Data Classifier: An AI-driven approach to Label LLM Training Data

Praneeth Vadlapati
Independent researcher
praneethv@arizona.edu
ORCID: 0009-0006-2592-2564

Abstract: Large language model (LLM) training and fine-tuning depend critically on the quality of underlying datasets. Imperfect datasets can introduce noise, bias, and safety risks that propagate through model behavior. This paper presents Data Classifier, a pragmatic rule-based system designed to label and triage candidate training examples for LLM datasets into categories such as unsafe, spammy, sensitive, personally identifying information (PII)-containing, and syntactically malformed. The system combines deterministic heuristics, lightweight linguistic analysis, and optional model-backed checks to produce interpretable and reproducible labels suitable for dataset curation, filtering, and downstream auditing. We describe the conceptual design, implementation considerations, detailed heuristics, and evaluation methodology for assessing classifier utility in curation workflows. Rather than reporting definitive empirical performance numbers—which require large-scale annotated corpora and controlled experiments beyond the scope of this manuscript—we provide a rigorous experimental protocol, qualitative analyses on representative examples, and a discussion of limitations, trade-offs, and integration strategies. We conclude that a transparent rule-based classifier offers a useful first-pass filter that complements model-based detectors and human review, enabling more efficient and accountable dataset preparation for LLM training.

The source code is available at github.com/Pro-GenAI/DataClassifier.

Keywords: Large Language Models, LLMs, data curation, dataset labeling, rule-based classifier, language model safety, data hygiene, training data filtering, interpretability, Artificial Intelligence, Generative AI

I. Introduction

The rapid expansion of large language models has intensified attention on the provenance and quality of training data. Contemporary LLMs are trained on diverse, large-scale corpora drawn from web text, code repositories, question-answering forums, and crowd-sourced collections. While scale remains important, the composition of the training corpus determines many emergent behaviors in deployed models, including tendencies toward hallucination, propagation of harmful content, and replication of sensitive personal information. Dataset curation, therefore, is not merely a preprocessing step, it is a core safety and quality control mechanism.

Current dataset curation pipelines vary widely. Many practitioners rely on ad hoc scripts, heuristic filters, or fully automated model-based detectors to remove undesired content. Model-based detectors can be effective but are often opaque, resource-intensive, and may require retraining or careful calibration for new domains. Purely manual review is costly and infeasible at scale. In this context, rule-based systems present an attractive middle ground. They are transparent, auditable, reproducible, and cheap to run, yet sufficiently expressive to capture many classes of undesirable content encountered in LLM training corpora.

This paper introduces Data Classifier, a modular rule-based system designed to label text for downstream dataset curation. The aim is not to replace more sophisticated detectors but to provide a reliable, explainable triage layer that reduces workload for human reviewers and limits the

propagation of high-risk examples into model training. We situate the tool in the broader landscape of dataset hygiene, describe its design and heuristics in detail, and propose an evaluation plan for future empirical validation.

A. Disadvantages with Current Approaches

Existing approaches to dataset filtering and labeling exhibit several limitations. Model-backed detectors, including classifiers trained to detect toxicity, privacy leakage, or spam, can provide flexible, high-coverage detection but suffer from several drawbacks. First, they are opaque; their decisions are difficult to trace, complicating audits and remediation. Second, they can be brittle when applied out-of-distribution, requiring thresholds that may not generalize across domains. Third, deploying model-based detectors at scale incurs computational cost and infrastructure complexity.

Rule-based heuristics and lexical filters, on the other hand, are transparent and inexpensive but can be brittle in different ways. Hard-coded rules can suffer from high false positive or false negative rates if they are too simple; they may miss subtle context-dependent harms. Purely statistical or embedding-based approaches can collapse subtle distinctions (for example, benign reporting of a sensitive topic versus malicious instruction). Many production pipelines therefore combine multiple techniques, which introduces integration complexity.

Another challenge is the lack of standardized taxonomies and reporting conventions for labels applied during curation. Different teams use different label sets, making interoperability difficult. There is also a tension between over-filtering—discarding potentially useful training examples—and under-filtering—allowing harmful or low-quality content to remain. Both outcomes carry costs: over-filtering reduces data diversity and may hurt model performance; under-filtering decreases safety and increases post-deployment risks. Data Classifier seeks to mitigate these disadvantages by prioritizing transparency, modularity, and an extensible taxonomy that supports human-in-the-loop workflows.

B. Proposed System and Its Benefits

Data Classifier is a pipeline component that accepts text inputs and produces an interpretable list of labels that describe issues detected in the text. The system is designed with a layered architecture: initial lexical and structural checks are applied first, followed by lightweight linguistic analyses and optional model-backed verifiers for tasks requiring semantic understanding.

The benefits of this design are severalfold. First, interpretability: every label is associated with deterministic rules or documented heuristics, enabling straightforward audits and targeted corrections. Second, computational efficiency: the core checks rely on string matching, regular expressions, and simple token-level analyses that scale well with large corpora. Third, extensibility: the modular design allows teams to selectively enable heavier-weight analyses—such as named entity recognition for PII detection or transformer-based classifiers for sensitive content—only when necessary. Fourth, human-in-the-loop compatibility: the output of the classifier is intended to inform curated review queues, enabling reviewers to prioritize high-risk examples and to see exact rule triggers that led to a label.

The classifier labels examples into categories such as Spammy, Unsafe (e.g., explicit content or violent instruction), Sensitive (topics with ethical or legal concerns), PII-containing, Low-Quality (truncated or malformed text), and Needs-Review (ambiguous or borderline cases). Labels

are deliberately permissive: an example may receive multiple labels, allowing downstream systems to apply policy logic (e.g., discard if Unsafe and Spammy, or flag for manual review if PII-containing).

C. New Use Cases of the System

Beyond basic dataset filtering, Data Classifier supports a range of use cases that are increasingly relevant in research and production. One such use case is audit logging: by retaining the classifier labels and rule triggers alongside training examples, organizations can reconstruct why particular data were excluded or included, providing evidence for compliance or post-hoc analysis. Another use case is targeted augmentation: labels can be used to guide data augmentation strategies—for instance, selectively augmenting underrepresented benign dialects while excluding harmful constructs. The classifier also facilitates fine-grained dataset partitioning for fairness analysis, enabling researchers to measure model performance across slices defined by content characteristics rather than demographic attributes alone.

Finally, the classifier is useful in continuous data ingestion pipelines where rapid triage is needed. In such pipelines, removing obvious spam or malformed content early reduces strain on downstream compute and human review resources. The classifier's explainable labels make it easier to set conservative filtering policies initially, then iteratively relax or tighten rules based on observed false positive and negative rates.

D. Related work

The problem of dataset curation intersects with prior efforts in content moderation, toxicity detection, privacy-preserving machine learning, and automated dataset cleansing. Tools and frameworks that perform automated data purification operate at different points on the transparency-versus-flexibility spectrum. On one end, rule-based filters and heuristics offer traceability and low-cost operation; on the other end, supervised detectors and large pre-trained models provide semantic nuance at the cost of opacity and higher resource use.

Existing rule-based efforts focus on tasks such as spam detection and simple profanity filtering, while more sophisticated detection tasks often rely on supervised classifiers trained on human-labeled data. Other relevant work includes techniques for detecting PII and for safe redaction, as well as methods for auditing datasets to measure the presence of harmful content. Data Classifier is positioned to complement these approaches: it formalizes a pragmatic rule set, documents triggers, and provides optional hooks for integrating model-backed checks when additional semantic resolution is required.

II. METHODS

A. Design principles

The system is guided by four design principles. The first principle is transparency: labeling decisions must be explainable and traceable to concrete rules or documented heuristics. The second principle is modularity: the pipeline should allow individual checks to be enabled, disabled, or tuned independently. The third principle is efficiency: the default configuration should be fast enough to run on commodity hardware for very large corpora. The fourth principle is conservatism: in ambiguous cases the system should prefer flagging for review rather than making irreversible deletions.

B. Taxonomy and label semantics

The classifier uses an extensible taxonomy of labels. Each label has a clear semantic definition and associated acceptance criteria. For example, "Spammy" denotes content dominated by advertising language, repetitive sequences, or high-density URLs indicative of bulk promotional content. "Unsafe" refers to text that instructs harmful behavior, contains explicit sexual or violent content, or clearly promotes illegal activity. "Sensitive" captures domestic or geopolitical topics where careful handling is warranted. "PII-containing" identifies explicit personal data such as phone numbers, email addresses, government identifiers, or contextually sensitive personal narratives that likely contain uniquely identifying information. "Low-Quality" includes severely truncated text, garbled Unicode, or machine-generated noise. "Needs-Review" is assigned when multiple weaker heuristics suggest potential harm but no single deterministic rule suffices for confident classification.

C. Heuristics and rule set

The core implementation relies on layered heuristics. At the lexical level, the system employs regular expressions and token-level checks for URLs, email addresses, phone-number patterns, credit card-like sequences, and common profanity word lists. Structural checks look for anomalous sentence lengths, extreme repetition, and unusual character entropy that suggest scraping errors or corruption.

A mid-level analysis extracts named entities using a lightweight NER module (when available) to identify strings likely to be PII. The classifier also computes simple statistical features—such as URL density per token, average token length, and proportion of non-alphanumeric characters—to support threshold-based rules. The heuristics incorporate domain-sensitive whitelists and blacklists to reduce false positives; for example, a URL found inside a code snippet requires different handling than a URL embedded in natural language.

Optional model-backed checks are supported through plug-in modules. These include transformer-based toxicity detectors and semantic similarity matchers for known malicious instruction templates. Model-backed checks are only invoked when configured and are intended as complementary signals; the classifier records model scores alongside deterministic triggers to provide auditors with full context for decisions.

D. Workflow integration

The classifier is designed for two primary modes of use. In batch mode, it consumes files (for example, JSONL where each line contains a text field) and emits an annotated output where each input receives metadata fields indicating assigned labels and rule triggers. This output is amenable to programmatic post-processing or manual review. In interactive mode, the classifier can annotate single texts for analysis or for integration into data collection tools.

Label provenance is a first-class artifact. For every label assigned, the system stores a compact explanation consisting of the rule identifier, the matching span (if applicable), and the numeric or boolean evidence that triggered the label. These explanations enable straightforward human review and reproducible audits.

E. Evaluation methodology

Assessing a classifier of this kind requires careful consideration because the goal is not to maximize a single aggregate metric but to measure utility in a curation pipeline. We propose a mixed-methods evaluation comprising intrinsic and extrinsic components.

The intrinsic evaluation focuses on label-level accuracy and explainability. It requires a human-labeled corpus that spans the taxonomy, ideally sampled from candidate training corpora. Human annotators would mark presence/absence of each label on the same examples and provide rationales. Metrics include precision, recall, and the confusion matrix for each label, as well as a qualitative analysis of explanation usefulness (for example, measured via annotator feedback on whether the explanation helped them decide).

The extrinsic evaluation measures downstream impact on model training and safety. This can be assessed by training or fine-tuning lightweight models on datasets filtered by different policies (for example, no filtering, strict rule-based filtering, combined model+rule filtering) and comparing model behaviors on safety and utility benchmarks, such as propensity to produce disallowed content or task performance on held-out validation sets. Because such experiments can be expensive, we emphasize designing them to be minimally sufficient: use smaller, controlled training runs and carefully selected evaluation tasks that expose the intended harms.

III. Results

This manuscript focuses on articulation of the system design, heuristics, and evaluation protocol rather than on exhaustive empirical claims. Nevertheless, to illustrate the classifier's practical behavior we present a qualitative analysis of representative example classes, describe a small-scale manual annotation exercise used during development, and outline expected behaviors under different pipeline policies.

A. Qualitative illustration

In development, the classifier effectively captured clear instances of spammy content characterized by high URL density and repetitive promotional tokens. When applied to messages consisting primarily of advertisements and repeated calls-to-action, the Spammy label was applied with high confidence based on deterministic URL and phrase matches. The PII-containing label reliably flagged explicit sequences such as phone numbers and email addresses using conservative regular expressions; these matches were accompanied by precise span-level provenance so reviewers could quickly identify the sensitive substrings.

Complex or ambiguous cases, such as first-person narratives describing personal trauma, were assigned a Needs-Review label rather than a final Unsafe or PII label. This behavior aligns with our conservative policy to defer ambiguous, context-sensitive judgments to human reviewers. Similarly, some political or geopolitical discussion segments were tagged as Sensitive, enabling downstream policy decisions about whether to exclude, red-team, or retain such examples.

B. Small-scale annotation pilot

During an informal development pilot, a small curated set of several hundred examples drawn from web-scraped text and public forum posts was annotated by two domain experts. The pilot was intended to exercise label coverage and explanation utility. Annotators found the rule-based explanations helpful when they provided clear span-level evidence; in cases where labels were driven by aggregate heuristics (for example, entropy-based low-quality detection), annotators requested additional context such as example tokenization to better understand the decision. The pilot highlighted the practical need for user-configurable thresholds and domain-specific whitelists; these adjustments reduced false positives on code snippets and legal text.

C. Expected system behavior under pipeline policies

We describe three representative policies to illustrate trade-offs. A conservative policy discards any example labeled Unsafe or PII-containing automatically and sends Needs-Review examples to human queues. This policy minimizes risk of harmful content entering training data but increases reviewer workload. A permissive policy only discards examples explicitly labeled as clearly invalid (such as empty or severely malformed text) and uses other labels for analytics; this preserves data diversity but risks retention of subtle harmful content. A hybrid policy applies automatic discarding for high-certainty deterministic labels (explicit PII, clearly illicit instructions) while sending ambiguous or model-scored cases to review. The hybrid policy is the recommended default for most practitioners.

D. Limitations of qualitative results

The qualitative observations above are grounded in small-scale development and pilot annotation; they are not substitutes for large-scale, statistically rigorous evaluation. The pilot nonetheless provided directionality on classifier practicality: deterministic span-based rules are highly useful and interpretable, while aggregate heuristics require careful calibration and informative provenance to be reliably auditable.

IV. Discussion

A. Integration strategies

The classifier is most effective when integrated as part of a layered curation workflow. A recommended strategy is to run the rule-based classifier first, applying high-confidence deterministic filters to remove obviously invalid examples while tagging others for model-based checks and human review. This pipeline reduces the computational footprint of downstream model-based detectors and concentrates human attention where it is most needed. Retaining the annotation metadata enables post-hoc reconciliation and supports compliance requirements.

B. Ethical considerations

Labeling training data for content and privacy-related concerns raises ethical questions of overreach and bias. Rules reflect choices about what content is considered acceptable and what warrants removal. It is vital to make these choices explicit, document them, and involve diverse stakeholder perspectives when creating production policies. Additionally, automated removal of examples may disproportionately affect content from particular communities or dialects if rules are not carefully designed and tested. We advise practitioners to pair automated labeling with careful stratified sampling audits and with metrics that measure impact on representation and fairness.

C. Limitations and future work

A rule-based classifier cannot solve every curation problem. Semantic nuance, sarcasm, and context-dependent harm frequently require human judgment or sophisticated models. Another limitation is maintenance: blacklists, whitelists, and regex patterns can become stale and must be actively managed. Future work includes systematic benchmarking of the classifier against curated human-labeled corpora, automated methods for discovering missing rules from reviewer feedback, and methods for combining rule explanations with model saliency to produce hybrid explainable detectors.

V. Conclusion

Data hygiene is fundamental to responsible LLM development. This paper described Data Classifier, a pragmatic rule-based system for labeling and triaging candidate training examples. Emphasizing transparency, modularity, and explainability, the system provides a low-cost first-pass filter that complements model-based detectors and human review. We outlined a detailed taxonomy, heuristics, and evaluation strategy, and presented qualitative evidence from a development pilot illustrating practical utility. While not a panacea, a carefully designed rule-based classifier can reduce reviewer workload, improve auditability, and serve as an important component of a layered safety and quality assurance pipeline. We encourage practitioners to adopt conservative, auditable policies, to pair automated labeling with continuous human oversight, and to evaluate classifier impact empirically on downstream model behaviors.

REFERENCES

[1] <To be added later>