Evolving LLM-Generated Features for Interpretable Classification

Jack Butler
Amazon Web Services
jackbtlr@amazon.com

Zainab Afolabi
Amazon Web Services
zafolabi@amazon.com

Nikita Kozodoi
Amazon Web Services
kozodoi@amazon.com

Abstract

Large language models (LLMs) are increasingly used as classifiers, yet they operate as opaque systems whose decisions are difficult to interpret, which complicates their use in regulated domains such as credit scoring or medical diagnosis. We propose an evolutionary framework that iteratively discovers natural language feature definitions (rubrics) for interpretable classification. An LLM generates candidate binary features, evaluates each sample against them, and the resulting vectors can be used to train a transparent classifier such as logistic regression. The feature set evolves over multiple iterations guided by classification errors, per-class activation rates, and feature ablation scores. We evaluate across three benchmarks, including a credit risk dataset representative of regulated domains, comparing single-shot LLM rubrics, evolved rubrics, and direct zero-shot LLM classification. Evolved features improve over single-shot rubrics by +2.9 pp on average and outperform zero-shot LLM classification on two of three tasks, while providing fully auditable decision logic. On the credit risk task, the zero-shot LLM performs at chance (50.7%) with a strong bias toward a single class, whereas evolved features achieve balanced, interpretable predictions. Crucially, this failure is invisible in aggregate accuracy and surfaces only under per-class auditing. Analysis reveals that evolution is most effective when label boundaries cannot be inferred from category names alone or when the LLM lacks reliable domain-specific reasoning.

1 Introduction

Large Language Models (LLMs) demonstrate strong performance across diverse classification tasks, from sentiment analysis to content moderation [4, 17, 21]. However, these models operate as opaque systems whose internal decision processes are difficult to interpret or audit. In high-stakes settings such as credit scoring, medical diagnosis, and content moderation, practitioners require not only accurate predictions but also transparent explanations of the factors driving them [13]. This is acute in consumer lending, where decisions must carry adverse-action explanations under the Equal Credit Opportunity Act, and a classifier can fail silently by defaulting to one class while aggregate accuracy still looks plausible.

Traditional interpretable models (linear classifiers, decision trees, and rule-based systems) provide transparent decision processes but rely on manually engineered features that may fail to capture complex semantic patterns in natural language. Recent work has explored using LLMs to generate interpretable features [2, 14]. However, these approaches typically produce static feature sets that do not adapt to dataset-specific patterns or classification errors observed during training.

We propose an evolutionary framework that bridges this gap by combining the semantic understanding of LLMs with the transparency of linear classifiers. Our approach uses an LLM to iteratively generate and refine binary features (rubrics), i.e., short, human-readable descriptions of binary attributes. An LLM evaluates these features on each sample, producing feature vectors that train a logistic regression

40th Conference on Neural Information Processing Systems (NeurIPS 2026). Workshop: TAE (Trust-AI-Eval): Can We Trust AI Evaluation?.

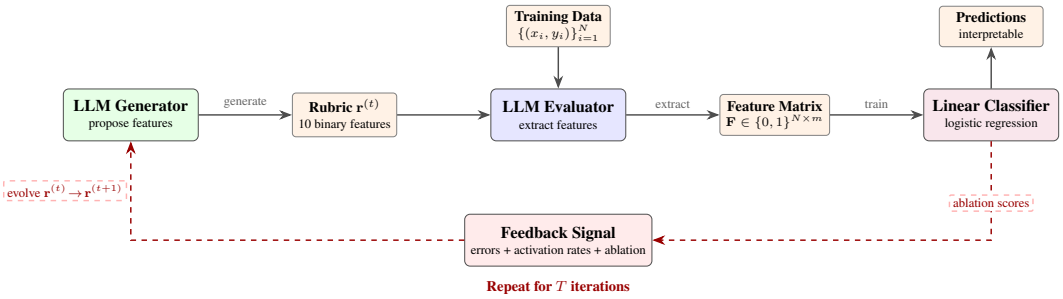


Figure 1: Overview of the evolutionary feature discovery framework. An LLM generates binary feature definitions (the rubric), evaluates each training sample against them, and a logistic regression classifier is trained on the resulting feature matrix. Classification errors, per-class activation rates, and feature ablation scores are fed back to guide the next rubric proposal (dashed arrows).

classifier. The feature set evolves over multiple generations, with performance and feature importance guiding which features are retained, modified, or created (Figure 1).

On a news task, the framework discovers features such as "Contains sports-specific terminology (athlete names, team abbreviations, league names)" ($|\bar{w}| = 0.85$) that a practitioner can validate.

We make two contributions. First, we introduce an evolutionary framework that iteratively discovers natural language feature definitions from LLM-generated hypotheses and downstream performance feedback, and we compare it against single-shot LLM rubrics and zero-shot LLM classification across three benchmarks spanning text classification and credit risk assessment. Second, we show that aggregate accuracy alone can hide a systematic single-class bias that drives a classifier to chance-level performance, which an interpretable feature layer exposes and which we characterise in terms of label taxonomy and LLM domain competence.

2 Related work

LLMs can perform classification through prompting, often achieving strong few-shot performance [4, 17, 21]. Key advantages include flexibility when class definitions change (no retraining required) and the ability to classify without labelled training data. However, these approaches rely on implicit reasoning within the model, making predictions difficult to inspect or audit. Even chain-of-thought prompting [17], while producing intermediate reasoning steps, does not guarantee that the stated rationale reflects the model's actual decision process.

Traditional interpretable models (linear classifiers, decision trees, and rule-based systems) provide transparent decision processes through explicitly defined features [13]. Concept Bottleneck Models (CBMs) formalize this by routing predictions through human-readable concept layers [15, 19], and textual CBMs build such concept bases for text classification without predefined human-labelled concepts [3]. Recent work on explainability for LLMs (e.g., attention visualisation, mechanistic interpretability) probes model internals, but these post-hoc methods remain far from the transparency of coefficient inspection in linear models, which regulators in finance and healthcare already accept as sufficient explanation [13]. We extend the concept bottleneck paradigm to text classification with automatically discovered concepts.

Singh et al. [14] demonstrate that LLMs can augment interpretable models by generating natural language explanations that serve as features. Balek et al. [2] propose using LLMs as interpretable feature generators for classification, showing that LLM-generated features can match or exceed hand-crafted ones. However, both produce static feature sets generated once at the start, which never adapt to error patterns observed during training.

OPRO [18] demonstrates that LLMs can serve as optimisers when provided with solution-score pairs, iteratively improving prompts through natural language feedback. Videau et al. [16] apply evolutionary algorithms to optimise few-shot prompt selection. These methods optimise the prompt given to an LLM classifier, whereas we optimise the features used by a separate transparent classifier, a different decomposition that preserves interpretability.

Genetic programming has long been used to discover features or model structures through iterative modification based on performance feedback [9]. More recently, Chen et al. [5] combine LLMs with evolutionary search to discover neural architectures, and AlphaEvolve [12] pairs an LLM mutation operator with automated evaluators to discover novel algorithms, underscoring the power of LLM-driven evolutionary search. Closest to our setting, LLM-FE [1] and CAAFE [8] have an LLM iteratively generate tabular features as executable code guided by validation feedback. The distinction is what the LLM writes: these methods emit transformations over existing columns, whereas our rubrics are natural language predicates an LLM evaluates against raw text, which applies where there are no columns to transform and keeps each feature legible to a non-programmer. Concurrent to our work, AutoQual [11] also discovers interpretable natural-language features for a linear predictor, but selects them by mutual-information beam search for review-quality regression; we instead drive discovery with classification errors, per-class activation rates, and ablation scores, and use the resulting transparency to expose a per-class failure mode. Our framework is thus evolutionary search where the LLM is both mutation operator and fitness evaluator over human-readable descriptions rather than code.

The LLM-as-judge paradigm [21] has shown that LLMs can reliably evaluate complex outputs, achieving agreement rates comparable to human annotators. Our framework uses this capability: the evaluator LLM acts as a structured judge, assessing whether each binary feature is present in a sample and returning parseable output. Judge reliability is therefore load-bearing here in a way it is not for post-hoc explanation, since an extraction error propagates directly into the feature matrix the classifier is trained on. Related evidence from tabular settings shows LLMs favour simple, low-order feature operations [10], which is consistent with the concrete, directly checkable predicates our generator is instructed to produce.

3 Methods

3.1 Framework overview

Our framework consists of three components: (1) a feature generator LLM that proposes natural language feature definitions, (2) a feature evaluator LLM that extracts binary feature values from text samples, and (3) a linear classifier trained on the resulting feature vectors. We use Claude Haiku 4.5 as both generator and evaluator. The feature set evolves over multiple generations using classification performance and feature importance as feedback signals.

Given a classification task with training samples $\{(x_i, y_i)\}_{i=1}^{N}$ and label set $\mathcal{Y}$, the system maintains a rubric $\mathbf{r} = \{r_1, r_2, \ldots, r_m\}$ of m natural language feature definitions. Each feature r_j is a short description of a binary attribute (e.g., "Does the text contain explicit criticism of a product or service?"). For each sample x_i, the evaluator LLM produces a binary feature vector $\mathbf{f}_i \in \{0,1\}^m$ indicating whether each feature is present. A logistic regression classifier is then trained on the feature matrix $\mathbf{F} \in \{0,1\}^{N \times m}$ to predict labels $\mathbf{y}$. The input x_i may be natural language text or structured data serialized to text (e.g., tabular features rendered as key-value descriptions).

3.2 Feature evolution

The evolutionary process proceeds over T iterations. At each iteration t:

1. **Evaluate**: Extract features from all training samples using the current rubric $\mathbf{r}^{(t)}$ and evaluate classification performance via k-fold cross-validation.
2. **Analyze**: Compute feature importance scores from the classifier coefficients, per-class activation rates for each feature, and feature ablation scores (accuracy change when each feature is removed). Identify misclassified samples and confusion patterns.
3. **Evolve**: Present the LLM with the current rubric, ablation scores (marking features as essential, useful, or unnecessary), per-class activation rates, and classification errors. The generator retains essential features and proposes replacements for unnecessary ones.

The generator is instructed to retain the top-k features by importance, propose replacements for low-importance features, and generate new features that address observed error patterns. Previously rejected rubrics are fed back to the generator to steer it away from repeating unproductive candidates,

Algorithm 1: Evolutionary Feature Discovery

Input: Training data $\mathcal{D}$, label set $\mathcal{Y}$, iterations T, retained top-k
Output: Optimised rubric $\mathbf{r}^*$

1. $\mathbf{r}^* \leftarrow$ LLM-GENERATE($\mathcal{Y}$, examples from $\mathcal{D}$)
2. $(s^*, \mathbf{w}, \mathcal{E}, \mathbf{a}, \mathbf{b}) \leftarrow$ EVALUATE($\mathbf{r}^*, \mathcal{D}$) // CV score, coefficients, errors, activations, ablations
3. $\mathcal{R} \leftarrow \emptyset$ // rejected candidates
4. **for** $t = 1$ **to** T **do**
   $\mathcal{K} \leftarrow$ top-k features of $\mathbf{r}^*$ by $|\mathbf{w}|$ // retained verbatim
   $\mathbf{r}' \leftarrow$ LLM-GENERATE($\mathcal{Y}, \mathcal{K}, \mathbf{w}, \mathcal{E}, \mathbf{a}, \mathbf{b}, \mathcal{R}$)
   $(s', \mathbf{w}', \mathcal{E}', \mathbf{a}', \mathbf{b}') \leftarrow$ EVALUATE($\mathbf{r}', \mathcal{D}$)
   if $s' > s^*$ **then** $s^* \leftarrow s'$; $\mathbf{r}^* \leftarrow \mathbf{r}'$; $(\mathbf{w}, \mathcal{E}, \mathbf{a}, \mathbf{b}) \leftarrow (\mathbf{w}', \mathcal{E}', \mathbf{a}', \mathbf{b}')$; $\mathcal{R} \leftarrow \emptyset$
   else $\mathcal{R} \leftarrow \mathcal{R} \cup \{\mathbf{r}'\}$
5. **return** $\mathbf{r}^*$

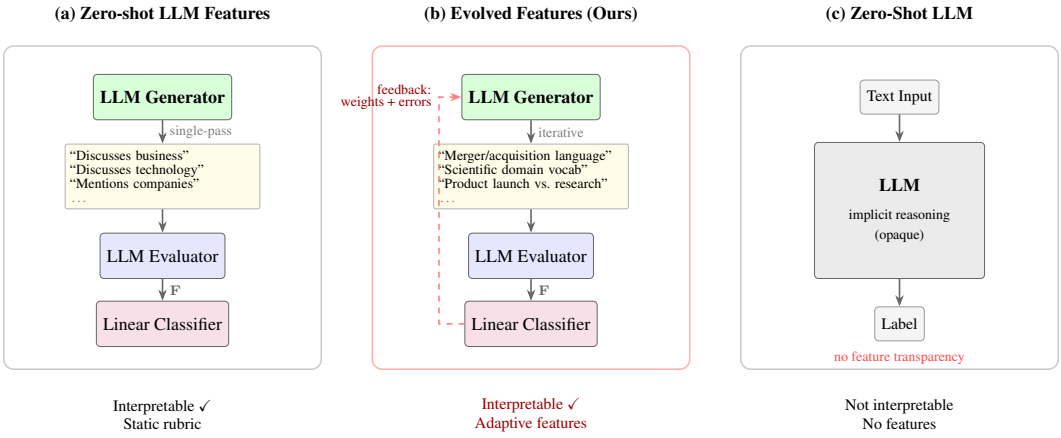


Figure 2: Comparison of three LLM classification approaches. (a) An LLM generates a static rubric of feature definitions in a single pass, evaluated by an LLM. (b) Our framework uses an LLM to iteratively generate and refine features via evolutionary feedback from classifier performance. (c) A zero-shot LLM classifies directly without explicit features. Only (a) and (b) produce transparent, auditable classifiers; only (b) adapts features to the data.

and are cleared whenever a candidate improves the score. The full procedure is summarised in Algorithm 1, where EVALUATE denotes feature extraction, k-fold cross-validation, and the analysis of step 2 (coefficients $\mathbf{w}$, errors $\mathcal{E}$, activations $\mathbf{a}$, ablations $\mathbf{b}$).

3.3 Feature extraction and classification

For each sample x_i and rubric $\mathbf{r}$, the evaluator LLM is prompted to output a binary vector indicating the presence or absence of each feature. The prompt includes the full rubric and the text sample, and the LLM returns structured output in a parseable format (the full prompt template is provided in Appendix A). Feature extraction is parallelized across samples for efficiency.

We use logistic regression as the downstream classifier, chosen for its transparency: the learned coefficients directly indicate the contribution of each feature to each class prediction.[1] The classifier is trained with ℓ_2 regularisation and evaluated using stratified k-fold cross-validation during the evolution process, with a held-out test set for final evaluation.

3.4 Baselines

Figure 2 illustrates the three LLM-based approaches we compare, alongside a TF-IDF baseline:

[1]We use scikit-learn's `LogisticRegression` with ℓ_2 regularisation ($C = 0.1$), the `lbfgs` solver, multinomial loss for multiclass tasks, and `max_iter`$= 1000$. Features are binary, so no standardisation is applied.

Table 1: Test accuracy (%) across 3 evaluation splits (mean ± std). Best per dataset in **bold**.

Dataset	TF-IDF	Zero-shot features	Evolved features	Zero-shot LLM
AG News	80.0±4.1	82.7±0.5	81.3±2.6	**89.7**±1.2
GoEmotions	31.3±8.2	34.7±6.2	**41.0**±1.6	38.7±3.7
HELOC	62.3±6.0	65.3±5.7	**69.0**±2.2	50.7±1.7
Average	57.9	60.9	**63.8**	59.7

- **TF-IDF + logistic regression**: A bag-of-words baseline using term frequency-inverse document frequency features with ℓ_2-regularised logistic regression. Fully transparent (individual word weights are inspectable) but limited to surface-level lexical patterns.
- **Zero-shot features** (Figure 2a): The LLM generates a rubric of m binary features in a single pass (iteration 0 of the evolutionary process), given the task description, label set, and a sample of labelled examples. No iterative refinement or performance feedback is used.
- **Evolved features** (proposed approach; Figure 2b): Features refined through the proposed evolutionary framework over T iterations, starting from the zero-shot rubric.
- **Zero-shot LLM classifier** (Figure 2c): The LLM classifies each sample directly with a single prompt, using no learned features. This black-box baseline offers no transparency but is expected to perform well given its flexibility.

4 Experimental setup and results

4.1 Datasets

We evaluate our evolutionary feature discovery framework across three classification benchmarks spanning different task types, difficulty levels, and domains:

- **AG News** [20]: 4-class news topic classification (World, Sports, Business, Sci/Tech). Well-separated categories with distinctive vocabulary.
- **GoEmotions** [6]: 7-class emotion classification (online comments mapped from 28 fine-grained labels to Ekman emotions). Complex taxonomy with arbitrary label boundaries.
- **HELOC** [7]: Binary credit risk classification (at-risk vs. not-at-risk for default on a Home Equity Line of Credit). Tabular credit report features serialized to natural language text. This dataset represents a regulated domain where interpretability is legally required and where LLM domain competence cannot be assumed.

4.2 Experimental configuration

The evolutionary search uses $n = 500$ samples per dataset with an 80/20 stratified train/test split, $T = 15$ iterations, 5-fold cross-validation for rubric selection, and rubrics of $m = 10$ binary features. We use Claude Haiku 4.5 as both the feature generator and evaluator, with ℓ_2-regularised logistic regression ($C = 0.1$) as the downstream classifier.

At each iteration, the LLM receives three forms of feedback: (1) classification errors showing confused class pairs, (2) per-class activation rates identifying non-discriminative features, and (3) ablation scores marking features essential, useful, or unnecessary, with instructions to keep the essential features and replace the rest.

To evaluate generalisation, we assess both the initial and final evolved rubrics on 3 independent stratified splits and report mean ± std. All baselines use identical data splits for fair comparison.

4.3 Main results

Table 1 presents the main results. The methods trade transparency against accuracy: TF-IDF offers full word-level interpretability but limited semantic understanding, while the zero-shot LLM classifier achieves the highest accuracy only on the one task with self-explanatory categories. Our evolved features achieve the best average accuracy across all three benchmarks while staying human-readable.

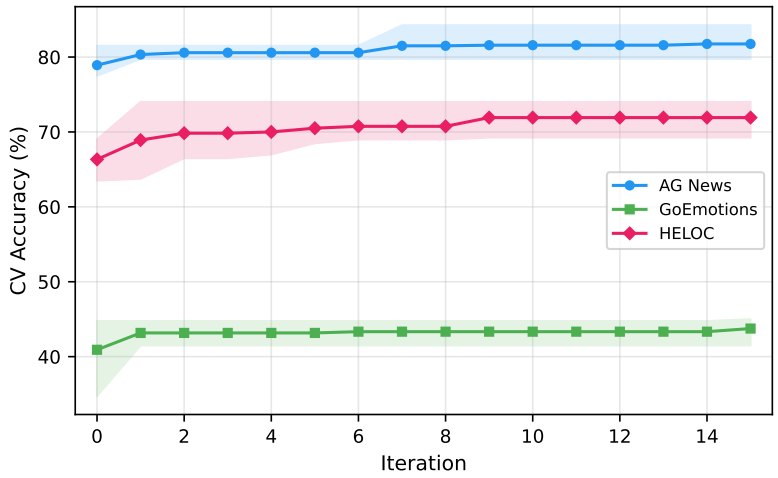


Figure 3: Cross-validation accuracy over evolutionary iterations. Solid lines show best-accepted CV score; shaded regions show the range across 3 seeds. Evolution makes decisive improvements early in the search on tasks where the initial rubric is suboptimal.

Evolved features improve over single-shot rubrics on all tasks except AG News (+2.9 pp average), with the largest gains on GoEmotions (+6.3 pp) and HELOC (+3.7 pp). On two of three tasks, evolved features also outperform the zero-shot LLM classifier while providing full interpretability. When label boundaries are non-obvious or the LLM lacks domain reasoning, the search finds discriminative patterns its implicit reasoning misses.

The HELOC result is particularly striking: the zero-shot LLM reaches only 50.7% accuracy, barely above chance, because it defaults to a strong "at_risk" bias (77–85% of predictions across seeds) when presented with structured credit data it cannot reliably reason about. This lopsided prediction pattern yields a high "at_risk" recall (0.80–0.84) but poor "not_at_risk" recall (0.14–0.26), so the two classes are far from balanced. In contrast, evolved features achieve 69.0% with balanced per-class predictions (per-class F1 of 0.64–0.71 for both classes across seeds), demonstrating that error-informed feature discovery produces functional classifiers even in domains where the LLM's zero-shot reasoning does not.

The GoEmotions result is also notable: evolved features (41.0%) outperform both the zero-shot LLM (38.7%) and TF-IDF (31.3%), suggesting that the 7-class Ekman emotion mapping creates a taxonomy that benefits from targeted feature discovery.

On AG News, the zero-shot LLM dominates (89.7%) because the four categories are well-separated and self-explanatory. Evolution does not improve on the initial rubric (81.3% vs 82.7%): refinement adds noise rather than signal when a single-shot rubric already captures the distinctions.

Across all tasks, the results reveal a clear pattern: evolutionary refinement is most valuable when the classification task requires features that cannot be inferred from label names alone, either because the taxonomy is arbitrary (GoEmotions) or because the domain requires specialised numerical reasoning that the LLM cannot perform zero-shot (HELOC). Evolved features also show substantially lower variance than zero-shot features on GoEmotions ($\pm$1.6 vs $\pm$6.2), suggesting that the evolutionary search produces more robust feature sets even when the initial rubric quality varies across data splits.

5 Analysis

5.1 Convergence analysis

Figure 3 shows the search trajectory for each dataset. On GoEmotions, the first accepted rubric yields substantial improvement (+10 to +12 pp), after which the search plateaus. HELOC follows a similar pattern: CV rises from a range of 0.65 to 0.69 into a range of 0.69 to 0.74 within the first few accepted iterations, then plateaus. This pattern of rapid initial gain followed by diminishing returns suggests that the primary value of evolution lies in the first feedback-informed refinements: the LLM

observes classification errors from the initial rubric and proposes a substantially better feature set. Subsequent iterations rarely improve further, as the bar for acceptance rises quickly. AG News shows no meaningful improvement, confirming that well-separated categories are adequately captured by a single-shot rubric without iterative refinement.

Based on these convergence patterns, we recommend $T = 5$ to 10 iterations as sufficient for most tasks. Across all three datasets, the vast majority of gains are realised within the first 5 accepted rubrics, though occasional late improvements occur (e.g., one HELOC seed accepted an improvement at iteration 9). Running beyond 10 iterations incurs additional LLM cost without measurable benefit.

5.2 Cost-accuracy trade-off

Table 2 shows the evolved features approach requires approximately 5,000 LLM calls during training (T iterations $\times$ n extraction calls per iteration, plus T generation calls). However, at inference time, all LLM-based feature methods require exactly one call per sample, the same as the zero-shot LLM classifier. Since inference dominates cost after deployment (training is amortised over all future predictions), the evolved and zero-shot feature approaches have identical operational cost. The extra $\sim$5,000 training calls buy +2.9 pp over zero-shot features at no added inference cost.

5.3 Feature evolution

We analyse the evolved rubrics to understand what evolution discovers (full visualisation in Figure 5, Appendix B). Across all three datasets, evolution aggressively replaces the initial rubric: AG News retains only 1 of 10 initial features, GoEmotions retains 3, and HELOC retains 2. The replacements follow task-specific strategies. On text tasks, evolution replaces generic, broadly-firing features with specialised discriminators targeting specific confusion pairs, for example, replacing "Presence of exclamation marks" (which fires across multiple emotion classes) with "Presence of concrete threat or aggressive action framing" (targeting anger specifically). On HELOC, evolution discovers numerical threshold features ("Nr Trades Insolvent For Over 60 Days $\geq$ 1") that are effectively decision stumps expressed in natural language, binary splits that a domain expert can immediately validate against regulatory guidelines.

5.4 Interpretability

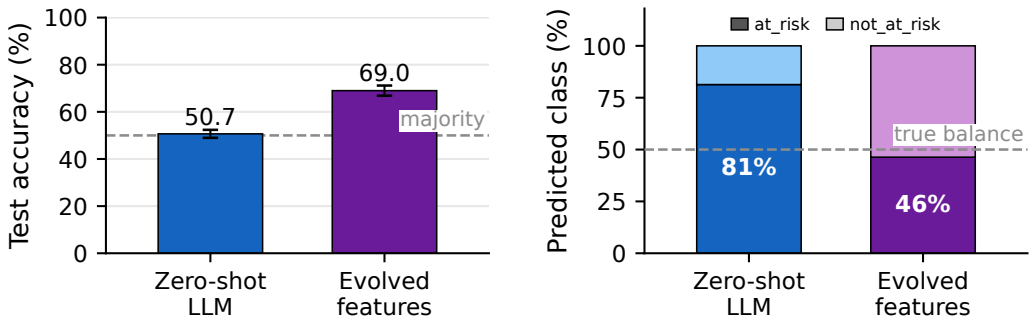
A key advantage of the framework is end-to-end transparency. For any prediction, a practitioner can inspect: (1) the 10 feature definitions in natural language, (2) the binary feature values extracted for the input, and (3) the logistic regression coefficients linking features to classes.

This distinction matters for evaluation practice: Figure 4 shows that the zero-shot LLM assigns `at_risk` to 81% of a balanced test set, a systematic bias that its near-chance aggregate accuracy conceals but the interpretable feature layer exposes.

This transparency is especially valuable in regulated domains. Table 3 gives the full evolved HELOC rubric, the decision logic behind every prediction on the task where the zero-shot LLM falls to chance. A credit decision reads as: "Applicant classified as at-risk because: Net Fraction Of Revolving Burden $>$ 40 (high revolving debt), Nr Trades Insolvent For Over 60 Days $\geq$ 1 (moderate delinquency history)." Such explanations are structured to support the adverse action notices required by the Equal Credit Opportunity Act, though establishing their legal validity would require fairness, proxy-discrimination, and domain-expert auditing beyond our scope. Every row is a threshold a credit officer can check against a report and dispute, which is what makes the failure of the zero-shot baseline visible in the first place. Rubrics for the two text tasks appear in Appendix C.

Table 2: LLM calls required for training (feature discovery) and inference (per sample). Evolved features assume $T = 10$ iterations with $n = 500$ training samples.

Method	Classifier training calls	LLM inference calls
TF-IDF + logistic regression	0	0
Zero-shot features	501	1
Evolved features (ours)	5,010	1
Zero-shot LLM classifier	0	1



(a) Aggregate view

(b) Per-class audit

Figure 4: Aggregate accuracy hides a per-class failure on HELOC. (a) Both methods sit near or above the majority baseline, so neither looks broken. (b) On the balanced 50/50 test set, the zero-shot LLM predicts `at_risk` for 81% of applicants (a systematic single-class bias), whereas evolved features stay balanced. The failure surfaces only under per-class auditing.

Table 3: Evolved rubric for HELOC, the task on which the zero-shot LLM performs at chance with a strong single-class bias. Evolution discovers numerical thresholds that act as decision stumps expressed in natural language, each checkable against a credit report.

#	Feature definition	$\lvert\bar{w}\rvert$
1	Net Fraction Of Revolving Burden $>$ 40 (high revolving debt)	0.652
2	Number Of Satisfactory Trades $<$ 15 (few satisfactory trades)	0.539
3	Months Since Last Illegal Trade between 0 and 12 (recent illegal-trade activity)	0.505
4	Percentage Of Legal Trades $<$ 90 (low legal-trade share)	0.399
5	Months Since First Trade $<$ 100 and Nr Trades Initiated In Last Year $\geq$ 3 (short history with recent activity)	0.374
6	Nr Banks With High Ratio $\geq$ 2 (multiple lenders stressed)	0.314
7	Nr Trades Insolvent For Over 60 Days $\geq$ 1 (moderate delinquency history)	0.285
8	Nr Inquiries In Last 6 Months $\geq$ 3 and Nr Total Trades $\leq$ 25 (excessive inquiries relative to portfolio)	0.131
9	Nr Total Trades $\leq$ 5 and Nr Trades Initiated In Last Year $\geq$ 2 (minimal portfolio with high recent activity)	0.088
10	Net Fraction Of Installment Burden $>$ 85 (extreme installment burden)	0.050

6 Conclusion

We presented an evolutionary framework that uses an LLM to iteratively discover natural-language features for interpretable classification, producing classifiers whose decision logic is human-readable and auditable. Across three benchmarks, evolved features improve over single-shot rubrics by +2.9 pp on average and outperform zero-shot LLM classification on two of three tasks. On the HELOC credit-risk task, where the zero-shot LLM performs at chance with a strong single-class bias, they remain balanced and auditable. Evolution helps most when label boundaries cannot be inferred from category names alone, or when the LLM lacks domain reasoning.

Evolution adds little when a single-shot rubric is already near-optimal or the LLM's general knowledge suffices for direct classification. The binary representation also bottlenecks fine-grained tasks, and our analysis uses a single LLM (Claude Haiku 4.5), leaving cross-model generalisation to future work. Other directions include broader evaluation on financial and medical data, feature-pool architectures that decouple generation from L1-regularised selection, and targeted mutation of individual features rather than whole-rubric regeneration.

Two implications for evaluation practice follow. First, aggregate accuracy can certify a classifier that is near chance for the wrong reason: the zero-shot LLM's 50.7% on HELOC reads as task difficulty, yet the model is heavily single-class-biased, and only per-class auditing separates the two cases. Per-class metrics should thus precede deployment, and a transparent representation makes that audit routine. Second, feature values come from an LLM judge, so a miscalibrated extractor can corrupt otherwise transparent logic; extraction accuracy needs validating separately from end-task accuracy.

References

[1] Nikhil Abhyankar, Parshin Shojaee, and Chandan K. Reddy. LLM-FE: Automated feature engineering for tabular data with LLMs as evolutionary optimizers. *arXiv preprint arXiv:2503.14434*, 2025.

[2] Vojtěch Balek, Lukáš Sýkora, Vilém Sklenák, and Tomáš Kliegr. LLM-based feature generation from text for interpretable machine learning. *arXiv preprint arXiv:2409.07132*, 2024.

[3] Milan Bhan, Yann Choho, Pierre Moreau, Jean-Noel Vittaut, Nicolas Chesneau, and Marie-Jeanne Lesot. Towards achieving concept completeness for textual concept bottleneck models. *arXiv preprint arXiv:2502.11100*, 2025.

[4] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

[5] Angelica Chen, David Dohan, and David So. EvoPrompting: Language models for code-level neural architecture search. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

[6] Dorottya Demszky, Dana Movshovitz-Attias, Jeongwoo Ko, Alan Cowen, Gaurav Nemade, and Sujith Ravi. GoEmotions: A dataset of fine-grained emotions. In *Proceedings of ACL*, 2020.

[7] FICO. FICO explainable machine learning challenge: HELOC dataset. https://community.fico.com/s/explainable-machine-learning-challenge, 2018. Home Equity Line of Credit default prediction dataset.

[8] Noah Hollmann, Samuel Müller, and Frank Hutter. Large language models for automated data science: Introducing CAAFE for context-aware automated feature engineering. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

[9] John R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, 1992.

[10] Jaris Küken, Lennart Purucker, and Frank Hutter. Large language models engineer too many simple features for tabular data. *arXiv preprint arXiv:2410.17787*, 2024.

[11] Xiaochong Lan, Jie Feng, Yinxing Liu, Xinlei Shi, and Yong Li. AutoQual: An LLM agent for automated discovery of interpretable features for review quality assessment. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: Industry Track*, 2025.

[12] Alexander Novikov, Ngân Vũ, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang, Adam Zsolt Wagner, Sergey Shirobokov, Borislav Kozlovskii, Francisco J. R. Ruiz, Abbas Mehrabian, M. Pawan Kumar, Abigail See, Swarat Chaudhuri, George Holland, Alex Davies, Sebastian Nowozin, Pushmeet Kohli, and Matej Balog. AlphaEvolve: A coding agent for scientific and algorithmic discovery. *arXiv preprint arXiv:2506.13131*, 2025.

[13] Cynthia Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1(5):206–215, 2019.

[14] Chandan Singh, Armin Askari, Rich Caruana, and Jianfeng Gao. Augmenting interpretable models with large language models during training. *Nature Communications*, 14(1):7913, 2023.

[15] Chung-En Sun, Tuomas Oikarinen, Berk Ustun, and Tsui-Wei Weng. Concept bottleneck large language models. *arXiv preprint arXiv:2412.07992*, 2024.

[16] Mathurin Videau, Alessandro Leite, Marc Schoenauer, and Olivier Teytaud. Evolutionary pre-prompt optimization for mathematical reasoning. *arXiv preprint arXiv:2412.04291*, 2024.

[17] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, volume 35, 2022.

[18] Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V. Le, Denny Zhou, and Xinyun Chen. Large language models as optimizers. In *Proceedings of ICLR*, 2024.

[19] Yue Yang, Artemis Panagopoulou, Shenghao Zhou, Daniel Jin, Chris Callison-Burch, and Mark Yatskar. Language in a bottle: Language model guided concept bottlenecks for interpretable image classification. In *Proceedings of CVPR*, 2023.

[20] Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems*, volume 28, 2015.

[21] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging LLM-as-a-judge with MT-Bench and chatbot arena. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

A Prompt templates

A.1 Feature extraction prompt

Feature Extraction

Extract the provided binary features from the text using the rubrics below. Output a list of exactly {num_features} values (0s and 1s, one per feature, in order) inside `<features></features>` tags.
Rubric:
1. {feature_1}
2. {feature_2}
...
Text:
{text}
Format:
`<features>`
0,0,...
`</features>`

A.2 Feature evolution prompt

The evolution prompt includes the task description, label set, a sample of labelled training examples, the current top-performing features with their classifier coefficients, error analysis from the previous iteration (confusion pairs and sample misclassifications), per-class feature activation rates, feature ablation scores marking each feature as essential, useful, or unnecessary, and any recently rejected rubrics. A simplified version is shown below; the full prompt is available in the code repository.

Feature Evolution (simplified)

Your task is to generate a rubric with 10 binary features which will be used as ML model features to determine a classification. Keep the top performing existing features and replace/iterate on the others to find new complementary features that will improve performance. We want generalisable features that transfer to unseen examples, not only training instances.
Each feature MUST be concrete and directly extractable: check for presence of specific words, phrases, named entities, or syntactic patterns. Do NOT use features that require counting, computing ratios, or subjective judgment. Think step-by-step about the evidence before generating your final rubric.
Problem Domain: {task_description}
Possible Labels: {labels}
Example input/output pairs:
{labelled_examples}
Top performing features:
{top_features_with_coefficients}
Previous iteration results:
CV accuracy: {cv_accuracy}
Most common confusions: {confusion_pairs}
Sample errors: {misclassified_examples}
Feature importance: {feature_coefficients}
Per-class activation rates: {activation_rates}
Feature necessity (ablation): {ablation_scores}
Replace features marked UNNECESSARY. Keep features marked ESSENTIAL unchanged.
Previously rejected rubrics:
{rejected_rubrics}

B Feature evolution analysis

Figure 5 shows the final evolved rubrics with features colour-coded by origin. The replacements reveal a consistent strategy: the LLM identifies which class pairs are most confused and proposes features that resolve those specific confusions. On the emotion task, generic indicators like "exclamation marks" fire across multiple classes and provide little discriminative signal. Evolution replaces them with targeted detectors: explicit sadness markers, sarcasm via negation patterns, rhetorical questions for surprise, and hostile second-person address for anger. Each is designed to separate a specific pair

of emotions that the initial rubric conflated. The retained features (positive markers, anxiety words, moral disapproval) survive precisely because they already map cleanly onto individual classes.

On HELOC, the zero-shot LLM performs at chance with a strong single-class bias because it cannot perform reliable quantitative reasoning over serialized numerical profiles, whereas evolution uses classifier feedback to identify which thresholds discriminate between risk classes. The evolved features are compound conditions (e.g., "Nr Inquiries In Last 6 Months ≥ 3 AND Nr Total Trades $\leq$ 25") that capture interaction effects between credit variables.

C Evolved rubric examples

D Detailed analysis

D.1 Feature-class coefficient heatmaps

Figures 6–7 show the logistic regression coefficients for each evolved feature (columns) and class (rows). Red indicates the feature increases the probability of that class; blue indicates it decreases it.

D.2 Confusion matrix comparison

Figures 8–9 compare the confusion matrices of the initial (zero-shot) rubric and the evolved rubric on held-out test data.

D.3 Feature necessity ablation

Figures 10–11 show the accuracy drop when each feature is removed from the evolved rubric. Red bars indicate features that contribute positively; green bars indicate features whose removal improves accuracy.

Table 4: Final evolved rubric for AG News (4-class topic classification) after 15 iterations. The initial rubric (iteration 0) was never improved upon. Features ordered by average absolute classifier coefficient.

#	Feature definition	$\lvert\bar{w}\rvert$
1	Contains sports-specific terminology (athlete names, team abbreviations, sport league names like NHL, NBA, or position/play descriptions)	0.854
2	Contains financial or market-specific terms (deficit, trade, stocks, revenue, earnings, merger, acquisition, deal)	0.622
3	Contains corporate or brand entity names (company names that are widely recognised business organisations)	0.576
4	Contains equipment, vehicle, or technology product names/models (gadgets, devices, software names, satellites, hardware)	0.470
5	Contains language of diplomatic conflict or legal action (accuse, sue, claim, testify, violation, treaty, embargo, sanction)	0.467
6	Contains scientific or technology terminology (algorithm, software, digital, innovation, research, quantum, genetic, climate, renewable)	0.441
7	Contains player/team performance statistics or game scores (numbers paired with athletic metrics like goals, assists, wins, points)	0.339
8	Mentions specific scientific studies, research findings, or experimental results with descriptive outcomes	0.122
9	Contains references to international locations, countries, or geopolitical entities	0.117
10	Contains entertainment industry references or Hollywood/media production terminology	0.078

Table 5: Evolved rubric for GoEmotions (7-class emotion). Features target specific emotion categories rather than generic sentiment indicators.

#	Feature definition	$\lvert\bar{w}\rvert$
1	Contains anxiety/worry-related words: nervous, worried, anxious, scared, terrified	0.384
2	Contains explicit sadness or loss markers: sad, disappointed, loss, heartbroken	0.370
3	Contains positive outcome or satisfaction markers: works, good, great, nice, love	0.362
4	Contains explicit moral disapproval or dismissal: freaky, weird, inappropriate, awful	0.347
5	Contains rhetorical question structure without interrogative intent	0.221
6	Contains unfamiliarity discovery markers: didn't know, never knew, how is, what is	0.168
7	Contains direct second-person hostile address: you're, you people, you all	0.167
8	Contains negation followed by positive words indicating sarcasm: not great, not nice	0.154
9	Contains casual dismissal or eye-roll language: yeah right, sure, whatever, ok	0.115
10	Contains regret or nostalgia markers: miss, used to, back when, wish I had	0.108

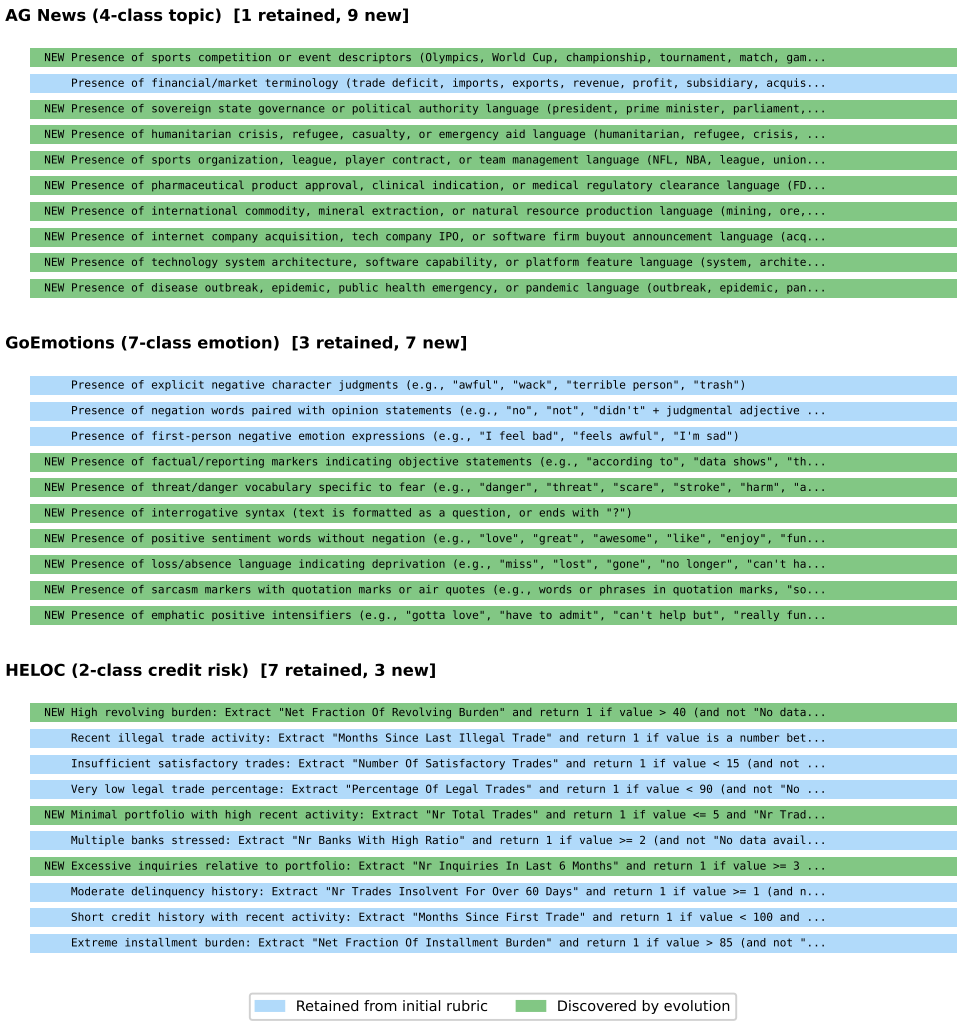


Figure 5: Evolved rubrics for all three datasets, showing which features were retained from the initial rubric (blue) and which were newly discovered by evolution (green). Evolution replaces generic features with task-specific discriminators, with the degree of replacement reflecting task difficulty.

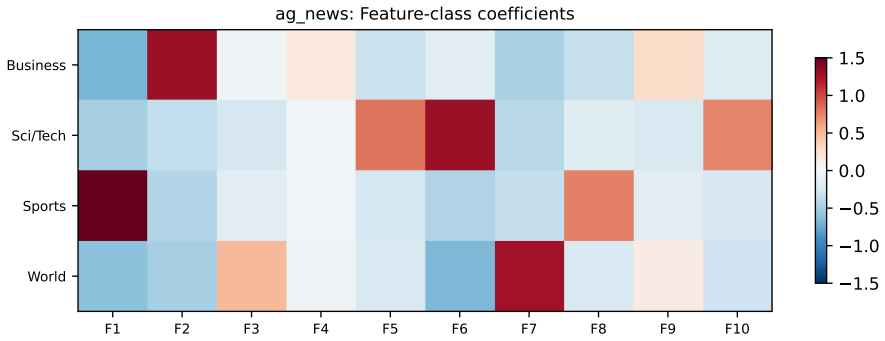


Figure 6: AG News: feature-class coefficients.

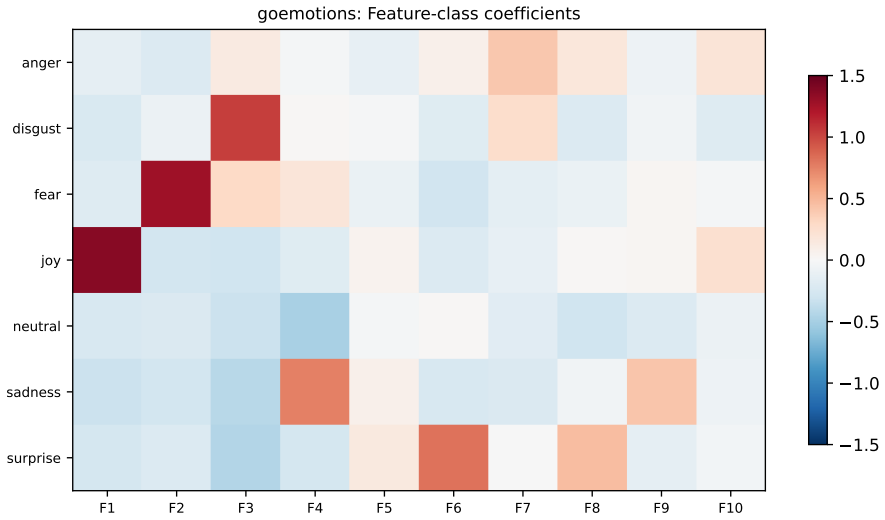


Figure 7: GoEmotions: feature-class coefficients.

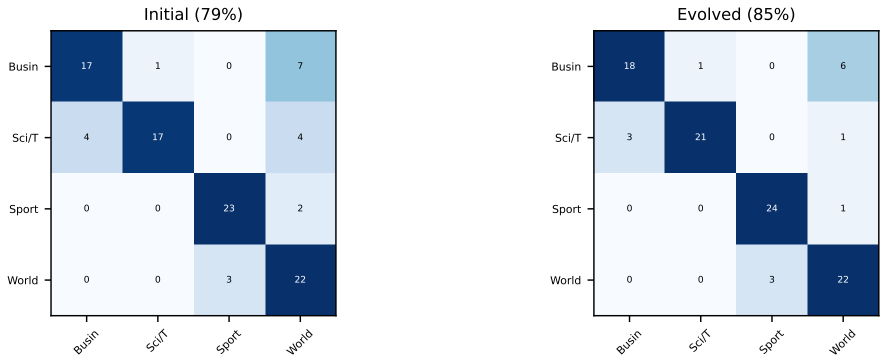


Figure 8: AG News: confusion matrices (initial vs evolved).

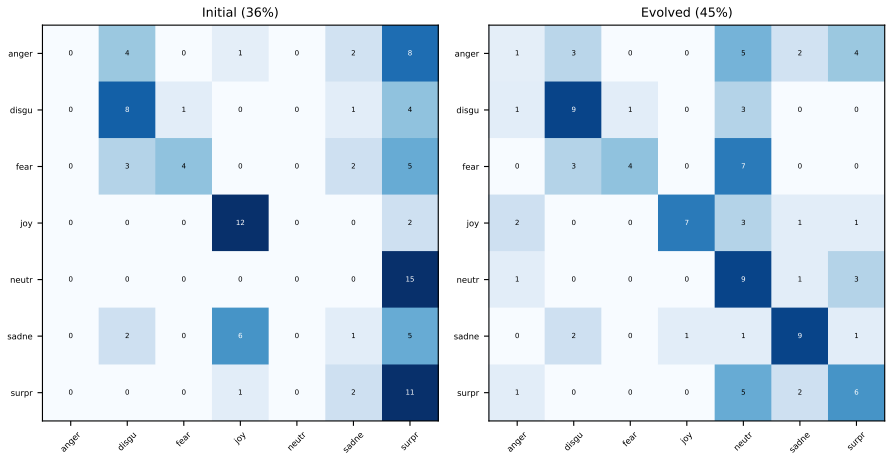


Figure 9: GoEmotions: confusion matrices (initial vs evolved).

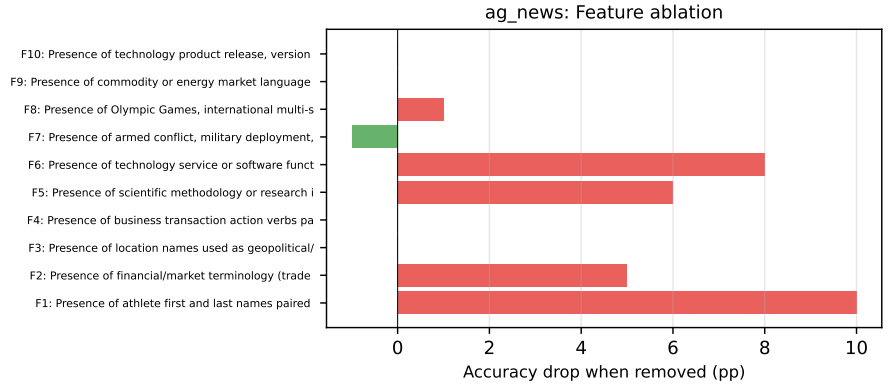


Figure 10: AG News: feature ablation.

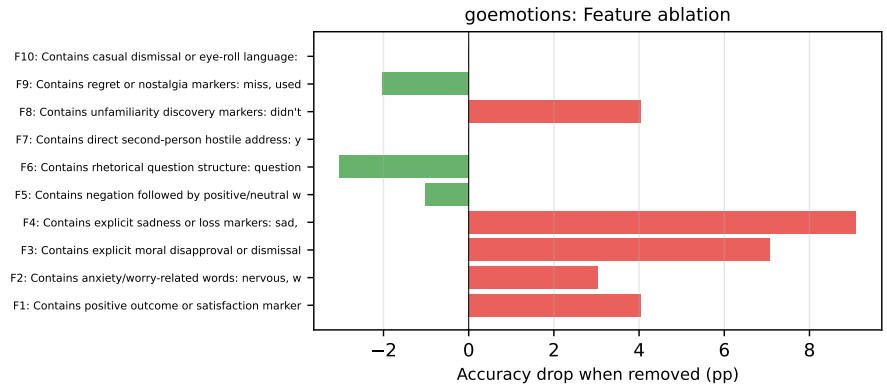


Figure 11: GoEmotions: feature ablation.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The claims in the abstract and introduction (a +2.9 pp average gain over single-shot rubrics, outperforming the zero-shot LLM on two of three tasks, and the zero-shot LLM's chance-level single-class-biased performance on HELOC) are all substantiated by the results in Section 4.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Limitations are discussed in Section 6: little benefit when a single-shot rubric is already near-optimal, the information bottleneck of binary features, and reliance on a single LLM.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [N/A]

Justification: The paper presents an empirical framework and contains no formal theoretical results.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: Section 3 and Section 4 specify the algorithm, datasets, train/test splits, and all hyperparameters, and Appendix A provides the prompt templates needed to reproduce the results.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in

some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: We do not release source code. Reproducibility is instead supported by other means: all three datasets (AG News, GoEmotions, HELOC) are publicly available and cited, and we provide the full algorithm, train/test splits, all hyperparameters, and the prompt templates (Appendix A) needed to reproduce the results.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: The experimental configuration in Section 4 reports sample size, splits, number of iterations, cross-validation folds, feature count, classifier regularisation, and the LLM used.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Tables in Section 4 report mean $\pm$ standard deviation across three independent stratified splits.

   Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Cost is dominated by LLM API calls, reported per training run and per inference in Table 2; the logistic-regression classifier is trained on 500 samples and is computationally negligible.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research uses only public benchmark datasets and conforms with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: Broader impacts are discussed in Section 6: improved auditability for regulated decisions, alongside risks of over-reliance on an LLM feature extractor and increased API cost.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: The paper releases no high-risk pretrained models or scraped datasets.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: All datasets and the LLM used are cited in Section 4; the datasets are standard, publicly available benchmarks.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: The paper does not release new datasets or models; evolved rubrics are reported in the appendix for illustration only.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing or research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: The paper does not involve research with human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [Yes]

Justification: LLMs (Claude Haiku 4.5) are a core component as the feature generator and evaluator, and are fully described in Section 3.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.