report also how the dataset sizes change in aggregate and classwise.

2 Tuck

2.1 Adversarial Machine Learning

Adversarial machine learning studies how small, targeted changes to inputs can cause a model to fail, typically within a constrained ℓ_p ball or via discrete token edits. Real systems face adaptive adversaries, and robust performance often lags clean accuracy. The challenge is crafting realistic perturbations (especially for text) that preserve meaning and measuring robustness in a reproducible way. In this track you will use and modify tools to generate attacks, evaluate defenses, and report clear metrics.

1. **Robustness Budget Curves (Text).** Fine-tune a classifier (e.g., DistilBERT) on a dataset (SST-2, IMDb, or AG-News), then evaluate robustness under `TextAttack` word, char, and semantic attacks at small budgets with standard constraints; report clean accuracy, robust accuracy vs. edit budget, and attack success rate, and test a defense or two (adversarial data augmentation using your generated examples, adversarial classifiers, etc.).
2. **Decision-Boundary Probing in Token Space (Text).** Use `TextAttack` to search for the minimal number of token-level edits that flip a trained model's prediction (you are estimating "how close" each example is to the boundary without gradients); summarize distributions of minimal edits overall and by part-of-speech, visualize neighborhoods with sentence embeddings (e.g., UMAP) to show clusters that are easy to flip, and test some preprocessing tweaks (spell-correction, synonym normalization, etc.) to see if minimal-edit counts increase.
3. **Visualizing ℓ_p Geometry (Images).** Train a CNN on MNIST/CIFAR-10, run Fast-Gradient Sign Method (FGSM) and Projected Gradient Descent (PGD) (ℓ_∞ and ℓ_2) with a ε sweep; report clean vs. robust accuracy, show a panel of adversarial examples along the ε path, and compare to a defense or two (input denoising, adversarial augmentation, etc.).

2.1.1 Starting Literature

- **Intriguing properties of neural networks** (Szegedy et al., ICLR 2013). Classic discovery of adversarial examples and transferability; motivates robustness beyond test accuracy.
- **Explaining and Harnessing Adversarial Examples** (Goodfellow et al., CoRR 2014). Introduces FGSM and the linearity hypothesis; simple, fast baseline attack/defense ideas.
- **Empirical Study of the Topology and Geometry of Deep Networks** (Fawzi et al., CVPR 2018). Analyzes decision-region geometry/topology, informing boundary thickness and connectivity.
- **TextAttack: A Framework for Adversarial Attacks, Data Augmentation, and Adversarial Training in NLP** (Morris et al., EMNLP 2020). Practical toolkit for text attacks/defenses.
- **Random Smooth-based Certified Defense against Text Adversarial Attack** (Zhang et al., EACL Findings 2024). Applies randomized smoothing to text for certified-style guarantees; useful for robustness vs. certificate trade-offs.
- **Assessing Representation Stability for Transformer Models** (Tuck & Verma, arXiv 2025). A model-agnostic detector that masks top-ranked words via importance heuristics and measures embedding drift to spot attacks, showing strong cross-dataset/attack generalization at low compute; also finds that gradient-based ranking is a better importance signal than attention or random selection, identifying perturbations more effectively.

2.2 Large Language Models

Large language models introduce security risks beyond standard classifiers because prompts, context windows, and tool use create new attack surfaces. Threats include jailbreaking, prompt injection/context hijacking, backdoors and data poisoning, encoding tricks, and style-based evasion. These problems are difficult since models are non-deterministic, instructions are malleable, and safety must be balanced with task utility. In this track you will build reproducible harnesses to measure attack success and evaluate practical mitigations.

1. **Jailbreak and Prompt-Attack Harness.** Curate a small set of safe prompts covering jailbreak styles (role-play, instruction override, coercive follow-ups), encoding tricks (base64, rot13, zero-width characters, homoglyphs), and style-transfer variants; measure attack success against models with a rubric, then test a defense or two (system prompt restatement, strict output schema, guardrail rephrasing, and removal of invisible characters) and compare success vs. utility.

2. **Context Hijacking and RAG Poisoning.** Build a retrieval corpus for a benign QA task and inject a few malicious documents that contain prompt-injection instructions or misleading answers; measure how often the model follows the injected instructions or cites poisoned content, then add a mitigation or two (source whitelisting, cite-then-answer prompting, instruction separators, refusal on conflict, truncation limits) and re-measure.

3. **Backdoors, Data Corruption, and Style Evasion.** In a few-shot classification setup, insert trigger phrases or style markers in the exemplars to bias outputs (backdoor), and create evasion examples that bypass a detector via paraphrase or style transfer; quantify flip rates with and without the trigger, track detector performance under style shifts, and test a few quick fixes (shuffle or sanitize exemplars, normalize style, strip triggers and invisible characters).

2.2.1 Starting Literature

- **Language Models are Few-Shot Learners** (Brown et al., NeurIPS 2020). Foundational GPT-3 paper; useful for understanding few-shot prompting, evaluation setups, and prompt sensitivity.
- **Mind the Style of Text! Adversarial and Backdoor Attacks Based on Text Style Transfer** (Qi et al., EMNLP 2021). Shows how style transfer can mount both adversarial and backdoor attacks in NLP, motivating style-based evasion and trigger analysis.
- **Poisoning Language Models During Instruction Tuning** (Wan et al., ICML 2023). Demonstrates that small amounts of poisoned instruction data can implant triggers and skew LLM behavior across tasks; relevant to backdoors and data corruption.
- **Jailbroken: How Does LLM Safety Training Fail?** (Wei et al., NeurIPS 2023). Investigates failure modes of safety training: competing objectives and mismatched generalization. Explores the elicited behavior of jailbreaks based on these failures.
- **LLMs for Explainable Few-shot Deception Detection** (Boumber, Tuck, Qachfar, and Verma, IWSPA 2024). Investigates LLMs detecting deception using a RAG framework for few-shot learning in domain-agnostic settings; useful for deception detection, RAG, and few-shot learning.
- **Unmasking the Imposters: How Censorship and Domain Adaptation Affect the Detection of Machine-Generated Tweets** (Tuck & Verma, COLING 2025). Examines how moderation/censorship and domain adaptation affect machine-generated text detection; useful context for evaluating detector robustness under distribution shift.
- **LLMs Under Attack: Understanding the Adversarial Mindset** (Bryan E. Tuck, IWSPA 2025). Tutorial overview of the LLM threat landscape: data poisoning, evasion, and prompt/jailbreak attacks; along with practical mitigations, limits of current detectors, and open research problems. *Please request the tutorial slides/materials from Bryan.*

3 Dunbar

3.1 SVD Projects

Software Vulnerability Detection (SVD) is an active area of research where researchers try to create models that are capable of detecting vulnerable code. Most of these datasets provide examples of vulnerable functions (rather than classes, projects, files or individual lines etc). There are many challenges in this domain that can be used for the basis of a project.

Suggested Potential Datasets

- PrimeVul
- DiverseVul

3.1.1 Imbalanced Data

1. Often SVD Datasets have far more non-vulnerable examples than vulnerable examples. For instance, the PrimeVul Dataset contains 228,800 non-vulnerable functions and only 6,968 vulnerable functions. This leaves the dataset imbalanced with respect to class labels which is difficult for many classifiers. Create a classifier that is resilient to this sort of class imbalance and show that it improves upon a more naive classifier's results. Make sure the dataset in question has imbalanced class representations.

3.1.2 Generalization

Note: Be careful when choosing your datasets when testing generalization. Many datasets explicitly borrow examples from one another or make use of the same source of functions. Make sure to deduplicate between datasets, and try to pick datasets that aren't explicitly dependent on one another.

1. Make a classifier that generalizes well to different datasets than the one it was trained up.
2. See how the quality of datasets effects the ability of a model to generalize. Many SVD datasets are of poor quality, even some of them that claim to be vast improvements on what came before, and it is a well known problem that many SVD datasets do not train models that generalize well to other datasets. See if using higher quality datasets allow the results to generalize better.

3.1.3 Code Representation

1. Compare the results of models trained on different ways of representing the code such as:
    - raw token sequence representations such as bag of words / tf-idf
    - AST based representations
    - Graph based representations such as Code Property Graphs / Data Flow Graphs
    - LLM embeddings based classifications, such as pretrained code models (GraphCodeBert, CodeT5+, etc)

3.1.4 Dataset Quality

1. Demonstrate the quality, or lack thereof, of several high-profile datasets in the SVD domain. Here are some potential datasets to sift through top of the suggested datasets above:
    - Devign
    - Big-Vul
    - MegaVul
    - Juliet
    - and many more (ReVeal, RealVul, D2A, etc)

3.1.5 Multimodal

Warning: Potentially difficult

1. Take a multimodal approach to classifying SVD datasets. Take multiple sources of information and fuse them together using multimodal architectures.