

AI-Based Measurement of Innovation: Mapping Expert Insight into Large Language Model Applications

Robin Nowak (University of Passau), Patrick Figge (University of the Bundeswehr Munich),
Carolyn Häussler (University of Passau)*

Abstract

Measuring innovation often relies on context-specific proxies and expert evaluation. Hence, empirical innovation research is often limited to settings where such data is available. We investigate how large language models (LLMs) can be leveraged to overcome the constraints of manual expert evaluations and assist researchers in measuring innovation. We first design an LLM framework that reliably approximates domain experts' assessment of innovation from unstructured text data. We then evaluate the performance and broad applicability of this framework through two demonstration studies in different contexts: (1) innovation in software application updates and (2) user-generated ideas in product reviews. We compared the performance (F1-score) and reliability (consistency rate) of our LLM framework against alternative measures used in prior innovation studies, and to state-of-the-art machine learning- and deep learning-based models. The LLM framework achieved higher F1-scores than the other approaches, and its results are highly consistent (i.e., results do not change across runs). This article equips R&D personnel in firms, as well as researchers, reviewers, and editors, with the knowledge and tools to effectively use LLMs for measuring innovation and evaluating the performance of LLM-based innovation measures. In doing so, we discuss the impact of important design decisions—including model selection, prompt engineering, training data size, training data distribution, and parameter settings—on performance and reliability. Given the challenges inherent in using human expert evaluation and existing text-based measures, our framework has important implications for harnessing LLMs as reliable, increasingly accessible, and broadly applicable research tools for measuring innovation.

Keywords: innovation measurement, expert evaluation, large language models, artificial intelligence, text analysis

* Please send correspondence to robin.nowak@uni-passau.de.

We are grateful for very helpful comments from Natalie Carlson, Julian Just, Matt Marx, Milan Miric and Thomas Widjaja, and participants of the 21st Open and User Innovation Conference, OpenAI DevDay 2024, the NBER i3 Technical Working Group Meeting, and the 86th Annual Meeting of the Academy of Management as well as seminars at Colorado State University and the University of Passau. Thanks also go to Daniel Groell and Charlotte Musso for excellent research assistance. We gratefully acknowledge funding by the German Research Foundation—project number 443732464. Generative AI tools were used to assist with coding and language editing; we thoroughly checked all AI-assisted work and take full responsibility for the manuscript's content.

the support of experts recruited to assist in a specific project to manually assess and code innovation activities and outcomes, which severely limits sample size.

In response, scholars have begun to adopt natural language processing (NLP) and machine learning (ML) techniques to create alternatives for constructing quantitative variables based on textual data (Gentzkow et al., 2019; Miric et al., 2023a). While often and successfully implemented (e.g., Bellstam et al., 2021; Miric et al., 2023b), these techniques come with substantial limitations. First, many NLP techniques depend on unambiguous, structured language for accurate performance (e.g., Loughran and McDonald, 2011). Second, while ‘classical’ ML approaches can tolerate linguistic noise, they generally rely on large sets of expert-labeled training data to yield reliable outcomes (Hirschberg and Manning, 2015). In contexts where the nature of the data and the construct to be measured involves understanding nuanced, multi-dimensional information, these approaches struggle to perform on par with human expert evaluation (Gentzkow et al., 2019).

Large language models (LLMs) may enable us to overcome some of these limitations. Built upon the transformer architecture (Vaswani et al., 2017) and pre-trained on vast text corpora, LLMs are able to accurately capture many subtleties of human language. The latest generation of LLMs bring enhanced predictive capabilities while requiring little to no additional data to perform tasks (Brown et al., 2020). They can be instructed by written human language, so-called prompts, that determine models’ output generation. Outputs are often remarkably human-like across scenarios (Jones and Bergen, 2024), with LLMs even capable of adopting different viewpoints in experimental settings (Boussiou et al., 2024; Doshi et al., 2025). These advancements might allow the analysis of text data in novel ways: LLMs may be able to capture and mimic the experts’ evaluation process, holding the potential to scale expert evaluation with unprecedented performance. This way, they can democratize the analysis of unstructured text data to a wider

audience of scholars by lowering barriers to use, e.g., by having little data requirements and reducing the need for repeated manual assessment.

In this paper, we introduce and empirically evaluate a framework for how to develop an LLM application to measure innovation-related constructs based on unstructured text data that mimics how an expert would have evaluated these constructs. Recognizing that evaluations yield the most accurate results when informed by specialized expertise (Böttcher and Klingebiel, 2025; Csaszar and Eggers, 2013), our LLM framework involves a structured process to capture the evaluation resulting from experts’ reasoning and decision-making steps. First, we inductively identify content dimensions that indicate innovation in our contexts, which we then validate with domain experts to establish the foundation for prompt design. Second, we move beyond the default capabilities of the pre-trained model by fine-tuning it with expert-labeled training data. Third, we address the reliability of results in repeated analyses by testing variations in design decisions including prompting, model selection, fine-tuning data, and parameters.

We evaluate the performance and broad applicability of this framework empirically through two demonstration studies. In Study 1, we measure the innovation-related outcomes from product development activities in the context of software application updates. In Study 2, we identify user-generated ideas for product development activities in the context of product reviews. Both demonstration studies traditionally require methods relying heavily on human-coded labels due to the inherent complexity of the constructs being analyzed. Moreover, in both studies, the textual data serving as the basis for evaluation is highly nuanced, context-specific, and lacks clear structures or standardized guidelines, making outcomes such as updates or reviews subject to subtle linguistic variations.

In both demonstration studies, our LLM application outperformed (in terms of F1-scores) innovation measures recently used in scholarly work as well as established NLP techniques. When

fine-tuning with expert-labeled training data is possible, LLMs achieve their highest performance, as this data allows for targeted mitigation of biases—such as those stemming from structural misclassifications or failed classifications of texts—and ensures high reliability independent of design decisions. But even without expert-labeled training data, LLMs still achieve strong performance and consistency. Overall, our findings reveal that our LLM framework is highly adjustable to both the measurement objective and context, provided that specialized human experts are available to define measurement criteria and assess its accuracy on validation data.

Our work makes two main contributions. First, we add to an emerging line of research that applies LLMs as research tools to construct variables based on text data (Carlson and Burbano, 2026; Cheng et al., 2024; de Kok, 2025; Fyffe et al., 2024; Just et al., 2024). Our framework presents a low-barrier approach for researchers to access reliable expert-level evaluations in various innovation domains. Specifically, we provide a systematic process that guides researchers and R&D personnel in firms through a structured prompt design, fine-tuning, and validation of a pre-trained LLM informed by validated domain-specific content dimensions to create measures of interest. Researchers wishing to adapt our framework to their own context need only to define the construct of interest by identifying key indicators, collaborate with a few specialized experts for validation, and then follow our guidelines. Second, our demonstration studies introduce novel, high-quality measures that enable more precise estimations of innovation-related constructs in software application development and product review contexts. They offer researchers and R&D personnel in firms a robust alternative to traditional proxies and human expert assessments. The respective code, local models, training and validation datasets, and fine-tuning approaches are all made public in an online repository¹ to support replication and adaptation for future research.

¹ All data, code, and supplementary materials needed to reproduce our results are openly accessible in the project's GitHub repository (https://github.com/robi979/AI-Innovation-Measurement_Extension). We also provide a

2. Expert Evaluation in Innovation Research

Measuring innovation-related constructs requires clarifying which aspect of innovative activity is meant to be captured. Prior literature has differentiated innovative activity according to the type and degree of change involved. For instance, innovations may preserve or disrupt existing technological and market linkages (Abernathy and Clark, 1985), differ in whether they alter a product's components or the architecture linking them (Henderson and Clark, 1990), or emerge from searching for unprecedented combinations of existing knowledge (Fleming, 2001). Reviews of this literature stream synthesize these perspectives by emphasizing that innovation can be conceptualized along multiple dimensions, including whether it concerns a process or an outcome, whether it occurs at the product, process, organizational, or market level, and whether it represents a minor improvement or a more substantial departure from existing practice (Crossan and Apaydin, 2010; Garcia and Calantone, 2002). These perspectives illustrate that innovation-related constructs are context-specific operationalizations of particular aspects of innovative activity. Accordingly, the measurement of such constructs rests on identifying the aspect of innovative activity that is theoretically relevant in a specific setting and the observable cues through which this aspect manifests in the available data.

Because innovation-related constructs are rarely directly observable and their cues are not self-evident, their measurement often depends on expert judgment. This helps explain why scholars frequently rely on measures whose institutional characteristics already embed elements of such judgment. Patents, in particular, are by legal definition related to inventiveness and undergo a rigorous examination process to ensure novelty, non-obviousness, and industrial applicability

lightweight 'application package' containing the essential code and example notebooks that show how to run our framework directly through common LLM-provider APIs.

(Title 35 United States Code), relying on specialized, highly skilled professionals such as patent examiners of patent offices to evaluate technical details and legal requirements. Extending this level of rigor to other spheres of innovation is challenging. Unlike patents, many forms of innovation lack formalized frameworks, legal structures, or dedicated expert evaluators. Still, reliable construct measurement often demands a clear understanding of the construct, deep domain knowledge and well-developed cognitive frameworks to comprehend, organize, and interpret construct-relevant information—expertise that is both scarce and difficult to scale.

This constraint has become increasingly salient as the growing availability of text data unlocks new opportunities to gain insights into innovative activities. On the one hand, organizations can now easily access broad pools of ideas from several sources, which allow them to source novel solutions to innovation challenges from both internal and external contributors (Boudreau et al., 2011; Jeppesen and Lakhani, 2010). On the other hand, such organizations themselves generate much text data, providing opportunities to detect patterns in their innovative activities beyond what is captured by traditional measures like R&D intensity or patent-related outputs (Bellstam et al., 2021). Exploiting these new data sources, however, requires expert judgment to identify construct-relevant information and evaluate it consistently.

Even when experts are available, scaling their evaluation is inherently constrained by time or budget constraints and human cognitive limits. Manually processing large volumes of complex information and identifying meaningful construct-relevant patterns is tedious and time-intensive. Evaluators often struggle to manage the complexity of vast (multi-dimensional) datasets, particularly when faced with uncertain aspects that cannot be easily connected to existing knowledge structures. Psychology research highlights that when overwhelmed with too much information at once, humans fail to organize and interpret the data thematically, leading to

suboptimal decision-making (Tversky and Kahneman, 1974). These cognitive limits hinder the ability to process complex information effectively and impede innovation evaluation at scale.

The difficulty of processing large datasets not only constrains evaluators’ ability to assess information but also narrows their scope of attention (Samuelson and Zeckhauser, 1988). Faced with complex or uncertain information at scale, evaluators often resort to heuristics, reinforcing existing biases or adhering to prevailing concepts (Tversky and Kahneman, 1974). Their attention space then tends to align with established knowledge structures, causing them to overlook unconventional information that falls outside their expertise (Boudreau et al., 2016; Piezunka and Dahlander, 2015). For instance, evaluators may judge the value of new unfamiliar information based on prior decisions rather than its intrinsic merits, potentially explaining why established technologies persist even when superior alternatives exist (Abrahamson, 1991). Taken to the extreme, this tendency to favor the status quo may result in the outright rejection of potential innovations when increased workload overwhelms evaluators’ cognitive resources.

However, thorough evaluation is essential for measuring valid constructs, particularly when leveraging the growing availability of large datasets to explore hitherto less researched contexts. Advances in NLP techniques, driven by progress in AI research, now present an opportunity to overcome constraints imposed by human cognitive limits in processing information (Korinek, 2023). And in fact, first studies have demonstrated how these novel techniques can complement or even substitute human capabilities in generating measures (e.g., Choudhury et al., 2019; de Kok, 2025; Fyffe et al., 2024; Miric et al., 2023a), decision-making (Doshi et al., 2025), and idea selection (Just et al., 2024), paving the way for a deeper understanding of how these tools can be utilized to enhance both research and practice. An overview of prior text-based approaches for approximating expert evaluation is provided in Online Appendix A.

3. Mapping Expert Insight into Large Language Model Applications

Evaluations in innovation contexts often place a strong emphasis on qualitative judgments such as assessing an idea’s novelty (Boudreau et al., 2016; Lane et al., 2024). Such judgments are inherently interpretive, drawing upon individual perceptions (Kim and DellaPosta, 2022; Lane et al., 2022), intuition (Huang and Pearce, 2015), and expertise (Boudreau et al., 2016; Li, 2017), and typically lack universally accepted benchmarks at the time of evaluation. Hence, selecting appropriate evaluators is essential for high evaluation quality and reliability.

Research on innovation evaluation indicates that uncertainty in evaluations is minimized when domain experts can be involved (Böttcher and Klingebiel, 2025; Boudreau et al., 2016; Csaszar and Eggers, 2013). Through years of learning and deliberate practice, experts develop distinctive know-how that enables them to, often intuitively, recognize intricate patterns and extract meaningful insights from complex information (Ericsson and Smith, 1991; Johnson et al., 1982; Kahneman et al., 1982). As a result, they exhibit better judgment, allowing them to assess tasks and contextual cues more accurately than nonexperts and ultimately make more informed evaluations in their domain.

At the same time, this raises the question of what qualifies an evaluator as a domain expert for evaluation tasks. Research on experts suggests that the validity of expert judgment is a function of the fit between the evaluator’s domain of practice and the characteristics of the judgment task and not a general attribute of credentials or seniority (Shanteau, 1992). Expert judgment is most reliable when the task environment exhibits learnable regularities and the evaluator has accumulated repeated, feedback-rich experience with those regularities (Ericsson and Smith, 1991; Kahneman et al., 1982). For text-based construct measurement, the relevant expertise therefore concerns the data-generating process behind the analyzed texts, meaning that evaluators need to

command the conventions and vocabulary of those who produce these texts, while familiarity with the theoretical construct alone is not sufficient. Conversely, when an evaluator’s domain of practice and the judgment task are misaligned, evaluation quality can degrade below that of generalist evaluators (Böttcher and Klingebiel, 2025; Csaszar and Eggers, 2013). Selecting evaluators thus involves three considerations: (i) defining *ex ante* which judgment task the construct of interest requires, (ii) choosing evaluators whose demonstrable, current practice aligns with the data-generating process underlying the text data, and (iii) documenting the recruitment and qualification criteria as part of the research design.

In this study, we build on these considerations and aim to integrate the distinctive know-how of appropriately selected domain experts into LLM-based applications, so that researchers can apply them as reliable tools for approximating expert evaluation. Although LLMs excel at extracting insights from large datasets and can generate assessments that are consistent with human judgment (Brown et al., 2020), it is essential to recognize that they operate on statistical pattern recognition and not genuine logical reasoning or expertise-based intuition (Mitchell and Krakauer, 2023). This reliance on probabilistic modeling often results in variability in outputs across runs. To address this, we suggest a systematic mapping of the cues and implicit criteria that experts draw upon in their judgments and evaluations. By explicitly codifying these aspects in the input options of an LLM, it becomes possible to steer its internal probabilities, enabling outputs to more closely mirror expert-level reasoning and assessments (Liu et al., 2023).

We begin with outlining a stylized framework to discern cues indicative of expert judgment (Einhorn, 1974). This involves initially *identifying relevant cues and information* by clarifying the evaluation objective and criteria and engaging with the available data to isolate evaluation-relevant cues. Next, the identified cues are *organized into clusters or dimensions* by contextualizing each through its connection to existing knowledge. This allows us to *assess the amount of each cue* as

it is reflected in the data element (e.g., a certain text) being evaluated. Finally, the organized and assessed *cues are weighed* according to their importance relative to the evaluation criteria, ultimately informing the final judgment. It is important to recognize that expert judgment often appears automatic and even instantaneous due to experts’ deeply internalized knowledge structures (Chase and Simon, 1973; Dane and Pratt, 2007; Ericsson and Smith, 1991; Sukhov et al., 2021). Such judgments may draw on learned, tacit conceptions of similarity that enable experts to recognize patterns that are difficult to reduce to explicit information-processing steps (Nightingale, 1998). Hence, it is unrealistic to assume that all experts consciously and explicitly follow the outlined steps. Still, this stylized framework assists in a structured identification of relevant cues indicative of expert judgment, enabling researchers to instruct an LLM to generate outputs that better reflect expert reasoning.

To map this framework into LLM applications, we create a prompt-data-output combination that mirrors the results of expert reasoning. In other words, rather than trying to replicate the complex and often implicit nature of expert reasoning, we embed relevant evaluation cues within the LLMs’ input options. LLMs operate by generating text token by token, with each token selected from a probability distribution that is conditioned on the input prompt, model tuning parameters (such as temperature settings), and, when available, fine-tuning data. When we craft prompts that mirror the described stylized LLM framework for an evaluation task—starting with a clear task definition, moving to the identification, clustering, and assessment of relevant cues, and finally weighing these cues relative to evaluation criteria—we can effectively steer the model’s internal probabilities toward outputs that reflect those from expert-level reasoning (Liu et al., 2023). Furthermore, fine-tuning the model with domain-specific, labeled data can further reinforce this alignment.

This mapping implies a division of roles between human experts and an LLM. Expert judgment informs the framework at the design stage, where experts define the construct and its indicative cues, validate the inductively derived data structure, weigh the relevance of its dimensions, and adjudicate the labels against which model outputs are validated. The LLM in turn executes this codified judgment by repeatedly applying the expert-defined evaluation scheme to large volumes of text. The division of roles also shapes the implementation effort of the framework. Human effort is concentrated in the iterative design and validation stages, with the required effort depending on the iterations needed to reach saturation and expert agreement and, where fine-tuning is used, on the amount of labeled training data required. Once a validated configuration is established, scaling the measure to additional observations requires little additional human involvement. Notably, the large-scale application of expert-developed evaluation schemes has traditionally rarely been performed by experts themselves. For instance, in quantitative content analysis, applying an expert-developed coding scheme at scale is routinely delegated to trained human coders whose reliability is monitored through inter-coder agreement statistics (Krippendorff, 2019).

In substituting for such non-expert coding work, the LLM takes on a role closely related to research that employs LLMs as simulacra of humans, for instance to augment survey data (Brynjolfsson et al., 2026). Operationally, these applications proceed much like our framework, conditioning a model on structured prompts and human-provided information so that its outputs approximate the judgments of individuals. Our framework also relies on this capability at the execution stage. What converts it into an expert-aligned measure is the apparatus built around it, as the execution is embedded in the codified evaluation scheme and the labeled fine-tuning data that reflect expert reasoning.

----- Insert Table 1 about here -----

Table 1 illustrates the framework for designing LLM inputs to measure constructs from unstructured text data, mimicking how an expert would have processed the evaluation. The framework begins with the researcher explicitly defining the task and desired output in the prompt’s opening section, aligning it with the specific measurement objectives.

Next, the LLM framework leverages inductive coding of data (Gioia et al., 2013; Strauss and Corbin, 1998) until thematic and meaning saturation are reached. This approach ensures that the prompt captures the critical cues that experts use to filter and prioritize information. In addition, relevance weights can be integrated directly into the prompt or applied during post-processing. This step is crucial because it aligns the model’s output with expert-informed priorities and reflects the key elements of the data structure relevant for the measured construct. It is imperative that experts are directly involved or iteratively consulted during this phase to validate the final data structure, thereby laying the groundwork for a robust coding scheme—especially if fine-tuning with further training data is planned.

The final stages of the LLM framework involve validating the LLM’s output against a holdout sample and, whenever possible, other established metrics. Based on performance outcomes, fine-tuning with labeled training data can be applied and design decisions may have to be adjusted. Researchers and R&D personnel need to systematically document decisions such as model selection², prompt engineering techniques (Wei et al., 2022), temperature settings (Peeperkorn et al., 2024), and fine-tuning configurations (Gao et al., 2020) to ensure reliability.

² Since 2024, reasoning models (e.g., OpenAI’s o-series) have emerged as a class of LLMs that generate intermediate reasoning tokens before producing an output. These models internalize a multi-step procedure similar to the one our framework specifies explicitly, with the difference that the generated reasoning is model-derived, typically acquired through reinforcement learning on tasks with verifiable solutions such as mathematics and code, and thus does not necessarily represent the expert reasoning required for the task. Because providers expose this reasoning only in truncated form, it cannot itself be verified, and validation must rest on the outputs alone. As fine-tuning access is currently unavailable for these models, they enter our framework as an alternative model choice in default configuration, subject to the same validation standards as any other model. We discuss the implications in our Demonstration Studies in Section 4 and the corresponding Online Appendix.

For instance, tuning parameters like temperature allow control over the randomness of token selection: lower temperatures favor high-probability tokens for more deterministic outputs, while higher temperatures facilitate more varied outputs.

Understanding the implications of such specific decisions is crucial for optimizing both performance and consistency, ensuring that the model’s outputs remain reliable across repeated runs. Similar to human expert evaluation, reliability measures such as intra-rater reliability (consistency within individual repeated assessments) and inter-rater reliability (consistency among multiple evaluators) are prerequisites for valid judgments, confirming the robustness of the evaluation process and derived conclusions (DeVellis and Thorpe, 2022; Einhorn, 1974).

Because expert judgments serve as the standard against which the LLM application is calibrated and validated, their reliability requires the same systematic treatment as any other design decision. We recommend that the labeling of training and validation data involve at least two raters and that inter-rater reliability be reported against conventional benchmarks (Krippendorff, 2019; Landis and Koch, 1977). Disagreement between raters should not be treated as noise to be resolved ad hoc; it is diagnostic information about ambiguity in the construct definition or the data structure. Recurring disagreement therefore feeds back into the refinement of the coding scheme, which is iterated until meaning saturation and stable inter-rater agreement are jointly reached (Gioia et al., 2013), with a documented adjudication rule governing residual cases.

Two implications follow. First, human inter-rater agreement constitutes the ceiling on validated model performance, since an LLM application cannot be meaningfully validated beyond the reliability of the benchmark against which it is assessed. Second, the expert-adjudicated holdout sample provides the standard that disciplines evaluator-related researcher degrees of freedom. A different selection of experts could, in principle, yield a different coding scheme. The framework

makes this choice testable as any alternative evaluator configuration constitutes a rival specification whose adequacy can be assessed against the same validation logic.

4. Demonstration Studies

4.1. Demonstration Study 1: Innovation in Software Application Updates

In the first demonstration study, we apply our systematic framework for innovation measurement to analyze and classify software application updates. Software application development is inherently generative, allowing for continuous product evolution through optimization, reuse, and novel combination of digital elements even after initial launch (Zittrain, 2006). Digital platforms enhance this process by providing infrastructure and tools that reduce costs and accelerate development, fostering rapid experimentation and sequential innovation (Parker and Van Alstyne, 2018). Low barriers to entry cultivate active developer communities and intense competition, leading to fast innovation cycles and early product releases (James et al., 2023; Lee and Raghu, 2014). However, these same dynamics also incentivize the frequent release of low-quality updates or updates with little improvement to the software application and without introducing novel elements (Comino et al., 2019). When studying innovation behavior in many digital contexts, it is therefore crucial to differentiate between updates that merely refine existing products or “fix bugs”, and those that contribute genuine innovation.

Recent methods for measuring innovation in software application updates include analyzing version numbers (Boudreau, 2012; James et al., 2023; Wen and Zhu, 2019) and release notes (Foerderer et al., 2018; Kircher and Foerderer, 2024; Leyden, 2022). Relying on version numbers assumes that significant numerical changes indicate major updates, but developers often have incentives not to adhere to versioning protocol conventions³, potentially distorting such

³ See <https://semver.org/> for a popular example of such conventions.

metrics. Text analysis of release notes improves upon this issue but is mostly conducted using dictionary-based approaches or ‘classical’ machine learning methods. However, the vast heterogeneity in how developers describe updates and the absence of standardized terminology make it challenging to capture all variations accurately. Since datasets in such settings can be extremely large—often involving up to millions of descriptive texts—relying on human experts becomes impractical, albeit necessary, underscoring the need for advanced methods that can accurately assess innovation in software updates.

4.1.1. Data

An important aim of this demonstration study is to evaluate our proposed LLM framework and compare it to methods used in previous studies. To achieve this, we require a coded sample of software application updates, specifically requiring knowledge about the update and classification of its content. For comparability with previous studies, our data must include update information reported in release notes as well as the corresponding version numbers.

We obtained the data for this demonstration study from MixRank, a provider of app data. From a large dataset covering the complete update history of every app active on the Apple App Store between 2018 and 2021, we randomly sampled 4,000 update descriptions⁴ to form a pool for creating training and validation datasets. To ensure consistent analysis, we filtered the update descriptions to include only those written in English.⁵ Each update description contains a corresponding release note and version number.

⁴ On the App Store, these release notes are displayed in a section titled “What’s New” and limited to 4,000 characters. Since 2018, it is mandatory for developers to specify update details and avoid generic texts.

⁵ Our demonstrations use English-language text to ensure comparability with dictionary and classical ML baselines. The framework itself is not language-specific as LLMs show increasing performance on multilingual datasets (e.g., OpenAI, 2023). Online Appendix K discusses its application to multilingual corpora.

To establish ground truth for the sample pool, a co-author and a research assistant coded the release notes in the pool according to a coding scheme developed by the research team through inductive coding. This scheme comprises seven content dimensions (classes) and was validated in interviews with software application developers. A subset of $n=1,000$ update descriptions was independently coded by both raters to ensure high inter-rater reliability (observed agreement = .91, $\kappa = .87$). Any discrepancies were discussed to be consistent across the remaining sample pool.

Subsequently, we generated two random subsamples from the labeled sample pool of $n=4,000$ release notes: a validation set of $n=1,000$ release notes serving as a holdout and a training set of $n=2,000$ release notes. Both subsamples were constructed to reflect a representative class distribution of the overall pool. The remaining $n=1,000$ release notes are used in additional analyses to allow variations with class distributions.

4.1.2. Research design

We anchored our measurement objective in the literature’s conceptualization of an innovative update in mobile software applications. In this context, innovative updates are defined as those that introduce additional, user-salient functionality or content to the app, rather than “under the hood” changes made primarily for internal purposes and imperceptible to consumers (Foerderer et al., 2018; Leyden, 2022). In other words, the primary task of our prompt is to extract and analyze the content of release notes which we then use for an informed decision whether an update is innovative according to this definition.⁶

In a next step, we conduct inductive coding of release notes until reaching meaning saturation. This iterative process yielded seven content dimensions that characterize the data

⁶ We acknowledge that back-end modifications such as performance improvements, security hardening, or code refactoring may also be labeled “innovative” when viewed as process innovations. Our study, however, focuses explicitly on consumer-visible product innovations that add new functionality or content.

structure of release notes and enable us to distinguish between updates reflecting innovative activity (dimensions 1 and 2) and those intended primarily to fix bugs, introduce minor improvements, or promote the app (dimensions 3-7; cf. Online Appendix B1). To validate our measurement objective and data structure, we conducted several interviews with software application developers (cf. Online Appendix B1).

----- Insert Figure 1 about here -----

Following our proposed framework, both inform the prompt that maps the process of expert reasoning into an LLM-compatible format (cf. Figure 1). The validated content dimensions build the foundation for assigning human-coded labels to the release notes in our sample pool. Assigning these labels also serves as an additional check on data structure saturation, confirming that no further coding dimensions are necessary.

Our framework is applied to LLMs of various sizes, versions, and providers—both in their default state and after fine-tuning—with all models accessed via provider APIs. For fine-tuning, the training data is prepared by integrating the release note text with its corresponding labeled output via our designed prompt. The generated outputs are evaluated against our validation dataset and benchmarked against the performance of methods from previous studies as well as ‘classical’ machine learning models, deep-learning approaches, and transformer-based pre-trained language models on the same validation dataset. The inference workflow and the output processing procedure are documented in Online Appendix C.

To replicate prior work, we incorporated the preprocessing steps, dictionaries and filters described in Agarwal and Kapoor (2023) and Kircher and Foerderer (2024) and we applied version logic similar to that used in Wen and Zhu (2019).⁷ For benchmarking the other models, we trained

⁷ Kircher and Foerderer (2024) classify an update as innovative (feature update) if the release note contains at least one of the keywords “new”, “added”, “upgrade”, or “major”. Agarwal and Kapoor (2023) classify an update as

each with the same labeled training data as for our LLM fine-tuning, albeit without the prompts. To identify the best-performing model setup, we conducted several iterations with different embedding approaches and hyperparameter settings. For each combination of embedding technique and classifier, we systematically optimized model hyperparameters via randomized search and stratified cross-validation across multiple dataset partitions. This entailed testing a range of embedding and model-specific configurations to robustly identify optimal setups for each method (Goodfellow et al., 2016). The full hyperparameter grids, search procedures, and rationale for model selection are described in detail in our Online Appendix D.

4.1.3. Results

Table 2 reports performance (Accuracy, F1-score, Precision, Recall) and reliability metrics (Consistency Rate) for identifying innovative updates, comparing (1) rule-based and dictionary-based approaches from prior literature, (2) ‘classical’ machine learning (ML) models, (3) neural architectures such as convolutional neural networks (CNN), (4) earlier transformer-based pre-trained language models (e.g., BERT), and (5) our framework applied to several large language models (LLMs) from different providers, with varied parameter sizes, and fine-tuning state. Accuracy measures the overall proportion of correct predictions, providing a straightforward assessment of model performance. However, to account for class imbalances and differentiate between false predictions, we use the F1-score, a standard composite metric calculated as the harmonic mean of Precision and Recall. Precision indicates the proportion of predicted innovative

innovative (new functionality) if the release note exceeds 200 characters or contains at least one of the keywords “introduce”, “feature”, “support”, “performance”, “improve”, “enable”, “update”, “enhance”, “modify”, “optimize”, “fast”, “adjust”, or “multitask”. To account for differences in text representation and ensure consistency in the keyword matching process, we applied several preprocessing steps. We converted each release note to lowercase, tokenized it into individual words, and each word was reduced to its stem. The keywords in each dictionary were processed similarly. In Wen and Zhu (2019), the baseline treats every update release as an innovative update. A version number rule is applied in robustness checks, classifying an update as innovative (major update) if the <major> field in the app’s version number (first digit) changed. When replicating this approach, we additionally controlled for changes in the <minor> field in the app’s version number (second digit).

updates that are actually correct, while Recall measures the proportion of actual innovative updates correctly identified. Additionally, we evaluate model stability using the Consistency Rate (CR), which quantifies how reliably a model produces the same predictions across different runs.

----- Insert Table 2 about here -----

Relying on version numbers (i.e., distinguishing whether the “first digit” or the “second digit” of the version changed) alone yields low F1-scores (below .30 for the “Version First Digit” heuristic), confirming prior concerns that app developers do not consistently follow versioning protocols. Dictionary-based methods show better performance (F1-score up to .67), but remain noticeably behind more sophisticated approaches, reflecting the limited ability of fixed keyword lists to capture the richness of how new features are described in release notes.

Among ‘classical’ ML approaches (e.g., logistic regression, random forest, and support vector machines), F1-scores range from approximately .70 to .85, with logistic regression and support vector machine performing best within this category. The convolutional neural network performs similarly with an F1-score of .828, and early transformer-based pre-trained language models (PLM) perform even better with F1-scores close to .90. Crucially, all three categories of models can be run locally and reproduced under identical settings and training data, ensuring full replicability. However, each requires a substantial volume of labeled data to achieve their upper performance bounds (in this study, $n = 2,000$).

Even more compelling is the performance of several LLMs integrated into our framework. Without any additional training data (i.e., in their default configuration), and based on our framework, several LLMs achieve F1-scores on par with or better than PLMs, with GPT-4.1 reaching the highest F1-score of .901. Still, other default configurations such as GPT-4.1 Nano or Claude 3.5 Haiku fall below the performance of ‘classical’ ML models, suggesting that not all LLMs excel ‘out of the box’ and that models with fewer parameters perform worse than their larger

counterparts. However, once fine-tuned with corresponding input-output pairs, performance improves across the board, with OpenAI’s GPT models edging F1-scores up to .926.

Notably, even in their default configuration, LLMs maintain high precision, meaning they rarely mislabel non-innovative updates as innovative. In contrast, recall (i.e., the ability to correctly identify the full range of innovative updates) improves more markedly with fine-tuning. This suggests that, much like ‘classical’ ML models or earlier PLMs, additional training data helps LLMs to learn the varied and sometimes subtle language used to describe new features, thus better capturing the ‘breadth’ of innovative updates.

In practice, many LLMs are accessible only via an external API, making local deployment impractical or impossible. To nonetheless ensure reliability in these scenarios, we standardize our setup by setting the temperature parameter to 0 and by defining a model seed⁸. This adjustment produces deterministic outputs from the same model versions under identical prompts, with CRs above .98. The implications of variations in model selection, temperature settings, and training data for performance and reliability are further discussed in Online Appendix E. To still enable scalable and reproducible deployment beyond external APIs, we additionally implement a teacher-student distillation setup described in Online Appendix F. In this setup, a high-performing API-based “teacher” LLM generates labels that are used to train smaller (open or local) “student” models, enabling low-cost, locally deployable inference with performance close to the teacher.

----- Insert Table 3 about here -----

Table 3 presents a more fine-grained, multi-class version of the update classification. Columns 1-7 reflect dimensions from the data structure and prompt, with per-class F1-scores,

⁸ Defining a seed parameter fixes the pseudo-random sampling state, so identical prompts and parameters return the same output when the model (*fingerprint*) is unchanged. Omitting the seed—even with temperature=0 calls—lets stochastic tie-breaks and sampling noise vary across runs and can reduce consistency across runs.

macro- and weighted-average results, and the CR across three independent runs. The Macro F1-score averages performance equally across all classes, ensuring both common and rare updates are considered, whereas the weighted F1-score uses sample-based weights to reflect the dataset’s true distribution. Comparing macro- and weighted-averages helps clarify whether models struggle on underrepresented classes (via macro-average) or show strong overall performance on the more frequent classes (via weighted-average). Large discrepancies between these averages can signal that certain classes are being overlooked.

Earlier dictionary- or version-based methods are not included in this analysis because they rely on fixed heuristics or keywords that cannot differentiate between subtle update types. ‘Classical’ ML and early PLMs (e.g., ELECTRA) perform robustly overall but fail to detect rare classes when training data are scarce. Although misclassifying classes that serve a similar measurement objective may not drastically affect the overall innovation score, failing to detect a class altogether can skew outcomes and introduce bias.

By contrast, default LLMs demonstrate more balanced performance across all classes. When fine-tuned with the same training data used by ML or PLM approaches, LLMs (e.g., GPT-4.1 or GPT-4.1 Mini) can achieve a macro-averaged F1-score of up to .720, showcasing the advantages of our proposed framework and the capacity of novel LLMs to precisely classify content. Consistency rates (CR) for both default and fine-tuned LLMs remain very similar and consistently far exceed typical human inter-annotator agreement standards (Artstein and Poesio, 2008; Carletta, 1996; Landis and Koch, 1977). Although the models are accessed via an API and not in a local setup, with the specified temperature and seed settings, both default and fine-tuned versions produce identical outputs across independent runs, with only rare discrepancies.

4.2. Demonstration Study 2: User Ideas in Product Reviews

In our second demonstration study, we apply and evaluate our framework to classify user-generated product reviews. External sources of innovation such as user ideas are invaluable for organizational innovation activities, providing access to knowledge and insights that lie beyond organizational boundaries (Chesbrough, 2003; Von Hippel, 1978). However, research on crowdsourcing and idea evaluation highlights that identifying ideas that are both original and practically relevant, and that genuinely add value to organizations' innovation activities, remains challenging due to the overwhelming number of suggestions, their often unstructured nature, and evaluators' limited attention in idea contests and review platforms (e.g., Piezunka and Dahlander, 2015).

Previous methods that analyze customer ideas and needs in product reviews often rely on expert committees or advanced NLP techniques backed by large training datasets (e.g., Timoshenko and Hauser, 2019; Zhang et al., 2021). While effective, these approaches can easily become resource-intensive, requiring extensive training data, and significant involvement from many domain experts. We illustrate LLMs' potential to generate valid classifications of user ideas with less reliance on extensive data or expert involvement.

4.2.1. Data

Our dataset, sampling, and preprocessing procedures are comparable to those used in Demonstration Study 1. From a large dataset covering the complete review history of all apps that have been listed at least once in the top-100 overall or category-specific rankings, we randomly sampled 4,000 reviews written in English.

To establish ground truth for the sample pool, a co-author and a research assistant independently coded the reviews using a coding scheme developed through inductive coding by our team. This scheme comprises nine content dimensions. A subset of $n=1,000$ review

descriptions was independently coded by both raters to assess inter-rater reliability (observed agreement = .89, macro-average κ = .77), and any discrepancies were discussed.

In line with Demonstration Study 1, we generated two distinct random subsamples from the labeled pool of $n=4,000$ reviews: a validation set of $n=1,000$ reviews serving as a holdout, and a training set of $n=2,000$ reviews. Both subsamples were constructed to reflect a representative class distribution of the overall pool. The remaining $n=1,000$ reviews of the labeled pool serve to back additional analyses to explore variations in class distributions.

4.2.2. Research design

Our measurement objective builds on the premise that product reviews often contain valuable suggestions, ideas, and expressions of customer needs (Timoshenko and Hauser, 2019; Zhang et al., 2021). These reviews are multifaceted, ranging from brief comments to extensive narratives where users may suggest or report several issues at once. As such, our prompt is aimed at distinguishing between different types of content and identifying instances where multiple topics are addressed simultaneously.

----- Insert Figure 2 about here -----

To further inform prompt design, we conducted inductive coding of reviews until meaning saturation was reached. This process resulted in nine content dimensions that characterize the reviews' data structure and provide a means to differentiate them at a highly granular level (cf. Online Appendix B2). User-generated ideas are most directly captured by dimension 3 (suggestions for new features or content or revival of removed features), while the remaining dimensions distinguish other feedback types, such as bug reports or general opinion.⁹ Both our measurement

⁹ Innovation-related input is not confined to explicit suggestions. For instance, reports of bugs or monetization issues can likewise reveal unmet customer needs (Timoshenko and Hauser, 2019). The granular data structure allows researchers to include or exclude such content depending on the construct of interest in subsequent analyses.

objective and the data structure are then integrated in the final design of our prompt (see Figure 2) and the content dimensions serve as the foundation for labeling reviews in our sample pool.

Our framework is applied to the same LLMs used in Demonstration Study 1. To prepare training data for fine-tuning, we created input-output pairs by combining every merged prompt and review text with its corresponding manual label. The LLM-generated outputs are then evaluated against our validation dataset and benchmarked against the performance of ‘classical’ machine learning models, deep-learning approaches, and transformer-based pre-trained language models on the same validation dataset. For benchmarking against the other models, we used the same labeled training data as our LLM fine-tuning, though without the prompt. Given the multi-label nature of our task, we employed a one-vs-rest strategy, where each label is encoded and classified as an independent binary feature to capture the presence of multiple labels per review. For instance, if a review’s labels were 8 and 4, this method converted it into a binary vector [0 0 0 0 1 0 0 0 1] representing the presence of labels from “0” through “8”. Similar to Demonstration Study 1, we experimented with different embedding techniques and hyperparameter settings to identify the best-performing setup. The rationale behind our configuration for each model is detailed in Online Appendix D.

Although product reviews, and particularly app reviews, are a common subject in NLP tasks and have been extensively explored for feature extraction and content classification (e.g., Gunathilaka and de Silva, 2025; Khlifi et al., 2024), no study has, to our knowledge, tested such fine-grained content differentiation or demonstrated LLMs’ capabilities in a multi-label setting.

4.2.3. Results

Table 4 shows the results of applying our framework to an analysis of reviews, reporting comparable performance and reliability metrics as in Demonstration Study 1. The nine columns (0-8) correspond to the content dimensions derived from the data structure and included prompt.

Because each review can receive multiple labels in the one-vs-rest approach, the total number of labeled instances (n) exceeds 1,000, even though the holdout sample contains 1,000 reviews.

----- Insert Table 4 about here -----

The overall observations from Demonstration Study 1 are corroborated here. ‘Classical’ machine learning models vary considerably, with random forest showing the highest Macro F1 score (.401), and K-Nearest-Neighbor (Macro F1 score = .208) yielding the lowest performance. Applying pre-trained language models (PLMs) results in modest improvements, with XLNet achieving a Macro F1 score of .436. Notably, applying our framework in LLMs outperforms these approaches, with even default configurations achieving superior results. The best performance is obtained through fine-tuning, with OpenAI’s GPT models achieving top Macro F1 scores (e.g., GPT-4.1 = .706).

Models that leverage additional training data or benefit from fine-tuning particularly excel in classes with larger sample sizes. Interestingly, default versions of LLMs can achieve F1 scores comparable to their fine-tuned counterparts and deliver moderate performance in scenarios where traditional methods struggle, such as with underrepresented samples or linguistically complex cases (e.g., class 6). Further performance enhancements are achievable by fine-tuning labeled prompt-review-output combinations. Consistent with Demonstration Study 1, the temperature parameter is set at 0, and the model seed is fixed to systematically assess reliability metrics. Also in this multi-label setting, outputs across three independent runs remain highly consistent, with consistency rates persistently above .91. We discuss variations in model selection, temperature settings, and training data in Online Appendix E.

4.3. The Relevance of Model Fine-Tuning

Both demonstration studies compare default and fine-tuned model configurations. The performance results reveal different gains from fine-tuning across settings, indicating that the value

that fine-tuning adds should be assessed with regard to a study's classification task and measurement objectives. The decision whether to include model fine-tuning is consequential: fine-tuning requires expert-validated training data that is time-intensive to create and entails further choices regarding the size and distribution of this data.

For classifications with common and balanced classes, default configurations approach the performance ceiling and the gains from fine-tuning are unlikely to alter substantive conclusions (Table 2; e.g., GPT-4.1 default F1 = .901 vs. .926 fine-tuned). Small increases in the F1-score should be weighed carefully against the costs of fine-tuning. In some cases, fine-tuning can even reduce performance. With limited training data under a representative class distribution, the performance of fine-tuned models may fall below the performance of their own default configurations. The training volume beyond which fine-tuning pays off varies by model and task and is only revealed through holdout validation (Online Appendix Figure E4; in our setting, a fine-tuned GPT-4.1 stays below its default performance until $n = 500$, Mistral models until $n = 2,000$).

Fine-tuning is most valuable when default configurations miss or under-recall construct-relevant instances. This applies when the construct resides in rare classes (Table 4; e.g., GPT-4.1, class 4 F1 from .727 to .807). Fine-tuning may also add value, even when performance gains are small, when a research design requires maximizing the identification of relevant instances, as default LLMs already classify with high precision while fine-tuning primarily raises recall (Table 2; e.g., GPT-4.1 recall from .879 to .961). Fine-tuning thereby changes the error structure. When a class is missed entirely without fine-tuning, all of its instances are measured incorrectly, which constitutes systematic measurement error and can bias downstream estimates. After fine-tuning, these classes are detected, and the remaining misclassifications are fewer and no longer concentrated in single classes. Because these gains are not uniform across models and classes

(Table 4; e.g., GPT-4.1 Mini, class 6 F1 from .629 to .435), per-class validation on the holdout sample should accompany any fine-tuning decision.

Finally, fine-tuning can serve deployment. In the teacher-student configuration (Online Appendix F), a high-performing, often proprietary and API-based, LLM labels training data to fine-tune a smaller, locally deployable student model. Fine-tuning on these labels brings the student model close to the teacher’s performance, while inference runs locally at a fraction of the teacher’s (recurring) costs and independent of provider-side changes.

Online Appendix G provides a structured summary of these contingencies. The overarching criterion is the structure of the remaining errors. Where default configurations already err in largely unsystematic ways, fine-tuning is less effective; where errors concentrate systematically in construct-relevant classes, fine-tuning may reduce bias in downstream estimates.

4.4. Measurement Performance and Implications for Downstream Estimates

The performance results presented above assess classification quality against human expert judgment. In empirical research, however, classifications typically serve as variables in subsequent analyses, and their measurement error may affect coefficient estimates. Recent work cautions that estimates based on LLM-generated variables can shift markedly with seemingly innocuous design choices (Camuffo et al., 2026; Ludwig et al., 2025).

We illustrate these consequences with a simulation on the Demonstration Study 1 holdout sample. Consider a researcher who uses the framework’s classifications (i) as an explanatory variable, for instance to estimate whether innovative updates increase subsequent user engagement, or (ii) as an outcome variable, for instance, to examine whether a specific policy changed the share of innovative updates. For case (i), we assume that innovative updates increase engagement and simulate an engagement outcome accordingly with the coefficient set at $\beta = .50$ and noise calibrated such that innovation explains a modest share of the variance ($R^2 = .15$). We estimate the regression

with each model configuration's holdout classifications in place of the expert labels. The gap to the set $\beta = .50$ then shows how strongly each configuration's errors distort the estimate. Online Appendix H1 details the design.

In case (i), Figure 3, Panel (a) reports the estimates for all configurations evaluated in Demonstration Study 1 for which the binary innovation variable exhibits sufficient variation to estimate a coefficient. Against the set $\beta = .50$, estimates range from approximately $-.02$ to $.45$. Fine-tuned LLM configurations produce estimates closest to the known coefficient (median $\beta = .42$), followed by default LLM (.37) and PLM configurations (.35), whereas classical ML configurations show stronger deviation (.31). More generally, classification performance is highly informative about the extent of downstream bias in this setting (see Online Appendix H2 for additional analyses). Across configurations, F1-scores are strongly negatively associated with absolute coefficient bias (Spearman $\rho = -.93$), so that better classification performance translates into less deviation from the true coefficient value.

Case (ii) yields a different pattern. We use the classifications as the outcome and compare the change in the measured innovation share between an earlier and later period with the expert-labeled change of $-.06$ percentage points (Figure 3, Panel b). Individual configurations generate both positive and negative deviations from this benchmark, in some cases exceeding several percentage points. Importantly, aggregate classification performance provides little guidance about the magnitude of this distortion. Among the same configurations used in case (i), the association between F1 and absolute pre-post distortion is weak (Spearman $\rho = .14$; see Online Appendix H3 for additional analyses).

----- Insert Figure 3 about here -----

Which error type is most consequential hence depends on the research design. As in our simulated case (i), when a binary measure serves as a regressor, false positives and false negatives

both attenuate the estimate toward zero. Aggregate performance measures such as the F1-score can therefore provide a useful indication of a configuration's suitability for subsequent analyses, although the exact bias depends on the underlying error structure. As in our case (ii), when the measure serves as the outcome in, for instance, a pre-post design, stable error rates do not fabricate a change when the true outcome remains unchanged, although they may attenuate an existing change. Yet, if either error rate differs across the compared periods, the measured share can change independently of the underlying construct and we may observe an effect that does not exist with an expert-labeled outcome. A configuration therefore has to be validated not only in aggregate but also within the cells which a research design compares. Beyond both cases, when the construct resides in rare classes, false negatives concentrated in a single class (i.e., systematic misses) can make corresponding effects difficult or impossible to estimate.

When applying the framework outside a simulation, the true coefficient is unknown *ex ante*. The holdout sample can nevertheless reveal a configuration's error structure (Ludwig et al., 2025). Our simulation illustrates what this structure can imply for downstream estimates. In particular, the observed pattern of false-positive and false-negative errors provides an indication of the expected magnitude of bias and, in comparative designs, the potential risk of fabricated effects when error rates vary across design cells. A valuable complement is thus to re-estimate the focal model under alternative validated configurations, which reveals how strongly substantive conclusions depend on design choices and the range of estimates that different plausible specifications produce (Carlson and Burbano, 2026). Researchers should accordingly document the selected configuration and its aggregate validation performance, as well as the error structure observed on the holdout sample for the empirical design of interest.

4.5. Framework Relevance Under LLM Evolution

Given the rapid pace of LLM development and the prevalence of silent model updates in commercial APIs, a natural concern is whether LLM-based measurement approaches become obsolete as model versions and capabilities evolve. For instance, over the course of developing the framework and conducting the two demonstration studies, several LLM providers released new model versions (e.g., OpenAI’s GPT-5 series). Such model updates can affect instruction-following, reasoning behavior, and sensitivity to linguistic cues, which may affect performance metrics or shift decision thresholds (e.g., Chen et al., 2024).

We argue that our framework is designed to be agnostic across model versions because it anchors the measurement task in expert-validated content dimensions and a structured evaluation procedure rather than model-specific heuristics. By mapping domain experts’ cues into explicit decision criteria (and, when feasible, incorporating expert-labeled examples via fine-tuning), the framework specifies what should be measured and why, while allowing the underlying model to improve in how it recognizes and integrates these cues. In this sense, newer models are expected to maintain or even enhance the framework’s effectiveness rather than invalidate it. To assess temporal robustness, we re-ran the baseline specification of both demonstration studies on the same validation data. Overall, we found strong robustness in performance metrics and no differences in the resulting patterns or interpretations. Full details are provided in Online Appendix I.

5. Conditions for Applying the Framework

Abstracting from the features that the two demonstration studies share, we identify three conditions under which the framework is applicable. First, the construct must manifest in the text itself. The text has to contain the cues a domain expert would rely on, such that an expert reading only the focal text could reach a judgment. Second, a small number of domain experts must be available to

articulate and validate these cues and to label a holdout sample. Third, these expert judgments of the construct must be sufficiently reliable to serve as ground truth (Einhorn, 1974).

These conditions also clarify which empirical settings the framework can be applied to. The availability of text data is necessary but has rarely been the binding constraint as text documenting innovative activity has become widely available (Bellstam et al., 2021). The main constraint was the need for expert-assessed labels at scale. Our framework relaxes this constraint considerably. The contexts that become more tractable are therefore those with construct-bearing text but no institutionalized expert evaluation of the kind that, for instance, patents provide. Examples include idea submissions on crowdsourcing platforms (Zaggl et al., 2023), crowdfunding pitches (Mollick and Nanda, 2016), and startup value propositions (Guzman and Li, 2023).

Within these conditions, the framework is not equally valuable for every classification task. Its comparative advantage lies in tasks where the construct is latent and judgment-intensive, the language describing it is heterogeneous and lacks standardized terminology, and expert-labeled training data is scarce or costly to produce. Where the construct resides in surface features of text, established alternatives remain competitive at lower cost. For instance, dictionary methods perform well when terminology is unambiguous and broadly shared (Miric et al., 2023a), particularly where fields maintain curated ontologies that encode expert judgment in the vocabulary itself (Petersen, 2022), and classical supervised ML performs well when large, labeled datasets exist and linguistic variation is more limited (Miric et al., 2023a). In such settings the framework works but is less competitive on cost.

Constructs that experts evaluate from non-linguistic evidence lie outside our demonstrations altogether. While current LLMs accept multimodal input, extending the framework’s expert-cue validation to such data remains a task for future research. Finally, strategic misrepresentation constrains any text-based method, not LLM-based measurement specifically.

Mild promotional language can be absorbed by the coding scheme, as the marketing and branding dimension in Demonstration Study 1 illustrates, but systematic misrepresentation that leaves no trace within the text cannot be corrected by a classifier. This concern intensifies once a measure itself becomes a target of strategic behavior, and it compounds as linguistic conventions drift over time, so that cue sets and fine-tuned models require periodic revalidation.

Relatedly, the framework’s validity is local to the corpus on which it is established. The training corpora of LLMs overrepresent recent, English-language, and Western material, and model outputs may by default align most closely with the judgments of the corresponding populations (Bender et al., 2021; Tao et al., 2024).¹⁰ Expert-validated cues and fine-tuning anchor the framework’s evaluations to the designated evaluators, yet applying these cues still draws on the model’s pre-trained language understanding. Performance can therefore degrade with temporal, linguistic, or cultural distance from this distribution, and validation on a contemporary corpus does not certify applications to distant ones.¹¹ Such applications remain possible in principle, but they require establishing an expert-validated holdout for the target period and domain, which in turn presupposes experts for that period and domain.

The framework relies on a ground truth. Its performance criterion is the approximation of expert judgment. For genuinely contested constructs on which domain experts do not converge there is nothing to approximate and no benchmark against which validation is meaningful (Einhorn,

¹⁰ This concern relates to the broader notion of WEIRD bias, which describes how evidence and data disproportionately drawn from Western, Educated, Industrialized, Rich, and Democratic populations may not generalize to populations outside these contexts (Henrich et al., 2010). In the context of LLMs, the overrepresentation of Western and English-language material in training data raises an analogous concern about whose language use, knowledge, and judgments are most strongly represented.

¹¹ Historical applications illustrate two failure modes. A present-day LLM assessing the innovativeness of the incandescent lightbulb knows its eventual significance, so that ex ante novelty is conflated with ex post importance. At the same time, nineteenth-century language and genre conventions are likely sparse in training data, which degrades comprehension of ordinary period documents (Lazaridou et al., 2021).

1974). The appropriate response in such cases is conceptual work on the construct or the aggregation of judgments across evaluator groups.

Table J13 in Online Appendix J summarizes these conditions as questions that researchers can apply to their own settings before adopting the framework.

6. Discussion

6.1. Contributions to Innovation Management Research

The measurement of innovation-related phenomena is often constrained by the challenges inherent in identifying and assessing contextual information (Boudreau et al., 2016). In many cases, such evaluation relies on expertise-based judgment, yet access to qualified experts is often limited. Moreover, analyzing large datasets is impeded by the inherent cognitive limits of human evaluators. As the volume of the data increases, the cognitive load on experts can lead to greater reliance on heuristics and biases, which may impair their ability to evaluate effectively (Tversky and Kahneman, 1974). Empirical research in innovation has therefore mainly been conducted in contexts where reliable measures are readily available. Our research addresses these challenges by developing an LLM-based framework. Our framework is based on the premise that certain contextual cues are highly indicative of expert judgment (Einhorn, 1974). We apply qualitative methods and collaborate with experts to identify these cues, assess their relevance, and integrate these insights systematically into prompt design and, where useful, fine-tuning for the LLM. Because expert judgments may draw on learned, tacit conceptions of similarity that are difficult to articulate explicitly (Nightingale, 1998), this process can also help make otherwise implicit construct boundaries and evaluation criteria more explicit. A key strength of our framework is that it reduces the need for repeated human expert evaluation at scale while still achieving comparable evaluations. The scalability enabled by this approach simplifies empirical innovation research

across a broader range of contexts, and also provides opportunities for additional robustness checks and analytical perspectives.

We successfully illustrate the applicability and utility of our framework in two demonstration studies conducted in different innovation contexts. Although the validation exercises are specific to innovation in digital platform ecosystems, they capture two central concepts in innovation research: the outputs of innovative activity and the generation of ideas, holding promise for extension to other innovation contexts as well. Our results further show that fine-tuning LLMs is not a prerequisite for such applications of our framework, but a design choice whose value depends on the structure of the remaining classification errors and the requirements of the intended empirical analysis. At the same time, even the best-performing models in our demonstration studies still misclassify roughly one in ten (positive) instances (precision $\approx .90$). Any high-stakes deployment should retain a human-in-the-loop review and undertake a careful error-impact analysis tailored to the application context.

Furthermore, the application of our framework makes specific contributions to the research areas of these two demonstration studies by enhancing existing metrics. First, in studying innovation in software application development in digital platform ecosystems (e.g., Agarwal and Kapoor, 2023; Kircher and Foerderer, 2024; Wen and Zhu, 2019), we improve measurement performance, assessed by F1-scores, of the innovation variable by 39% relative to the best-performing measure from previous studies (see Table 2), and additionally enable a more fine-grained differentiation of update content. Second, in identifying innovation-related user-generated ideas in product reviews (e.g., Timoshenko and Hauser, 2019; Zhang et al., 2021), our framework allows for a more detailed, multi-class screening of ideas.

The implications of these improvements are twofold. First, they introduce new levels of analysis and promising directions for future research. Specifically, the capability to examine update

release notes and product reviews at such detailed levels allows researchers to accurately trace how user innovations are adopted and integrated over time. Consequently, researchers can gain deeper insights into when and how user-suggested ideas become part of products and assess the extent to which these incorporations contribute to outcomes like sustained competitive advantage. For instance, products with larger and more engaged user communities may disproportionately benefit from a greater influx of user-generated ideas, potentially accelerating innovation cycles or enhancing adaptability to market demands. Moreover, such detailed analyses facilitate better evaluation of the impact of external events that influence development and user behavior. For example, regulators and platform owners of digital platforms are increasingly emphasizing user privacy by introducing privacy-preserving regulations. The refined metrics demonstrated in our studies allow researchers to quantify the effects of these regulations, such as their influence on application development practices and user perceptions regarding the regulations and corresponding implementations by developers. Second, the improvements may increase the validity of empirical findings and conclusions such as in studies on complementor innovation on app platforms, by reducing potential systematic measurement errors that could result in substantial bias in coefficient estimates (e.g., deHaan et al., 2025; Millimet and Parmeter, 2022). Our additional analyses further show that the consequences of such errors depend on the empirical design and the underlying error structure, highlighting the importance of validating measures with the intended downstream application in mind. Future research, through replication studies and further exploration, is needed to fully assess the implications of these enhancements.

6.2. The Role of the Expert Evaluator

Our framework is centered on relevant domain expertise being available for cue identification and weighting. The underlying assumption is that bounded rationality and imperfect judgment in evaluation tasks affect all possible evaluators, while those whose expertise aligns most closely with

the evaluation context exhibit lower uncertainty in their judgments (Böttcher and Klingebiel, 2025; Boudreau et al., 2016; Csaszar and Eggers, 2013). This is because domain experts are able to disproportionately identify and interpret more relevant information compared to less specialized evaluators (Ericsson and Smith, 1991; Johnson et al., 1982; Kahneman et al., 1982).

However, we also recognize that identifying the most suitable expert *ex ante* is challenging. There exists a risk of cognitive entrenchment, where experts may be less inclined to identify and integrate new information that challenges their existing views (Almandoz and Tilcsik, 2016; Dane, 2010; Furr et al., 2012). As a result, they may inadvertently extrapolate from their established knowledge into domains where their expertise is less applicable. This misalignment can degrade evaluation performance, even to a point that their judgment becomes inferior to that of generalist evaluators (Böttcher and Klingebiel, 2025; Csaszar and Eggers, 2013).

Although the framework is designed to enforce robust evaluation processes, it is not immune to the risk of replicating or even reinforcing such evaluator bias. These risks are partially mitigated by the framework’s data-centric approach, where the extraction of the data structure is carried out by the researcher, and domain experts are engaged primarily for validation and adjustment purposes. Still, there remains a potential for bias, particularly when fine-tuning with biased data is not carefully managed or when experts and the model are subject to the same bias. Where expert coders hold, for instance, a WEIRD-normed conception of what constitutes innovation (Henrich et al., 2010), both deviate in the same direction, and the error would survive holdout validation undetected.

Given these limitations, alternative evaluation scenarios are worth considering. In contexts where identifying the most relevant domain experts is challenging, integrating insights, for instance, from crowds (e.g., Mollick and Nanda, 2016; Wimbauer et al., 2019) or evaluator panels (e.g., Criscuolo et al., 2017) may yield an increase in evaluation performance. The primary

advantage of the proposed framework lies in its data-centric approach, where expert validation is confined to the data structure. This design makes the framework agnostic to the type and format of expert input. Future research could explore how integrating different evaluator groups and formats (e.g., ranking, voting, aggregation) in our framework impacts performance, providing deeper insights into the trade-offs and benefits of each approach. This exploration is particularly relevant as evaluation methods for LLMs continue to evolve, with hybrid strategies combining human and automated evaluations emerging as a promising direction.

6.3. Ethical and Practical Considerations

LLMs are often considered “black boxes” due to their complexity and opacity. They are pre-trained on exceptionally large datasets and build on advanced deep learning architectures with billions of parameters. Despite their impressive performance, our current understanding of their internal mechanisms is limited (Bommasani et al., 2021), and independently constructing comparable models at similar performance scales is typically beyond the capabilities of individual researchers or smaller institutions. Moreover, the processes by which LLMs generate outputs and the biases introduced during pre-training, particularly with datasets undisclosed, are not yet fully understood. Variations between different models, or even between different versions of the same model, are both inevitable and expected. When applying the framework, it is therefore essential to systematically validate the outputs against a holdout sample or established ground truth to accurately assess performance and detect potential systematic measurement errors.

Practically, using LLMs involves navigating operational considerations related to both proprietary and open-weight models. Proprietary models, typically accessed via APIs, often provide minimal and non-specific information about their internal configurations and are subject to potential modifications or discontinuation by providers. Therefore, detailed documentation of the exact model version used and transparent reporting of all design decisions is essential for

ensuring reproducibility. While proprietary models still offer advantages in efficiency and effectiveness, API costs, despite decreasing over time, may accumulate significantly when working with large datasets. Open-weight models, on the other hand, reduce the risks associated with proprietary models, such as unexpected accessibility changes or alterations in performance. However, using medium or large-sized open-weight models locally demands substantial hardware infrastructure. Alternatively, cloud-based deployment can also incur significant scaling costs. Despite platforms like Hugging Face considerably reducing entry barriers, substantial technical expertise remains essential. A practical middle ground is to use proprietary, high-performing models as teachers to generate labels for training smaller, open or locally deployable student models, thereby reducing recurring API costs and dependence on provider-side changes while retaining much of the predictive quality.

Given the rapid advancements in LLM technology, questions may arise regarding the long-term relevance of our proposed framework. The proposed framework is designed to be model-agnostic, allowing it to leverage ongoing advancements in LLM capabilities and adapt to future improvements in the field. By focusing on mapping expert cues into the evaluation process, the framework provides a robust and adaptable foundation for scaling the assessment of innovation-related constructs in diverse scenarios.

References

- Abernathy, W.J., Clark, K.B., 1985. Innovation: Mapping the winds of creative destruction. *Research Policy* 14, 3–22. [https://doi.org/10.1016/0048-7333\(85\)90021-6](https://doi.org/10.1016/0048-7333(85)90021-6)
- Abrahamson, E., 1991. Managerial Fads and Fashions: The Diffusion and Rejection of Innovations. *The Academy of Management Review* 16, 586–612. <https://doi.org/10.2307/258919>
- Agarwal, S., Kapoor, R., 2023. Value Creation Tradeoff in Business Ecosystems: Leveraging Complementarities While Managing Interdependencies. *Organization Science* 34, 1216–1242. <https://doi.org/10.1287/orsc.2022.1615>
- Almandoz, J., Tilcsik, A., 2016. When Experts Become Liabilities: Domain Experts on Boards and Organizational Failure. *AMJ* 59, 1124–1149. <https://doi.org/10.5465/amj.2013.1211>

- Artstein, R., Poesio, M., 2008. Inter-Coder Agreement for Computational Linguistics. *Computational Linguistics* 34, 555–596. <https://doi.org/10.1162/coli.07-034-R2>
- Bellstam, G., Bhagat, S., Cookson, J.A., 2021. A Text-Based Analysis of Corporate Innovation. *Management Science* 67, 4004–4031. <https://doi.org/10.1287/mnsc.2020.3682>
- Bender, E.M., Gebru, T., McMillan-Major, A., Shmitchell, S., 2021. On the Dangers of Stochastic Parrots: Can Language Models Be Too Big?, in: *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, ACM, 610–623. <https://doi.org/10.1145/3442188.3445922>
- Bommasani, R., Hudson, D.A., Adeli, E., Altman, R., Arora, S., [...], K., Liang, P., 2021. On the Opportunities and Risks of Foundation Models. <https://doi.org/10.48550/ARXIV.2108.07258>
- Böttcher, L., Klingebiel, R., 2025. Organizational Selection of Innovation. *Organization Science* 36, 387–410. <https://doi.org/10.1287/orsc.2023.17357>
- Boudreau, K.J., 2012. Let a Thousand Flowers Bloom? An Early Look at Large Numbers of Software App Developers and Patterns of Innovation. *Organization Science* 23, 1409–1427. <https://doi.org/10.1287/orsc.1110.0678>
- Boudreau, K.J., Guinan, E.C., Lakhani, K.R., Riedl, C., 2016. Looking Across and Looking Beyond the Knowledge Frontier: Intellectual Distance, Novelty, and Resource Allocation in Science. *Management Science* 62, 2765–2783. <https://doi.org/10.1287/mnsc.2015.2285>
- Boudreau, K.J., Lacetera, N., Lakhani, K.R., 2011. Incentives and Problem Uncertainty in Innovation Contests: An Empirical Analysis. *Management Science* 57, 843–863. <https://doi.org/10.1287/mnsc.1110.1322>
- Boussiou, L., Lane, J.N., Zhang, M., Jacimovic, V., Lakhani, K.R., 2024. The Crowdless Future? Generative AI and Creative Problem-Solving. *Organization Science* 35, 1589–1607. <https://doi.org/10.1287/orsc.2023.18430>
- Brown, T.B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., [...], Amodei, D., 2020. Language Models are Few-Shot Learners. <https://doi.org/10.48550/ARXIV.2005.14165>
- Brynjolfsson, E., Enríquez, J.R., Kazinnik, S., Nguyen, D., 2026. Augmenting Survey Data with Generative AI: An Application to Economic Research. <https://doi.org/10.2139/ssrn.6343598>
- Camuffo, A., Gambardella, A., Kazemi, S., Malachowski, J., Pandey, A., 2026. Variance-Aware LLM Annotation for Strategy Research: Sources, Diagnostics, and a Protocol for Reliable Measurement. <https://doi.org/10.48550/arXiv.2601.02370>
- Carletta, J., 1996. Assessing Agreement on Classification Tasks: The Kappa Statistic. *Computational Linguistics* 22, 249–254.
- Carlson, N.A., Burbano, V., 2026. The use of LLMs to annotate data in management research: Foundational guidelines and warnings. *Strategic Management Journal* 47, 699–725. <https://doi.org/10.1002/smj.70023>
- Chase, W.G., Simon, H.A., 1973. Perception in chess. *Cognitive Psychology* 4, 55–81. [https://doi.org/10.1016/0010-0285\(73\)90004-2](https://doi.org/10.1016/0010-0285(73)90004-2)
- Chen, L., Zaharia, M., Zou, J., 2024. How Is ChatGPT’s Behavior Changing Over Time? *Harvard Data Science Review* 6. <https://doi.org/10.1162/99608f92.5317da47>
- Cheng, X., Mayya, R., Sedoc, J., 2024. From Human Annotation to LLMs: SILICON Annotation Workflow for Management Research. <https://doi.org/10.48550/ARXIV.2412.14461>
- Chesbrough, H.W., 2003. *Open innovation: the new imperative for creating and profiting from technology*. Harvard Business School Press, Boston, MA.

- Choudhury, P., Wang, D., Carlson, N.A., Khanna, T., 2019. Machine learning approaches to facial and text analysis: Discovering CEO oral communication styles. *Strategic Management Journal* 40, 1705–1732. <https://doi.org/10.1002/smj.3067>
- Comino, S., Manenti, F.M., Mariuzzo, F., 2019. Updates management in mobile applications: iTunes versus Google Play. *Journal of Economics & Management Strategy* 28, 392–419. <https://doi.org/10.1111/jems.12288>
- Criscuolo, P., Dahlander, L., Grohsjean, T., Salter, A., 2017. Evaluating Novelty: The Role of Panels in the Selection of R&D Projects. *Academy of Management Journal* 60, 433–460. <https://doi.org/10.5465/amj.2014.0861>
- Crossan, M.M., Apaydin, M., 2010. A Multi-Dimensional Framework of Organizational Innovation: A Systematic Review of the Literature. *Journal of Management Studies* 47, 1154–1191. <https://doi.org/10.1111/j.1467-6486.2009.00880.x>
- Csaszar, F.A., Eggers, J.P., 2013. Organizational Decision Making: An Information Aggregation View. *Management Science* 59, 2257–2277. <https://doi.org/10.1287/mnsc.1120.1698>
- Dane, E., 2010. Reconsidering the Trade-off Between Expertise and Flexibility: a Cognitive Entrenchment Perspective. *Academy of Management Review* 35, 579–603. <https://doi.org/10.5465/amr.35.4.zok579>
- Dane, E., Pratt, M.G., 2007. Exploring Intuition and its Role in Managerial Decision Making. *Academy of Management Review* 32, 33–54. <https://doi.org/10.5465/amr.2007.23463682>
- de Kok, T., 2025. ChatGPT for Textual Analysis? How to Use Generative LLMs in Accounting Research. *Management Science* 71, 7888–7906. <https://doi.org/10.1287/mnsc.2023.03253>
- deHaan, E., Lawrence, A., Litjens, R., 2025. Measuring Investor Attention Using Google Search. *Management Science* 71, 6275–6297. <https://doi.org/10.1287/mnsc.2022.02174>
- DeVellis, R.F., Thorpe, C.T., 2022. Scale development: theory and applications, 5th ed. SAGE, Thousand Oaks, CA.
- Doshi, A.R., Bell, J.J., Mirzayev, E., Vanneste, B.S., 2025. Generative artificial intelligence and evaluating strategic decisions. *Strategic Management Journal* 46, 583–610. <https://doi.org/10.1002/smj.3677>
- Einhorn, H.J., 1974. Expert judgment: Some necessary conditions and an example. *Journal of Applied Psychology* 59, 562–571. <https://doi.org/10.1037/h0037164>
- Ericsson, K.A., Smith, J. (Eds.), 1991. *Toward a general theory of expertise: prospects and limits*. Cambridge University Press, Cambridge.
- Fleming, L., 2001. Recombinant Uncertainty in Technological Search. *Management Science* 47, 117–132. <https://doi.org/10.1287/mnsc.47.1.117.10671>
- Foerderer, J., Kude, T., Mithas, S., Heinzl, A., 2018. Does Platform Owner’s Entry Crowd Out Innovation? Evidence from Google Photos. *Information Systems Research* 29, 444–460. <https://doi.org/10.1287/isre.2018.0787>
- Furr, N.R., Cavarretta, F., Garg, S., 2012. Who Changes Course? The Role of Domain Knowledge and Novel Framing in Making Technology Changes. *Strategic Entrepreneurship Journal* 6, 236–256. <https://doi.org/10.1002/sej.1137>
- Fyffe, S., Lee, P., Kaplan, S., 2024. “Transforming” Personality Scale Development: Illustrating the Potential of State-of-the-Art Natural Language Processing. *Organizational Research Methods* 27, 265–300. <https://doi.org/10.1177/10944281231155771>
- Gao, T., Fisch, A., Chen, D., 2020. Making Pre-trained Language Models Better Few-shot Learners. <https://doi.org/10.48550/ARXIV.2012.15723>

- Garcia, R., Calantone, R., 2002. A critical look at technological innovation typology and innovativeness terminology: a literature review. *Journal of Product Innovation Management* 19, 110–132. <https://doi.org/10.1111/1540-5885.1920110>
- Gault, F., 2018. Defining and measuring innovation in all sectors of the economy. *Research Policy* 47, 617–622. <https://doi.org/10.1016/j.respol.2018.01.007>
- Gentzkow, M., Kelly, B., Taddy, M., 2019. Text as Data. *Journal of Economic Literature* 57, 535–574. <https://doi.org/10.1257/jel.20181020>
- Gioia, D.A., Corley, K.G., Hamilton, A.L., 2013. Seeking Qualitative Rigor in Inductive Research: Notes on the Gioia Methodology. *Organizational Research Methods* 16, 15–31. <https://doi.org/10.1177/1094428112452151>
- Goodfellow, I., Bengio, Y., Courville, A., 2016. *Deep learning*, Adaptive computation and machine learning. MIT press, Cambridge, MA.
- Griliches, Z., 1990. Patent Statistics as Economic Indicators: A Survey. *Journal of Economic Literature* 28, 1661–1707.
- Gunathilaka, S., de Silva, N., 2025. Automatic Analysis of App Reviews Using LLMs, in: *Proceedings of the 17th International Conference on Agents and Artificial Intelligence*, 828–839. <https://doi.org/10.5220/0013375600003890>
- Guzman, J., Li, A., 2023. Measuring Founding Strategy. *Management Science* 69, 101–118. <https://doi.org/10.1287/mnsc.2022.4369>
- Haeussler, C., Assmus, A., 2021. Bridging the gap between invention and innovation: Increasing success rates in publicly and industry-funded clinical trials. *Research Policy* 50, 104155. <https://doi.org/10.1016/j.respol.2020.104155>
- Henderson, R.M., Clark, K.B., 1990. Architectural Innovation: The Reconfiguration of Existing Product Technologies and the Failure of Established Firms. *Administrative Science Quarterly* 35, 9–30. <https://doi.org/10.2307/2393549>
- Henrich, J., Heine, S.J., Norenzayan, A., 2010. The weirdest people in the world? *Behavioral and Brain Sciences* 33, 61–83. <https://doi.org/10.1017/S0140525X0999152X>
- Hirschberg, J., Manning, C.D., 2015. Advances in natural language processing. *Science* 349, 261–266. <https://doi.org/10.1126/science.aaa8685>
- Huang, L., Pearce, J.L., 2015. Managing the Unknowable: The Effectiveness of Early-stage Investor Gut Feel in Entrepreneurial Investment Decisions. *Administrative Science Quarterly* 60, 634–670. <https://doi.org/10.1177/0001839215597270>
- James, T., Qiao, Z., Shen, W., Wang, G.A., Fan, W., 2023. Competing for Temporary Advantage in a Hypercompetitive Mobile App Market. *MIS Quarterly* 47, 1177–1212. <https://doi.org/10.25300/MISQ/2022/15079>
- Jeppesen, L.B., Lakhani, K.R., 2010. Marginality and Problem-Solving Effectiveness in Broadcast Search. *Organization Science* 21, 1016–1033. <https://doi.org/10.1287/orsc.1090.0491>
- Johnson, P.E., Hassebrock, F., Durán, A.S., Moller, J.H., 1982. Multimethod study of clinical judgment. *Organizational Behavior and Human Performance* 30, 201–230. [https://doi.org/10.1016/0030-5073\(82\)90218-5](https://doi.org/10.1016/0030-5073(82)90218-5)
- Jones, C.R., Bergen, B.K., 2024. People cannot distinguish GPT-4 from a human in a Turing test. <https://doi.org/10.48550/ARXIV.2405.08007>
- Just, J., Ströhle, T., Füller, J., Hutter, K., 2024. AI-based novelty detection in crowdsourced idea spaces. *Innovation: Organization & Management* 26, 359–386. <https://doi.org/10.1080/14479338.2023.2215740>
- Kahneman, D., Slovic, P., Tversky, A., 1982. *Judgment under uncertainty: heuristics and biases*. Cambridge University Press, Cambridge.

- Khelifi, G., Jenhani, I., Messaoud, M.B., Mkaouer, M.W., 2024. Multi-label Classification of Mobile Application User Reviews Using Neural Language Models, in: Bouraoui, Z., Vesic, S. (Eds.), *Symbolic and Quantitative Approaches to Reasoning with Uncertainty*, Lecture Notes in Computer Science. Springer Nature Switzerland, Cham, pp. 417–426. https://doi.org/10.1007/978-3-031-45608-4_31
- Kim, M., DellaPosta, D., 2022. The Fickle Crowd: Reinforcement and Contradiction of Quality Evaluations in Cultural Markets. *Organization Science* 33, 2496–2518. <https://doi.org/10.1287/orsc.2021.1556>
- Kircher, T., Foerderer, J., 2024. Ban Targeted Advertising? An Empirical Investigation of the Consequences for App Development. *Management Science* 70, 1070–1092. <https://doi.org/10.1287/mnsc.2023.4726>
- Korinek, A., 2023. Generative AI for Economic Research: Use Cases and Implications for Economists. *Journal of Economic Literature* 61, 1281–1317. <https://doi.org/10.1257/jel.20231736>
- Krieger, J.L., 2021. Trials and Terminations: Learning from Competitors' R&D Failures. *Management Science* 67, 5525–5548. <https://doi.org/10.1287/mnsc.2020.3775>
- Krippendorff, K., 2019. *Content Analysis: An Introduction to Its Methodology*. SAGE, Thousand Oaks, CA. <https://doi.org/10.4135/9781071878781>
- Landis, J.R., Koch, G.G., 1977. The Measurement of Observer Agreement for Categorical Data. *Biometrics* 33, 159–174. <https://doi.org/10.2307/2529310>
- Lane, J.N., Boussioux, L., Ayoubi, C., Chen, Y.H., Lin, C., Spens, R., Wagh, P., Wang, P.-H., 2024. The Narrative AI Advantage? A Field Experiment on Generative AI-Augmented Evaluations of Early-Stage Innovations. Harvard Business School Working Paper, No. 25-001.
- Lane, J.N., Teplitskiy, M., Gray, G., Ranu, H., Menietti, M., Guinan, E.C., Lakhani, K.R., 2022. Conservatism Gets Funded? A Field Experiment on the Role of Negative Information in Novel Project Evaluation. *Management Science* 68, 4478–4495. <https://doi.org/10.1287/mnsc.2021.4107>
- Lazaridou, A., Kuncoro, A., Gribovskaya, E., Agrawal, D., Liska, A., Terzi, T., Gimenez, M., d'Autume, C. de M., Kocisky, T., Ruder, S., Yogatama, D., Cao, K., Young, S., Blunsom, P., 2021. Mind the Gap: Assessing Temporal Generalization in Neural Language Models, in: *Proceedings of the 35th International Conference on Neural Information Processing Systems*. pp. 29348–29363.
- Lee, G., Raghu, T.S., 2014. Determinants of Mobile Apps' Success: Evidence from the App Store Market. *Journal of Management Information Systems* 31, 133–170. <https://doi.org/10.2753/MIS0742-1222310206>
- Leyden, B.T., 2022. There's an App (Update) for That: Understanding Product Updating Under Digitization.
- Li, D., 2017. Expertise versus Bias in Evaluation: Evidence from the NIH. *American Economic Journal: Applied Economics* 9, 60–92. <https://doi.org/10.1257/app.20150421>
- Liu, P., Yuan, W., Fu, J., Jiang, Z., Hayashi, H., Neubig, G., 2023. Pre-train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing. *ACM Comput. Surv.* 55, 1–35. <https://doi.org/10.1145/3560815>
- Loughran, T., McDonald, B., 2011. When Is a Liability Not a Liability? Textual Analysis, Dictionaries, and 10-Ks. *The Journal of Finance* 66, 35–65. <https://doi.org/10.1111/j.1540-6261.2010.01625.x>

- Ludwig, J., Mullainathan, S., Rambachan, A., 2025. Large Language Models: An Applied Econometric Framework (No. w33344). National Bureau of Economic Research, Cambridge, MA. <https://doi.org/10.3386/w33344>
- Millimet, D.L., Parmeter, C.F., 2022. Accounting for Skewed or One-Sided Measurement Error in the Dependent Variable. *Political Analysis* 30, 66–88. <https://doi.org/10.1017/pan.2020.45>
- Miric, M., Jia, N., Huang, K.G., 2023a. Using supervised machine learning for large-scale classification in management research: The case for identifying artificial intelligence patents. *Strategic Management Journal* 44, 491–519. <https://doi.org/10.1002/smj.3441>
- Miric, M., Ozalp, H., Yilmaz, E.D., 2023b. Trade-offs to using standardized tools: Innovation enablers or creativity constraints? *Strategic Management Journal* 44, 909–942. <https://doi.org/10.1002/smj.3457>
- Mitchell, M., Krakauer, D.C., 2023. The debate over understanding in AI's large language models. *Proceedings of the National Academy of Sciences* 120, e2215907120. <https://doi.org/10.1073/pnas.2215907120>
- Mollick, E., Nanda, R., 2016. Wisdom or Madness? Comparing Crowds with Expert Evaluation in Funding the Arts. *Management Science* 62, 1533–1553. <https://doi.org/10.1287/mnsc.2015.2207>
- Moser, P., 2012. Innovation without Patents: Evidence from World's Fairs. *The Journal of Law and Economics* 55, 43–74. <https://doi.org/10.1086/663631>
- Nightingale, P., 1998. A cognitive model of innovation. *Research Policy* 27, 689–709. [https://doi.org/10.1016/S0048-7333\(98\)00078-X](https://doi.org/10.1016/S0048-7333(98)00078-X)
- OpenAI, 2023. GPT-4 Technical Report. <https://doi.org/10.48550/ARXIV.2303.08774>
- Parker, G., Van Alstyne, M., 2018. Innovation, Openness, and Platform Control. *Management Science* 64, 3015–3032. <https://doi.org/10.1287/mnsc.2017.2757>
- Peeperkorn, M., Kouwenhoven, T., Brown, D., Jordanous, A., 2024. Is Temperature the Creativity Parameter of Large Language Models? <https://doi.org/10.48550/ARXIV.2405.00492>
- Petersen, A.M., 2022. Evolution of biomedical innovation quantified via billions of distinct article-level MeSH keyword combinations. *Advances in Complex Systems* 25, 2150016. <https://doi.org/10.1142/S0219525921500168>
- Piezunka, H., Dahlander, L., 2015. Distant Search, Narrow Attention: How Crowding Alters Organizations' Filtering of Suggestions in Crowdsourcing. *Academy of Management Journal* 58, 856–880. <https://doi.org/10.5465/amj.2012.0458>
- Rosenkopf, L., McGrath, P., 2011. Advancing the Conceptualization and Operationalization of Novelty in Organizational Research. *Organization Science* 22, 1297–1311. <https://doi.org/10.1287/orsc.1100.0637>
- Samuelson, W., Zeckhauser, R., 1988. Status quo bias in decision making. *Journal of Risk and Uncertainty* 1, 7–59. <https://doi.org/10.1007/BF00055564>
- Schumpeter, J.A., 1934. *The Theory of Economic Development. An Inquiry into Profits, Capital, Credit, Interest, and the Business Cycle*. Harvard University Press, Cambridge, MA.
- Shanteau, J., 1992. Competence in experts: The role of task characteristics. *Organizational Behavior and Human Decision Processes* 53, 252–266. [https://doi.org/10.1016/0749-5978\(92\)90064-E](https://doi.org/10.1016/0749-5978(92)90064-E)
- Smith, K., 2005. Measuring Innovation, in: Fagerberg, J., Mowery, D.C., Nelson, R.R. (Eds.), *The Oxford Handbook of Innovation*. Oxford University Press, Oxford, pp. 148–177.
- Strauss, A.L., Corbin, J.M., 1998. *Basics of qualitative research: techniques and procedures for developing grounded theory*, 2nd ed. SAGE, Thousand Oaks, CA.

- Sukhov, A., Sihvonen, A., Netz, J., Magnusson, P., Olsson, L.E., 2021. How experts screen ideas: The complex interplay of intuition, analysis and sensemaking. *Journal of Product Innovation Management* 38, 248–270. <https://doi.org/10.1111/jpim.12559>
- Tao, Y., Viberg, O., Baker, R.S., Kizilcec, R.F., 2024. Cultural bias and cultural alignment of large language models. *PNAS Nexus* 3, pgae346. <https://doi.org/10.1093/pnasnexus/pgae346>
- Timoshenko, A., Hauser, J.R., 2019. Identifying Customer Needs from User-Generated Content. *Marketing Science* 38, 1–20. <https://doi.org/10.1287/mksc.2018.1123>
- Tversky, A., Kahneman, D., 1974. Judgment under Uncertainty: Heuristics and Biases: Biases in judgments reveal some heuristics of thinking under uncertainty. *Science* 185, 1124–1131. <https://doi.org/10.1126/science.185.4157.1124>
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I., 2017. Attention Is All You Need, in: *Advances in Neural Information Processing Systems*. Curran Associates, Inc.
- Verhoeven, D., Bakker, J., Veugelers, R., 2016. Measuring technological novelty with patent-based indicators. *Research Policy* 45, 707–723. <https://doi.org/10.1016/j.respol.2015.11.010>
- Von Hippel, E., 1978. A customer-active paradigm for industrial product idea generation. *Research Policy* 7, 240–266. [https://doi.org/10.1016/0048-7333\(78\)90019-7](https://doi.org/10.1016/0048-7333(78)90019-7)
- Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E., Le, Q.V., Zhou, D., 2022. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. *Advances in Neural Information Processing Systems* 35, 24824–24837.
- Wen, W., Zhu, F., 2019. Threat of platform-owner entry and complementor responses: Evidence from the mobile app market. *Strategic Management Journal* 40, 1336–1367. <https://doi.org/10.1002/smj.3031>
- Wimbauer, L., Figge, P., Haeussler, C., 2019. Distant Search, but Local Implementation? Using the Crowd's Evaluation to Overcome Organizational Limitations in the Selection of Crowdsourced Ideas, in: *ICIS 2019 Proceedings*.
- Zaggl, M.A., Malhotra, A., Alexy, O., Majchrzak, A., 2023. Governing crowdsourcing for unconstrained innovation problems. *Strategic Management Journal* 44, 2783–2817. <https://doi.org/10.1002/smj.3505>
- Zhang, M., Fan, B., Zhang, N., Wang, W., Fan, W., 2021. Mining product innovation ideas from online reviews. *Information Processing & Management* 58, 102389. <https://doi.org/10.1016/j.ipm.2020.102389>
- Zittrain, J., 2006. The Generative Internet. *Harvard Law Review* 119, 1974–2040.

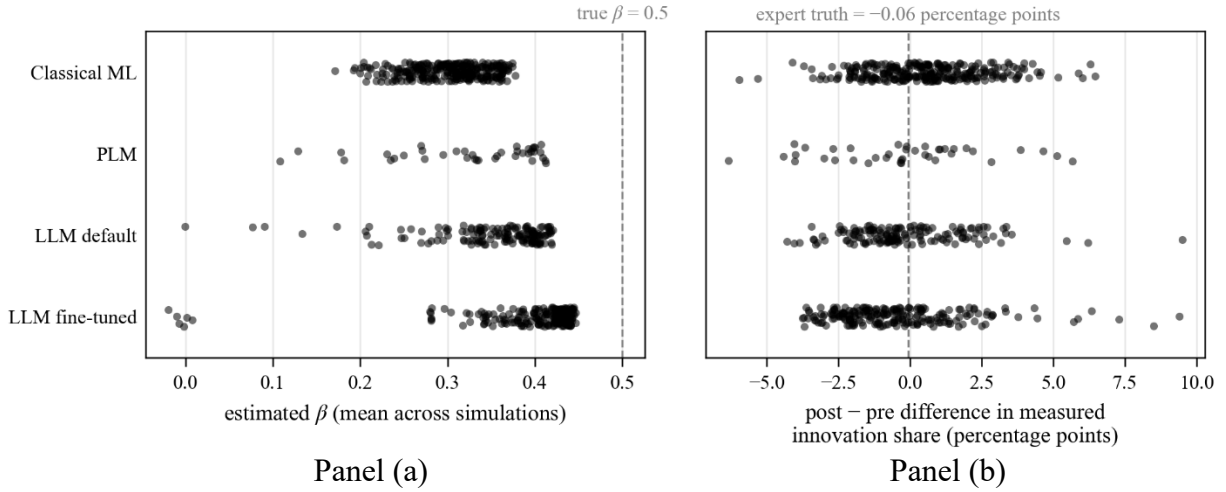
Figure 1 System prompt and prompt breakdown for software application update classification (Study 1)

<p>ONLY provide the category number (1-7) in response. Determine the category for the following app update text. If multiple categories seem applicable, always choose the lowest category number (1<2<3<4<5<6<7):</p> <ol style="list-style-type: none"> (1) Novel features – Introducing or enhancing significant functionalities that modify user experience (2) Content extensions – Adding or expanding curated or user-facing content (3) Platform and device support – Enabling compatibility with new OS versions, development kits, device types, or hardware features (4) Specific fixes of bugs – Addressing distinct known issues (5) Privacy & Security – Strengthening user data protection, permissions, or security protocols (6) General Improvements – Tweaks addressing undisclosed bug fixes, performance improvements, minor changes, or UI/UX polish (7) Marketing & Branding – Changes purely about promotional or visual branding assets <p>App Update: [RELEASE NOTE]</p>	<p>Section 1 Output definition and task description according to measurement objective</p> <p>Section 2 Expert validated data structure</p> <p>Section 3 Text data for evaluation</p>
---	--

Figure 2 System prompt and prompt breakdown for product review classification (Study 2)

<p>ONLY provide a number (0-8) in response. Categorize the following app review text by assigning the most fitting category/categories out of the following nine categories. If the text contains elements from multiple categories, provide the categories separated by ;</p> <ol style="list-style-type: none"> (0) User Opinion without specific reports, issues, suggestions – i.e., review only about good/bad perception of the app but nothing else (1) Reports of bugs, errors, or bad quality issues – i.e., something does not work in the app or is of bad quality (2) Issues of the app's monetization model – i.e., complaints or issues of how the app monetizes content (3) Suggestions for new features or content or revival of removed features (4) Customer support issues – i.e., problems or complaints regarding customer support (5) Performance issues – i.e., the app needs too much space, is too slow or similar (6) Security concerns – i.e., user is concerned about their data or privacy (7) Ethical concerns – i.e., user is concerned about practices in the app, fairness, discrimination (8) Community related issues – user asks openly for help (no feedback at customer support) <p>Review text: [REVIEW TEXT]</p>	<p>Section 1 Output definition and task description according to measurement objective</p> <p>Section 2 Expert validated data structure</p> <p>Section 3 Text data for evaluation</p>
---	--

Figure 3 Downstream estimates based on the classified innovation variable across model configurations (Study 1)



Note. Expert-labeled holdout sample of Demonstration Study 1 ($n = 1,000$). Each dot is one model configuration, grouped by model family. Panel (a): classified innovation variable serves as explanatory variable in a simulated outcome with true $\beta = .50$ (dashed line); estimates are averaged across simulation draws. Panel (b): classified innovation variable serves as outcome; dashed line marks the expert-labeled post-pre difference of -0.06 percentage points.

Table 1 Framework for mapping expert insights into LLM applications

Expert Evaluation	LLM Process Mapping	Design Decisions
<i>Task Definition</i>	Prompt Section 1: <ul style="list-style-type: none"> - Researcher describes task and defines desired output according to the measurement objective. - Task description and output definition are included in the prompt. 	<ul style="list-style-type: none"> - Expert selection criteria (number of experts, task-expertise alignment) - Prompt engineering
<i>Data Engagement</i>	Inductive Coding AND Prompt Section 2: <ul style="list-style-type: none"> - Researcher conducts inductive three-pronged (open, axial, selective) coding procedure with the data until code/thematic saturation is reached. - Data structure is validated with expert(s) to assure meaning saturation and adequate structuring of cues. - Data dimensions (or categories) are included in the prompt. 	<ul style="list-style-type: none"> - Prompt engineering
<i>Relate to Prior Knowledge</i>	Prompt Section 3 OR Manual Output Interpretation: <ul style="list-style-type: none"> - Expert assesses relevance and weight of each data dimension (or category) for the measurement. - Relevance and weights are included in the prompt or researcher interprets model output based on the relevance of dimensions for the defined task 	<ul style="list-style-type: none"> - Prompt engineering - Weight elicitation format (in-prompt vs. post-processing)
<i>Judgment Formation</i>	Output validation AND Fine-Tuning: <ul style="list-style-type: none"> - Researcher and expert validate model output and measurement - Researcher labels training and validation data for model fine-tuning on desired outcome (input: prompt + text data, output: classification/measure in expected format) 	General: <ul style="list-style-type: none"> - Model selection - Temperature setting Fine-Tuning: <ul style="list-style-type: none"> - Training data size - Data distribution
<i>Reliability Assurance</i>	Design Decisions: <ul style="list-style-type: none"> - Researcher analyzes how model selection, prompt engineering, training data, and temperature setting affect consistency rate of repeated runs - Researcher optimizes and documents design decisions for reliability 	

Table 2 Model performance for identification of innovative updates

		F1-score	Precision	Recall	Accuracy	CR
Version First Digit (1.0 → 2.0)	Literature	.211	.559	.130	.574	
Version Second Digit (1.1 → 1.2)		.472	.475	.470	.540	
Dictionary-Based		.666	.799	.571	.749	
Dictionary-Character-Based		.615	.477	.863	.526	
K-Nearest-Neighbor	Classical Machine Learning Models	.713	.828	.626	.779	
Logistic Regression		.850	.810	.895	.862	
Naïve Bayes		.835	.839	.831	.856	
Random Forest		.789	.740	.845	.802	
Support Vector Machine (SVM)		.849	.832	.868	.865	
Extreme Gradient Boosting (XGBoost)		.834	.808	.863	.850	
Convolutional Neural Network (CNN)	Deep Learning	.828	.817	.838	.847	
BERT	Transformer-Based PLMs	.899	.865	.936	.908	
ELECTRA		.889	.849	.934	.898	
RoBERTa		.902	.879	.927	.912	
XLNet		.887	.838	.943	.895	
Claude 3.5 Haiku	Large Language Models	.746	.944	.616	.816	1.000
Claude Sonnet 4		.896	.938	.858	.913	.999
Mistral Small		.765	.934	.648	.826	.992
Mistral Large		.851	.930	.785	.880	.984
Mistral Small (fine-tuned)		.887	.888	.886	.901	.984
Mistral Large (fine-tuned)		.909	.891	.929	.919	.985
GPT-4.1 Nano		.760	.952	.632	.825	.999
GPT-4.1 Mini		.869	.894	.845	.888	.987
GPT-4.1		.901	.923	.879	.915	.982
GPT-4.1 Nano (fine-tuned)		.919	.907	.932	.928	.998
GPT-4.1 Mini (fine-tuned)		.914	.888	.941	.922	.992
GPT-4.1 (fine-tuned)		.926	.894	.961	.933	.996
N		1,000	1,000	1,000	1,000	3,000

Note. The version approach follows Wen and Zhu (2019), the dictionary approach follows Kircher and Foerderer (2024), and the dictionary-character approach follows Agarwal and Kapoor (2023). For the ML- and transformer-based classifications, updates classified as 1 or 2 were considered innovative. ML-based classifiers were trained and transformer-based models were fine-tuned with a dataset $n=2,000$, with categories representative of the validation data. Embeddings for the ML classifiers were created using TF-IDF, and embeddings for the CNN were generated using glove-42b-300d. LLM temperature = 0.0, LLM seed = 94032. F1-score is reported for the first run. Consistency Rate (CR) is calculated based on three independent runs.

Table 3 Model performance for update classification

	1	2	3	4	5	6	7	Macro F1	Weighted F1	CR
K-Nearest-Neighbor	.665	.217	.629	.372	.118	.776	.167	.420	.639	
Logistic Regression	.793	.449	.673	.569	.211	.873	.000	.510	.759	
Naïve Bayes	.788	.496	.621	.563	.320	.878	.000	.524	.760	
Random Forest	.702	.232	.542	.607	.105	.816	.000	.429	.682	
SVM	.788	.463	.647	.583	.190	.873	.000	.506	.757	
XGBoost	.764	.353	.686	.605	.105	.853	.143	.501	.737	
CNN	.774	.349	.667	.457	.286	.854	.000	.484	.729	
BERT	.833	.500	.721	.750	.000	.906	.000	.530	.804	
ELECTRA	.850	.458	.691	.776	.000	.919	.000	.528	.813	
RoBERTa	.864	.647	.776	.840	.000	.926	.000	.579	.844	
XLNet	.842	.531	.739	.803	.000	.916	.000	.547	.819	
Claude 3.5 Haiku	.642	.471	.475	.498	.444	.756	.174	.494	.646	.997
Claude Sonnet 4	.865	.504	.556	.524	.571	.799	.296	.588	.758	.994
Mistral Small	.587	.441	.527	.466	.545	.738	.200	.501	.619	.971
Mistral Large	.772	.582	.637	.558	.389	.818	.348	.586	.742	.973
Mistral Small (fine-tuned)	.821	.511	.752	.745	.500	.904	.276	.644	.812	.966
Mistral Large (fine-tuned)	.863	.563	.764	.758	.583	.929	.400	.607	.845	.970
GPT-4.1 Nano	.615	.439	.463	.315	.462	.455	.333	.440	.501	.997
GPT-4.1 Mini	.798	.595	.614	.552	.522	.819	.286	.598	.753	.965
GPT-4.1	.849	.523	.741	.738	.435	.911	.400	.657	.825	.970
GPT-4.1 Nano (fine-tuned)	.876	.681	.764	.797	.538	.940	.375	.710	.865	.993
GPT-4.1 Mini (fine-tuned)	.882	.682	.779	.838	.522	.938	.400	.720	.870	.991
GPT-4.1 (fine-tuned)	.888	.602	.800	.877	.455	.949	.435	.715	.875	.988
N	368	70	58	78	15	400	11	1,000	1,000	3,000

Note. ML-based classifiers were trained and transformer-based models were fine-tuned with a dataset $n=2,000$, with categories distributed similarly to the validation data. Embeddings for the ML classifiers were created using TF-IDF, and embeddings for the CNN were generated using glove-42b-300d. LLM temperature = 0.0, LLM seed = 94032. F1-score is reported for the first run. Consistency Rate (CR) is based on three independent runs.

Table 4 Model performance for review classification

	0	1	2	3	4	5	6	7	8	M F1	W F1	CR
K-Nearest-Neighbor	.826	.431	.465	.092	.063	.000	.000	.000	.000	.208	.529	
Logistic Regression	.858	.644	.610	.481	.359	.087	.143	.000	.000	.354	.672	
Naïve Bayes	.850	.677	.553	.553	.407	.204	.129	.098	.100	.397	.687	
Random Forest	.813	.600	.602	.424	.511	.387	.000	.174	.100	.401	.645	
SVM	.821	.660	.609	.483	.356	.250	.118	.174	.000	.386	.664	
XGBoost	.846	.640	.573	.445	.356	.087	.000	.000	.000	.328	.655	
CNN	.800	.571	.620	.406	.379	.174	.091	.133	.000	.353	.623	
BERT	.895	.733	.410	.551	.000	.000	.000	.000	.000	.288	.687	
ELECTRA	.909	.767	.000	.611	.000	.000	.000	.000	.000	.254	.674	
RoBERTa	.923	.758	.697	.700	.439	.000	.000	.100	.000	.402	.766	
XLNet	.917	.764	.651	.674	.642	.000	.000	.273	.000	.436	.766	
Claude 3.5 Haiku	.924	.829	.753	.741	.622	.609	.625	.429	.333	.652	.827	.989
Claude Sonnet 4	.914	.791	.780	.732	.621	.427	.632	.411	.444	.639	.813	1.000
Mistral Small	.909	.776	.766	.690	.686	.161	.615	.386	.154	.571	.793	.989
Mistral Large	.900	.779	.745	.724	.641	.415	.667	.346	.364	.620	.798	.911
Mistral Small (fine-tuned)	.927	.814	.822	.703	.583	.519	.667	.526	.250	.646	.825	.971
Mistral Large (fine-tuned)	.937	.837	.814	.763	.476	.429	.643	.333	.500	.637	.836	.965
GPT-4.1 Nano	.898	.764	.675	.524	.300	.261	.519	.160	.154	.473	.740	.995
GPT-4.1 Mini	.905	.835	.739	.731	.651	.373	.629	.371	.256	.610	.812	.960
GPT-4.1	.916	.828	.749	.731	.727	.516	.647	.491	.370	.664	.825	.943
GPT-4.1 Nano (fine-tuned)	.938	.856	.785	.729	.781	.500	.727	.438	.375	.681	.846	.984
GPT-4.1 Mini (fine-tuned)	.944	.850	.844	.770	.793	.471	.435	.323	.267	.633	.851	.996
GPT-4.1 (fine-tuned)	.948	.872	.853	.769	.807	.500	.667	.467	.471	.706	.867	.997
N	475	240	92	146	31	19	13	19	12	1,047	1,047	3,000

Note. ML-based classifiers were trained and transformer-based models were fine-tuned with a dataset n=2,000, with categories distributed similarly to the validation data. Embeddings for the ML classifiers were created using TF-IDF, and embeddings for the CNN were generated using glove-42b-300d. LLM temperature = 0.0, LLM seed = 94032. F1-score is reported for the first run. Consistency Rate (CR) is based on three independent runs.

Online Appendix

AI-Based Measurement of Innovation: Mapping Expert Insight into Large Language Model Applications

Appendix A. Towards Large Language Models for Expert Evaluation	2
Appendix B. Data Structure and Coding Dimensions.....	6
Appendix C. Large Language Model Inference via Batch API	7
Appendix D. Hyperparameter Tuning.....	10
Appendix E. Implications of Design Decisions	15
Appendix F. Teacher-Student Distillation Setup	25
Appendix G. Summary of Fine-Tuning Decisions.....	28
Appendix H. Measurement Performance and Implications for Downstream Estimates.....	29
Appendix I. Temporal Relevance Robustness Check	33
Appendix J. Summary of Conditions for Applying the Framework	35
Appendix K. Handling Multi-Lingual Data	36
References	37

Paper Companion Repository:

https://github.com/robi979/AI-Innovation-Measurement_Extension

Appendix A. Towards Large Language Models for Expert Evaluation

Traditionally, scholars have operationalized variables at scale by using keyword-based methods to approximate the conceptual judgments that domain experts otherwise would have to make (e.g., Foerderer et al., 2018; Kircher and Foerderer, 2024). This method involves the use of predefined dictionaries that map specific terms to theoretically grounded constructs.

The approach is particularly effective when theoretical concepts and the terminology used to communicate them are well-defined and broadly recognized. However, keyword-based methods come with limitations. First, constructing comprehensive and reliable dictionaries can be challenging, especially when no related dictionary exists. The process often relies on subjective judgments, and validating dictionaries is non-trivial. Second, keyword-based approaches struggle to capture the context in which a word appears. For instance, they may fail to identify nuances such as negations or different semantic meanings depending on the surrounding text. While these approaches are computationally efficient and easily interpretable, their ability to comprehensively capture complex constructs is very limited (see Miric et al., 2023 for a detailed discussion).¹

To address these issues, scholars have increasingly turned to machine learning (ML) models. ML encompasses statistical algorithms that refine their performance through iterative “learning” from data. Depending on the research objective, scholars can choose between supervised and unsupervised techniques. Supervised algorithms classify observations into predefined categories, making them well-suited for tasks where evaluators assess content according to specific patterns or develop categorical variables (e.g., Miric et al., 2023). In contrast, unsupervised

¹ A notable recent development is curated ontologies, which organize expert-maintained vocabularies in multi-level hierarchies. For instance, the Medical Subject Headings in biomedicine are continuously revised by expert indexers and have been used to quantify innovation at scale (Petersen, 2022). Still, sustaining such ontologies requires an institutionalized curation effort which few research settings can draw on.

methods discover emergent themes in text data without relying on prior frameworks, making them useful for exploratory analyses (e.g., Hannigan et al., 2019).

Effective application of ML to textual data requires transforming language into numerical representations that algorithms can process. Early approaches, such as bag-of-words models, convert text into numeric representations by counting term frequencies, disregarding order and contextual information (Manning et al., 2008). For instance, popular models like TF-IDF (i.e., Term Frequency-Inverse Document Frequency) assign weights to words based on their frequency across documents, or use co-occurrence patterns to group words into latent themes as in latent topic distributions. However, these techniques often fail to capture deeper semantic relationships and struggle with comparing large numbers of documents due to their sparse, context-free representations (Turney and Pantel, 2010).

Embedding-based approaches have improved on these limitations by creating numerical representations (embeddings) of words that encode their relationships in a given text corpus. Based on the idea that words appearing in similar contexts tend to have related meanings (Harris, 1954), approaches (continuous models) like Word2Vec (Mikolov et al., 2013) and GloVe (Pennington et al., 2014) build on large text corpora and neural architectures to infer an embedding for every vocabulary item, analyzing how frequently they co-occur in a text corpus. Each word can then be represented as a vector based on all surrounding words (or vice versa). This results in vectors where semantically similar words are geometrically close, while unrelated words are farther apart. However, the embeddings generated by continuous models are static, meaning each word is assigned the same numerical representation regardless of the context in which it appears in downstream tasks. Simply put, the word “launch” would always have the same vector based on the initial text corpus, regardless of whether it now refers to “releasing a new product” or “propelling a rocket into space”.

Dynamically adjusting a word’s representation based on its meaning and context has become possible with the revolutionary transformer architectures (transformers). Transformers are deep neural networks that include positional encodings of each word’s relative position in the text document and a self-attention mechanism to produce rich contextual embeddings (Vaswani et al., 2017). In essence, self-attention allows the model to weigh surrounding words differently when updating a word’s representation, so the same word can have varying vector representations depending on its context. This breakthrough has enabled universal text representations across massive corpora, opening unprecedented avenues for analyzing the semantic content in text data.

Transformer-based models (transformer models) have excelled in a wide range of NLP tasks that previously required human expertise to achieve sufficient performance. Early pre-trained language models (PLM) such as BERT (Devlin et al., 2019) and its derivatives rely on an encoder-based transformer architecture. The encoder processes the input text holistically, capturing intricate relationships between words by producing dynamic, context-aware representations. This process results in high-quality embeddings that form the foundation for tasks such as text classification, sentiment analysis, named entity recognition, and text similarity—often achieving or even surpassing human expert-level performance.

More recently, LLMs, which are typically built on decoder-based transformer architectures, have attracted much attention. Models like GPT-3 (Brown et al., 2020) and its successors generate text by predicting the next (sub)word (token) from its preceding tokens. This generative capability makes LLMs highly effective for tasks such as text generation, summarization, and conversational artificial intelligence, while also performing well in other NLP tasks. Unlike encoder-based models, which typically require further task-specific fine-tuning with labeled datasets, decoder-based models can be instructed via natural language prompts. In this context, a prompt conditions the model’s response by providing contextual and directive information. Technically, the prompt

is concatenated with the input sequence, and through the self-attention mechanism, the model weights different parts of the prompt and context accordingly, thereby guiding the generation process (Brown et al., 2020; Liu et al., 2023). ‘Reacting’ to prompts, they can perform tasks directly, often without task-specific fine-tuning, significantly reducing the barriers to their use.

The scaling of model size and pre-training data has driven significant performance improvements (Brown et al., 2020), enabling these models to exhibit emergent abilities such as answering unseen questions, performing arithmetic, and reasoning over multiple steps. This evolution not only makes fine-tuning an option for many use cases (de Kok, 2025) but also provides researchers with the ability to incorporate expert knowledge into models for various applications with minimal prerequisites.

Appendix B. Data Structure and Coding Dimensions

B1 Demonstration Study 1

Figure B1 presents the data structure derived from an inductive, multi-stage analysis of software application release notes. Drawing on our sample pool of $n = 4,000$ release notes, we first conducted open coding to identify categories of informant terms. Then, we clustered these categories into broader themes and even further to aggregate dimensions, forming the data structure of this context. We coded sequentially and iterated across these stages until no new categories and relationships emerged (Gioia et al., 2013; Strauss and Corbin, 1998).

To assess validity and refine the data structure, we reviewed release-note content in light of Apple's developer guidelines and discussed typical software development workflows, update reporting, versioning logic, as well as our preliminary data structure in interviews with several domain experts (software application developers recruited via Fiverr, see Table B1). The final data structure comprises 18 second-order themes that formed seven aggregate dimensions.

----- Insert Figure B1 and Table B1 about here -----

B2 Demonstration Study 2

Figure B2 presents the data structure derived from an inductive, multi-stage analysis of user-generated product reviews of software applications. Our coding followed the procedure used in Demonstration Study 1. The final data structure comprises 20 second-order themes that formed nine aggregate dimensions.

----- Insert Figure B2 about here -----

Appendix C. Large Language Model Inference via Batch API

C1 Batch Processing

To evaluate large language model (LLM) performance at scale in both Demonstration Studies, we leveraged batch inference APIs from all three model providers included in the experiments—OpenAI, Mistral, and Anthropic—across all design decision variations.

For each provider, a systematic procedure was implemented to generate, upload, and execute batch requests for both base and fine-tuned models, adhering to the respective provider’s API documentation.² The set of evaluated models included:

- OpenAI: *GPT-3.5-turbo-0125* (default & fine-tuned), *GPT-4o-mini-2024-07-18* (default & fine-tuned), *GPT-4o-2024-08-06* (default & fine-tuned), *GPT-4.1-2025-04-14* (default & fine-tuned), *GPT-4.1-mini-2025-04-14* (default & fine-tuned), *GPT-4.1-nano-2025-04-14* (default & fine-tuned), *o3-2025-04-16* (default), *o4-mini-2025-04-16* (default)
- Mistral: *mistral-large-2411* (default & fine-tuned), *mistral-medium-2505* (default), *mistral-small-2503* (default & fine-tuned), *open-mistral-nemo-2407* (default & fine-tuned), *ministral-8b-2410* (default & fine-tuned), *ministral-3b-2410* (default)
- Anthropic: *claude-sonnet-4-20250514* (default), *claude-3-7-sonnet-20250219* (default), *claude-3-5-haiku-20241022* (default), *claude-3-5-sonnet-20241022* (default), *claude-3-haiku-20240307* (default), *claude-opus-4-20250514* (default), *claude-3-opus-20240229* (default)

Fine-tuned models were trained on datasets (distribution and size) similar to those in the main article above but included the respective prompts. Further design decisions reported in Online Appendix E included *prompt* engineering and *temperature* setting. As such we iterated each model over the *prompt* templates (default, few-shot, automatic-chain-of-thought, manual-chain-of-thought, contrastive-chain-of-thought) and *temperature* settings (0, 0.5, 1.0, 1.5). For Anthropic models, the maximum *temperature* was limited to 1.0 per provider documentation. For each model-

² See <https://platform.openai.com/docs/guides/batch> (Retrieved February 18, 2026), <https://docs.mistral.ai/capabilities/batch/> (Retrieved February 18, 2026), and <https://docs.anthropic.com/en/docs/build-with-claude/batch-processing> (Retrieved February 18, 2026) for details.

prompt-temperature combination, three repeated runs were performed on the full validation dataset, using a fixed *seed* (OpenAI and Anthropic) or *random_seed* (Mistral) parameter set to 94032 for reproducibility. For all batch jobs, *max_completion_tokens* (OpenAI) or *max_tokens* (Mistral/Anthropic) was set to 1000.

For OpenAI’s reasoning models (i.e., *o3-2025-04-16* and *o4-mini-2025-04-16*), the *temperature* parameter was omitted in all batch requests, as these models do not accept temperature control. Additionally, for these models, the *reasoning_effort* parameter was set to ‘low’.

After all batch files were constructed, they were uploaded using the respective provider’s API. For OpenAI and Mistral, each uploaded batch file produced a file identifier, which was then referenced when creating a batch classification job, specifying the model and endpoint (*/v1/chat/completions*), along with metadata for job tracking. For Anthropic, batch submission was performed via the *messages.batches.create* endpoint, directly uploading a list of parameterized requests. Unique identifiers (*custom_id*) were assigned to every request for traceability and later result matching.

C2 Post-processing and Output Cleaning

To evaluate large language model (LLM) performance at scale in both Demonstration Studies, we leveraged batch inference APIs from all three model providers included in the experiments—OpenAI, Mistral, and Anthropic—across all design decision configurations.

To ensure comparability and accurate evaluation of LLM predictions, a standardized post-processing pipeline was applied to all raw outputs from the batch API runs across providers. For each provider (OpenAI, Mistral, Anthropic), provider-specific parsing functions were implemented to extract the predicted content and align it with the corresponding sample identifiers. Within each output file, the *custom_id* field was parsed to reconstruct the relevant design configuration including the model name, prompt template, temperature setting, run index, and sample identifier.

Further the raw batch outputs (*message*) were parsed to extract the predicted content. The extracted predictions were mapped back to the holdout sample.

Given the potential variability in free-form LLM outputs, especially under more complex prompts or increased temperature values, we implemented an output cleaning procedure to reliably extract class labels. In particular, we used *regular expressions* to accommodate a wide range of output formats, correctly extracting single integer predictions, multiple class outputs separated by semicolons (as in Demonstration Study 2), as well as labels embedded in extra text and further non-numeric characters such as parentheses or stray punctuation. Any invalid, missing, or out-of-range predictions (i.e., anything not mapping to the defined target labels of the Demonstration Study) were assigned a fallback class label to ensure these were penalized in downstream performance evaluation.

Appendix D. Hyperparameter Tuning

D1 Tuning Numerical Text Representations and Hyperparameters for Classical Machine Learning Classifiers

To identify the optimal model configuration for update classification, we systematically tuned both feature representations and classifier hyperparameters using an automated and reproducible pipeline via the *scikit-learn* (Pedregosa et al., 2011) and *xgboost* (Chen and Guestrin, 2016) libraries. For the analyses in the main article, we used representative training sets consisting of $n = 2,000$ release notes and labels for Demonstration Study 1, and $n = 2,000$ reviews (including title and body) and labels for Demonstration Study 2. For the additional analysis in Online Appendix E4 of the paper, we varied the training data distribution (balanced) and size ($n = 100, 250, 500, 1,000$) in both studies. To ensure compatibility with *scikit-learn*, labels in Study 1 were zero-indexed. For the multi-label classification task in Study 2, labels were transformed into multi-hot vectors. A fixed holdout sample ($n = 1,000$) was reserved for final evaluation.

Two feature representation approaches were evaluated. First, *TF-IDF* vectorization was applied, with the following hyperparameters included in the search: *max_features* (random integers between 2,000 and 10,000), *ngram_range* ((1,1), (1,2), or (1,3)), *min_df* (1-4), and *max_df* (uniformly between 0.6 and 1.0). Second, distributed representations were obtained by averaging pre-trained *GloVe* word embeddings, considering four variants: *6B-100d*, *6B-300d*, *42B-300d*, and *840B-300d* (Pennington et al., 2014). Document vectors were precomputed and cached for each embedding.

For each representation, we evaluated *Random Forest*, *Logistic Regression*, *K-Nearest Neighbors*, *Support Vector Machine*, and *XGBoost* classifiers. *Multinomial Naive Bayes* was included for *TF-IDF* features but excluded for *GloVe* representations as the algorithm is designed specifically for discrete feature counts (such as those generated by *TF-IDF*) and does not perform

reliably on continuous or dense embeddings like *GloVe*. The hyperparameter ranges were as follows:

- Random Forest: *n_estimators* (50-500), *max_depth* (5-100), *min_samples_split* (2-20), *min_samples_leaf* (1-10), *max_features* (0.1-1.0), *criterion* ('gini', 'entropy'), *bootstrap* (True, False)
- Logistic Regression: *C* (0.01-10), *penalty* ('l2'), *solver* ('lbfgs', 'saga')
- K-Nearest Neighbors: *n_neighbors* (1-30), *weights* ('uniform', 'distance'), *leaf_size* (10-50)
- Multinomial Naive Bayes: *alpha* (1e-6-1.0), *fit_prior* (True, False)
- Support Vector Machine: *C* (0.1-10), *kernel* ('linear', 'rbf', 'poly'), *gamma* ('scale', 'auto'), *degree* (2-5)
- XGBoost: *n_estimators* (50-500), *max_depth* (3-10), *learning_rate* (0.01-0.3), *subsample* (0.5-1.0), *colsample_bytree* (0.5-1.0)

Hyperparameter optimization was conducted using scikit-learn’s *RandomizedSearchCV* with 50 randomly sampled parameter sets per classifier. For *TF-IDF* pipelines, both vectorizer and classifier parameters were tuned jointly. For *GloVe*, only classifier parameters were tuned using the precomputed document vectors. All searches used 3-fold stratified cross-validation and optimized the macro-averaged F1 score, with a fixed random seed (*random_state*=94032) for full reproducibility. Computation was parallelized for efficiency.

For Demonstration Study 1 (single-label), standard scikit-learn classifiers were used. For Demonstration Study 2 (multi-label), all classifiers were wrapped with *OneVsRestClassifier*, and multi-hot encoded targets were used as labels. The best configuration for each representation, embedding, classifier, and training set was selected based on cross-validation performance, retrained on the full training data, and evaluated on the holdout sample.

----- Insert Table D2 about here -----

Table D2 summarizes the best-performing hyperparameter configurations identified via randomized search for the machine learning classifiers used in the analyses in the main article. For each classifier, we report the macro-averaged F1 scores from both cross-validated training and evaluation on the holdout sample. Comprehensive details including all tested configurations, cross-

validation results, search parameters, predictions on the holdout sample, and summary metrics are provided in the online repository to ensure full reproducibility.

D2 Tuning Hyperparameters for Convolutional Neural Network Classifier

To identify optimal configurations for neural network-based text classification, we performed automated hyperparameter tuning of a convolutional neural network for sentence classification (*TextCNN*) with pre-trained embeddings (Kim, 2014). Data preparation was not repeated; the same training and validation splits as for the machine learning classifiers were used for the *TextCNN* runs.

For each dataset, the pipeline consisted of a custom tokenizer and the CNN implemented via the *skorch scikit-learn* library (Tietz et al., 2017). Tokens were mapped to 300-dimensional *GloVe* embeddings (42B-300d), which were preloaded and assigned to each vocabulary item. The following parameter ranges were included in the search:

- Tokenizer: *num_words* (10,000-30,000), *seq_len* (100-300)
- CNN architecture: *n_filters* (64-256), *kernel_size* (3-8), *dense_units* (32-256), *dropout* (0.1-0.5), *trainable* (True, False)
- Training: *batch_size* (32-128), *max_epochs* (5-15)

The hyperparameters were jointly optimized using scikit-learn’s *RandomizedSearchCV*, sampling 30 parameter configurations per run.³ All searches used 3-fold stratified cross-validation and optimized the macro-averaged F1 score, with a fixed random seed (*random_state*=94032) for full reproducibility. Computation was parallelized for efficiency.

In Demonstration Study 1, the network was trained using categorical cross-entropy loss for single-label classification. In Demonstration Study 2, a multi-label extension of CNN with a binary cross-entropy loss function (*BCEWithLogitsLoss*) was employed to accommodate multi-hot target

³ Given the increased computational effort associated with neural model training, 30 randomized hyperparameter configurations were evaluated for each CNN pipeline, compared to 50 for classical models.

vectors. The configuration achieving the best mean cross-validated macro-averaged F1 score was retrained on the full training data and evaluated on the holdout sample.

----- Insert Table D3 about here -----

Table D3 summarizes the best-performing hyperparameter configuration identified via *RandomizedSearchCV* for the Convolutional Neural Network used in the analyses in the main article. Macro-averaged F1 scores from both cross-validated training and evaluation on the holdout sample are reported. Comprehensive details including all tested configurations, cross-validation results, search parameters, predictions on the holdout sample, and summary metrics are provided in the online repository to ensure full reproducibility.

D3 Tuning Hyperparameters for Pre-Trained Language Models

To identify optimal configurations for transformer-based text classification, we fine-tuned several pre-trained language models (PLMs) on the same training and validation splits as the classical machine learning and CNN experiments.

We evaluated four widely used pre-trained transformer models—*BERT-base-cased* (Devlin et al., 2019), *RoBERTa-base* (Liu et al., 2019), *XLNet-base-cased* (Yang et al., 2019), and *ELECTRA-base-discriminator* (Clark et al., 2020)—using the Hugging Face *transformers* library (Wolf et al., 2020). For each model, the associated tokenizer was used to preprocess text, truncating or padding each sample to a fixed *maximum_length* (=512 tokens). In Demonstration Study 1, labels were zero-indexed, and models were trained using a standard cross-entropy objective. In Demonstration Study 2, labels were transformed to multi-hot vectors, and models were configured for multi-label classification using a sigmoid output and binary cross-entropy loss.

Hyperparameter optimization was performed using *Optuna* (Akiba et al., 2019) with 5 trials per model and training set. For each trial, the following parameters were tuned: *learning_rate* (log-uniform between 1e-5 and 5e-5), *batch_size* (8 or 16), *num_train_epochs* (3-5), and *weight_decay*

(0.0-0.01). Each configuration was trained and evaluated using the Hugging Face *Trainer* framework, and the macro-averaged F1 score on the holdout sample was used as the selection criterion. All experiments were seeded (*random_state*=94032) for reproducibility. For each model and training split, the configuration with the highest validation macro-averaged F1 score was selected and saved for downstream evaluation.

----- Insert Table D4 about here -----

Table D4 summarizes the best-performing hyperparameter configurations identified via *Optuna* trials for the PLMs used in the analyses in the main article. For each model, we report the macro-averaged F1 score of the selected configuration on the holdout sample. Comprehensive details including all tested configurations, cross-validation results, search parameters, predictions on the holdout sample, and summary metrics are provided in the online repository to ensure full reproducibility.

Appendix E. Implications of Design Decisions

Design decisions, such as model selection, prompt engineering, temperature settings, training data distribution, and data size, are critical when applying the proposed evaluation framework because they can affect the model’s performance, output consistency, and applicability to specific tasks. These decisions shape how well the model aligns with the intended use case and determine the quality and interpretability of its outputs (Carlson and Burbano, 2026; Cheng et al., 2024). Therefore, transparency in design decisions is required. For reviewers and editors tasked with verifying approaches, understanding these decisions is essential to critically assess the validity of the results, as well as to ensure that the chosen decisions are appropriate for the research objectives.

E1 Model Selection

When applying the proposed evaluation framework, researchers have to select an appropriate LLM from the numerous available models from providers like OpenAI (GPT), Anthropic (Claude), and Mistral (Ministral/Mistral). Their models differ in attributes such as model age, model size in terms of parameters and pre-training data scale, reasoning capabilities, and general evaluation benchmarks⁴. In both demonstration studies, we evaluate the model families available from each provider at the time of analysis (see Appendix C).

----- Insert Table E5 about here -----

⁴ General-purpose evaluation of language models relies on standardized benchmarks such as GPQA, Humanity’s Last Exam, and Massive Multitask Language Understanding (MMLU). These tests probe a wide range of competencies through multiple-choice and short-answer questions that span dozens of academic and professional subjects. Each item is written by domain experts and linked to a single, unambiguous, and easily verifiable answer that cannot be retrieved through a quick web search, ensuring that scores do not reflect a simple information lookup. Complementing these broad assessments are specialized benchmarks that focus on domain-specific abilities—for example, mathematics (e.g., GSM-8K), programming (e.g., HumanEval), or creative writing. They emphasize complex problem-solving within their respective fields. Thought experiments such as the Turing Test add a qualitative dimension by examining how convincingly a model can mimic human-like understanding and conversation. For fair comparison, models are evaluated in their default configurations. Past developments have shown that newer and larger-sized model versions or explicitly reasoning-optimized models tend to achieve the highest scores.

Table E5 displays the performance results by model provider for both demonstration studies. The results confirm the earlier observations from Tables 3 and 4 that larger models outperform their smaller counterparts in default configurations. For example, GPT-4o demonstrated notably higher performance (Macro F1-Scores of (1) .602 and (2) .640) than GPT-4o-mini (Macro F1-Scores of (1) .484 and (2) .576) or the predecessor GPT-3.5 Turbo (Macro F1-Scores of (1) .439 and (2) .564). This pattern holds consistently across Claude and Mistral model families. However, model recency does not guarantee superior performance within similar size categories. In Demonstration Study 2, Claude Sonnet versions showed comparable results across Sonnet 3.5 (Macro F1-score = .649), Sonnet 3.7 (Macro F1-score = .638), and Sonnet 4 (Macro F1-score = .631), while in Demonstration Study 1 earlier versions even outperformed the newest model, with Sonnet 3.5 achieving a Macro F1-score of .644 and Sonnet 3.7 achieving .613, compared to Sonnet 4's score of .588 (Table 3, main article).

Fine-tuning significantly improves model performance across all tested models. Most remarkably, fine-tuned GPT-3.5 models exceeded the performance of GPT-4.1 models in Demonstration Study 1 and nearly matched their performance levels in Demonstration Study 2. Smaller Mistral models (Ministral and Open Mistral Nemo), designed primarily for local implementation with minimal hardware requirements, initially performed poorly with F1-scores below .4 in Demonstration Study 1. However, these models showed substantial performance improvements through fine-tuning, approaching the performance levels of larger fine-tuned Mistral model classes.

The performance of new reasoning models varies across providers. While Claude Opus 4 underperforms compared to Sonnet models in Demonstration Study 1 and matches them in Demonstration Study 2, o3 achieved performance values close to fine-tuned GPT models. This capability simplifies large-scale text analysis implementation within our framework without

requiring additional training data. However, reasoning models are considerably more resource-intensive, requiring up to several hundred times more tokens to generate outputs similar to non-reasoning models.

Taken together, larger-sized or reasoning models already achieve strong results in their default versions. Provider differences are minor, with OpenAI’s GPT models slightly outperforming others particularly when fine-tuned. Notably, when fine-tuned on adequately sized, task-specific datasets, mid- and large-sized models exhibited negligible performance differences regardless of provider or size. Consequently, researchers applying our framework should systematically monitor multiple LLM candidates to identify the best fit for their evaluation tasks rather than assuming that higher parameter count, more recent releases, or provider reputation automatically translates into superior outcomes. Appendix I extends this comparison to the most recent model generation.

In addition to performance considerations, practical factors such as usage costs, energy consumption, and model accessibility are also worth considering. The majority of the models utilized in our demonstration studies are proprietary, accessed via APIs with usage-based fees, typically lower for smaller-sized models. Many LLM candidates are also available as open-weight, downloadable, and runnable on local computing infrastructure. However, while smaller models such as Ministral 8B or Open Mistral Nemo are compatible with standard computational resources, mid-sized open-weight models like Mistral Small already require significantly enhanced computational capabilities, especially when fine-tuning them.⁵ Due to such resource constraints, our demonstration studies did not include locally run tests of models comparable in size to those

⁵ Running Mistral Small on a single GPU requires at least 44 GB of GPU RAM (see <https://huggingface.co/mistralai/Mistral-Small-Instruct-2409> (Retrieved February 18, 2026) for details). Fine-tuning Mistral Small is optimized for multi-GPU, single-node training setups. For best efficiency, it is recommended to use an A100 or H100 GPU (see <https://github.com/mistralai/mistral-finetune> (Retrieved February 18, 2026) for details).

analyzed via APIs. Still, we expect their performance to align with our observed results from proprietary models. Since their architectures are identical, performance discrepancies would stem from differences in computing infrastructure.

E2 Prompt Engineering

Prompt engineering refers to the strategic modification of prompts to enhance the performance of LLMs on specific tasks (Brown et al., 2020). Studies in several contexts demonstrate that these modifications can yield significant performance improvements for LLM applications (see, for example, the systematic survey by Sahoo et al., 2024). In both demonstration studies, we assessed different prompt engineering techniques using OpenAI’s GPT models: few-shot prompting, automatic chain-of-thought, manual chain-of-thought, and contrastive chain-of-thought. Few-shot prompting involves presenting the model with a limited number of example input-output pairs (Brown et al., 2020). Automatic chain-of-thought prompting aims at generating intermediate reasoning steps by adding a “Let’s think step by step” instruction at the end of the prompt (Zhang et al., 2022). In contrast, manual chain-of-thought prompting integrates a specific logical reasoning chain into the prompt (Wei et al., 2022). Contrastive chain-of-thought prompting adds learning from mistakes by providing both valid and invalid reasoning explanations alongside the original prompt (Chia et al., 2023). The prompt texts for each technique are documented in Tables E6 and E7.

----- Insert Tables E6, E7, & E8 about here -----

Table E8 displays the performance results by prompt engineering technique for both demonstration studies. For default models, few-shot prompting showed slight performance improvements compared to the “default” prompt. However, it is important to understand that few-shot prompting introduces the risk of selection bias stemming from the choice of examples. Chain-of-thought prompting yielded mixed results. While GPT-4.1 Mini and GPT-4.1 performance

decreased with automatic and manual variants but improved with contrastive chain-of-thought prompting in Study 1, all chain-of-thought variations slightly improved performance in Study 2. Notably, GPT-4.1 Nano performance dropped strongly with chain-of-thought prompting in Study 1 (baseline Macro F1-Score with the default prompt = .440).

Interestingly, prompt engineering had only minimal performance implications for fine-tuned models, even though these models had been fine-tuned with a different prompt. Fine-tuned models thus appear largely robust to prompt variation once prompt-specific training has taken place. However, the fine-tuned GPT-4.1 Nano model showed a different pattern, with performance declining when prompts deviated from training data. In Study 1, the ‘default’ prompt achieved a Macro F1-score of .710, while alternative prompt variations: few-shot (.650), automatic chain-of-thought (.593), manual chain-of-thought (.554), and contrastive chain-of-thought (.649) scored lower. Similar declines, though less pronounced, appeared in Study 2.

The results highlight that prompt engineering can impact performance across different models and contexts, even for fine-tuned systems. However, for larger models we observe that when prompts are carefully crafted and validated with expert input—particularly by guiding through data structures and classification patterns—the potential for performance gains through prompt engineering appears limited.

E3 Temperature Value

The temperature is a hyperparameter used in language models to control how random or predictable the generated outputs are (Ackley et al., 1985). When models produce outputs, they first compute raw scores (logits) for each possible next token. These logits are converted into probabilities using the softmax function, which normalizes them so that they sum to 1. The temperature parameter influences the shape of this probability distribution by redistributing the probabilities across tokens. A higher temperature (> 1) flattens the distribution, increasing uncertainty and randomness.

Conversely, a lower temperature (< 1) sharpens the distribution, emphasizing tokens with higher probability and reducing randomness.⁶ Temperature values typically range from 0 to 2, though some providers use different scales (e.g., Claude models cap at 1). Setting temperature = 0 removes randomness entirely, causing the model to always choose the token with the highest probability (greedy sampling). Conversely, higher temperature settings can introduce randomness, enabling more creative or unexpected outputs (Peeperkorn et al., 2024). However, increased randomness comes at the expense of output reliability across repeated runs, which is a critical factor in research-oriented evaluations.

We analyze the implications of this setting in our demonstration studies by gradually increasing temperature values. Figure E3 displays F1-scores of the first run and Consistency Rate (CR) across three independent runs for temperature values of 0.0, 0.5, 1.0, and 1.5 for both demonstration studies. The metrics indicate that higher temperature settings do not improve model performance and instead result in a slight decline in performance alongside a reduction in reliability as measured by the Consistency Rate for some models. Most changes are minor, with OpenAI’s GPT models in particular showing stable performance and reliability. For example, in Demonstration Study 1, our framework applied to a GPT-4.1 (default) model shows a decline in macro averaged F1-score from .657 at temperature = 0.0 to .638 at 0.5, .636 at 1.0, and .650 at 1.5. The Consistency Rate changes from .970 at temperature = 0.0 to .98 at 0.5, .979 at 1.0, and .968 at 1.5. In contrast, a more pronounced decrease in Consistency Rate is evident for the Claude models and Mistral Large (default). For instance, the Consistency Rate of Mistral Large falls from .973 at

⁶ *Top_p* is a further parameter used to adjust the output distribution. The *top_p* parameter selects a subset of tokens whose probability exceeds a threshold *p*. A lower *top_p* value restricts generation to the most likely tokens and thus reduces randomness, while a higher value allows more tokens to be considered. For instance, a *top_p* of 0.2 means only the tokens that together account for the top 20% of the probability mass are considered. In practice, since both temperature and *top_p* influence the output by adjusting the effective token probability distribution, tuning them simultaneously can lead to redundant effects.

temperature = 0.0 to .828 at temperature = 1.5 in Demonstration Study 1, and from .911 to .611 in Demonstration Study 2.

----- Insert Figure E3 about here -----

Practically, this decrease in consistency means that at higher temperature settings, models like Mistral Large and Claude are less likely to generate the same output for identical input across runs, which undermines reproducibility. When operationalizing text-based variables this variability can introduce noise, making it difficult to reliably measure constructs, replicate results, or compare findings across studies. In addition, increased randomness raises the effort required for output cleaning, as deviations from the defined format become more frequent. This problem is particularly pronounced at the maximum temperature setting (2.0), where our tests revealed that model outputs often deviated so strongly from expected formats that label identification became unfeasible, ultimately rendering these outputs unusable for analysis and leading to their exclusion from further evaluation. Based on these findings, we recommend that researchers adopt a conservative default temperature (≤ 0.5) for tasks requiring strict output structure or having high reproducibility demands, unless a higher value is conceptually justified.

E4 Training Data Distribution and Data Size

As demonstrated in previous analyses across both demonstration studies, LLMs fine-tuned with additional training data can lead to substantial improvements in performance. Likewise, earlier ML and PLM approaches have performed well in both studies. However, when the distribution of the training data reflects that of the real-world holdout sample, some labels become underrepresented. Models that rely heavily on these training data distributions tend to perform worse on underrepresented labels compared to those labels that have sufficient representation in the training set.

To address this imbalance, two primary approaches can be considered (e.g., Johnson and Khoshgoftaar, 2019). The first approach focuses on the algorithmic level, where model weights are adjusted to assign greater importance to underrepresented classes, thereby imposing heavier penalties for classification errors related to these classes in the training or fine-tuning process. Essentially, this method instructs the model to pay greater attention to rare labels by amplifying the consequences of mistakes associated with them. The second is at the data level, adjusting the training data directly. This involves modifying the distribution of the training set through strategic oversampling of underrepresented labels, thereby enhancing model exposure to these less frequent examples. This method is particularly relevant and effective when algorithm-level adjustments are infeasible, as is often the case with fine-tuning proprietary LLMs.

Building on these considerations, it is important to recognize that, in practice, particularly data-level approaches depend on the availability of sufficiently large datasets. In previous analyses, we deliberately created extensive training and validation datasets to avoid performance differences arising solely from data limitations. However, creating large training and validation datasets is often time-consuming, complex, or constrained by data or the availability of human coders, depending on the evaluation task. To better understand how data constraints interact with distributional characteristics, we systematically reduced the size of the training datasets for both balanced and representative class distributions. Specifically, we generated subsamples with $n = 1,000$, $n = 500$, $n = 250$, and $n = 100$, maintaining either a balanced or representative class distribution.

----- Insert Figure E4 about here -----

Figure E4 presents the macro-averaged F1-scores as training data size increases, for both balanced and representative class distributions across both demonstration studies.⁷ In Demonstration Study 1 with the representative (real-world) class distribution, clear differences between models emerge. The fine-tuned GPT-4.1 Nano and Mini models outperform their default counterparts even with smaller sample sizes, showing steady gains up to $n = 500$, a slight decline at $n = 1,000$, and reaching peak performance at $n = 2,000$. The larger GPT-4.1 model also starts above its default version with small sample sizes, dips below default at $n = 500$, but then shows a steady increase, plateauing from $n = 1,000$ onwards. In contrast, both fine-tuned Mistral models initially underperform relative to their default versions, only surpassing default performance at the largest sample size ($n = 2,000$). When using a balanced class distribution, performance increases more smoothly for all models as sample size grows, and maximum F1-scores are higher than with the representative distribution. Still, the Mistral Small model does not surpass its default until $n = 1,000$, and Mistral Large until $n = 2,000$. In Demonstration Study 2, balanced distributions again yield steady growth and higher peak F1-scores compared to representative distributions, and no fine-tuned model underperforms its default version at any sample size. The representative distribution in Study 2 shows less pronounced overall gains, echoing the overall moderate improvements seen in Study 1.

Taken together, larger-sized or reasoning models already achieve strong results in their default versions. Provider differences are minor, with OpenAI’s GPT models slightly outperforming others particularly when fine-tuned. Notably, when fine-tuned on adequately sized, task-specific datasets, mid- and large-sized models exhibited negligible performance differences regardless of provider or size. Consequently, researchers applying our framework should

⁷ Additional analyses for variations in training data size and distribution for classical ML models, CNN, and earlier PLMs are presented in Figure E5.

systematically monitor multiple LLM candidates to identify the best fit for their evaluation tasks rather than assuming that higher parameter count, more recent releases, or provider reputation automatically translates into superior outcomes.

In addition to performance considerations, practical factors such as usage costs, energy consumption and model accessibility are also worth considering. The majority of the models utilized in our demonstration studies are proprietary, accessed via APIs with usage-based fees, typically lower for smaller-sized models. The tested Mistral models are also available in open-weight versions, and can be run on local computing infrastructure. However, while smaller models such as Ministral 8B or Open Mistral Nemo can operate effectively on standard computational resources, mid-sized open-weight models like Mistral Small already require significantly enhanced computational power, especially when fine-tuning them. Due to such resource constraints, our demonstration studies did not include local testing of models comparable in size to those analyzed via APIs. Still, we expect their performance to align with our observed results from proprietary models, particularly given models such as Mistral’s offerings are available in both proprietary and open-weight versions. Since their architectures are identical, any performance discrepancies would likely stem from variations in computing infrastructure.

Appendix F. Teacher-Student Distillation Setup

Both demonstration studies show that the highest-performing applications of our framework rely on proprietary, frontier LLMs that are pre-trained on extremely large corpora. While frontier models are expected to perform best in this setting, in many cases they are accessible only via provider APIs, which limits control over model versioning and may create uncertainty regarding long-run reproducibility. Moreover, deploying open-weight models at comparable capability levels often requires substantial hardware resources or cloud infrastructure, making large-scale inference costly or operationally complex. At the same time, smaller models and earlier text-classification approaches typically underperform unless substantial expert-labeled training data is available—data that is often expensive or infeasible to obtain when expert time is scarce.

One opportunity to address these concerns is to use an LLM-based application of our framework—once it achieves high expert-aligned performance—as a labeling instrument to expand training data for more lightweight approaches (e.g., Durvasula et al., 2024). The objective of this extension is not to surpass the teacher LLM’s performance, but to produce a cost-efficient “clone”—a student model that approximates the teacher model’s expert-mimicking evaluations while being locally deployable and stable over time. Conceptually, this approach transfers the teacher model’s ability to operationalize expert cues (as encoded in our framework) into models that are cheaper to run and easier to reproduce. This teacher-student distillation setup yields three practical benefits. First, it improves control and replicability because student models can be run locally, independent of API provider changes. Second, it reduces marginal cost for large-scale measurement once a student model is trained. Third, it can retain much of the evaluation quality to the extent that student models learn the teacher’s decision boundary from sufficiently large pseudo-labeled datasets.

We implement this approach in Demonstration Study 1. Specifically, we use one of the best-performing framework specification as a teacher to generate labels for progressively larger sets of additional release notes. We then train and tune a set of student models, including classical ML baselines and smaller transformer-based models, on these pseudo-labeled datasets and evaluate them on the same expert-labeled holdout sample used throughout the study.

----- Insert Figure F6 about here -----

Figure F6 plots macro-averaged F1-Score as a function of scaled training size (2k expert labels; 10k and 20k teacher-generated labels). Two patterns emerge. First, teacher labeling improves student performance broadly. Relative to the 2k baseline, the classical ML student models move into the mid-.60 Macro F1-Scores at 10k (Naïve Bayes = .651; SVM = .664; XGBoost = .624), and the PLM student models improve more strongly (BERT = .710; XLNet = .736). Second, additional scale from 10k to 20k yields more model-specific returns. The PLM students continue to close the gap to the teacher model benchmark (dashed line at Macro F1 = .744). In fact, BERT rises further to a Macro F1-Score = .743 (essentially matching the benchmark), while XLNet remains high at .733.

Two additional results are noteworthy. GPT-4.1 Nano benefits from additional teacher model-labeled data and slightly exceeds the teacher benchmark at 20k (Macro F1 = .751), indicating that distillation can transfer much of the teacher model’s decision boundary into a smaller, cheaper-to-run model when sufficient pseudo-labeled data are available. The open-weight Ministral 8B model also improves with scale (Macro F1-Score = .623 at 20k), but its gains are comparatively muted in this setup, indicating that not all compact open-weight models may absorb the teacher signal equally well in this setting—potentially due to architectural constraints or optimization limits defined by pre-training (Mirzadeh et al., 2019).

Overall, the results indicate that teacher-generated labels can materially reduce the dependence on scarce human expert labeling while enabling locally deployable models that approach teacher-level measurement quality. However, with any distillation approach, this procedure inherits the teacher model’s strengths and weaknesses, meaning that systematic teacher model errors can be propagated into the student models. For this reason, we recommend maintaining a fixed expert-labeled benchmark for periodic auditing of disagreement cases when scaling the approach.

Appendix G. Summary of Fine-Tuning Decisions

----- Insert Table G9 about here -----

Appendix H. Measurement Performance and Implications for Downstream Estimates

H1 Simulation Design

With the simulation we aim to quantify how the measurement error of each evaluated configuration can impact downstream estimates in empirical analyses. For case (i), we adopt a *plasmode* design⁸ (Franklin et al., 2014; Vaughan et al., 2009) in which a simulation builds on real data and only one component is constructed, so that its true properties are known. In our setting, the constructed component is the outcome, which we generate from the expert labels with a defined effect of innovation-related updates. The classifications of each configuration and their errors are real. Estimating the outcome model with these classifications in place of the expert labels then shows how each configuration’s errors distort the estimate.

For each of the $n = 1,000$ release notes in the Demonstration Study 1 holdout we observe the expert label and every configuration’s ($n = 676$) actual classification (see Appendix E). The 676 configurations comprise 155 default LLM configurations (models, prompt designs, and temperature settings), 211 fine-tuned LLM configurations (additionally varying training data size and composition), 270 classical ML configurations (algorithms and training data), and 40 PLM configurations. The binary indicator marks innovation-related updates (dimensions 1–2 of the coding scheme, Appendix B). For case (i), six configurations produce no variation in the resulting binary measure and thus do not permit estimation of a regression coefficient, leaving 670 estimable configurations.

We construct a synthetic outcome

⁸ Plasmode simulations originate in biostatistics and pharmacoepidemiology, where methods are evaluated on real datasets in which one component is replaced by a construct with known properties. The design fits our purpose as it preserves each configuration’s empirical error structure, which a fully synthetic simulation would have to assume, while providing the known benchmark that real applications lack.

$$y_i = \beta x_i^* + \varepsilon_i \quad (1)$$

from expert labels x_i^* with $\beta = .50$ and ε_i drawn independently for each release note from a normal distribution with mean zero and variance chosen such that innovation-related updates explain $R^2 = .15$ of the outcome variance. We then estimate Equation (1) with each configuration's classification in place of x_i^* . Estimates are averaged over 1,000 simulation draws with noise realization shared across configurations within each draw, so that differences between configurations reflect only their error structures. Because the true coefficient is known by construction, deviations from it can be directly interpreted as bias.

Case (ii) requires no synthetic component. We split the holdout sample at a fixed date (April 26, 2021) into an earlier (56% of release notes) and a later period (44%) and compute for each configuration the change in the measured innovation share between the two periods. The expert labels provide the true change of $-.06$ percentage points. Sampling uncertainty in the measurement distortion is assessed using 2,000 paired bootstrap resamples, stratified by period. Within each resample, we recompute both the expert-labeled and configuration-based changes and calculate their difference. We further decompose each measured change into the expert-labeled true change, the attenuation induced by baseline false-positive and false-negative rates, and the additional contributions from drift in false-positive and false-negative rates between periods.

Two assumptions warrant attention. First, the synthetic outcome serves only to establish a known benchmark against which bias can be measured. The simulation therefore demonstrates how measurement error propagates into estimates, not the magnitude of bias in any particular identification strategy. Second, the split date carries no substantive interpretation. It provides two subsamples across which the stability of error rates can be assessed, as would any partition along the cells of a design.

H2 Innovation-Related Updates as Regressor (Case i)

Table H10 reports one representative configuration per model family: default and fine-tuned GPT-4.1 configurations, the best-performing classical ML configuration, and BERT as the PLM benchmark reported in Table 2 of the main article. Column β reports the estimated coefficient over the 1,000 simulation draws. The distance from the set $\beta = .50$ reflects the bias each configuration's errors produce. The remaining columns report the 95% simulation interval and the error-type analyses.

----- Insert Table H10 about here -----

For each configuration, we isolate the contribution of one error type by correcting the other and re-estimating. Column $\beta^{FN\ only}$ reports estimates in which only the false negatives remain and all false positives are corrected. Column $\beta^{FP\ only}$ reports estimates in which only the false positives remain and all false negatives are corrected. The relative contribution of both error types varies across configurations. For the default GPT-4.1 configuration, false positives and false negatives produce similar attenuation, with somewhat greater attenuation from false negatives. For the fine-tuned GPT-4.1, classical ML, and BERT configurations, false positives account for a larger share of the bias. In all cases, both error types attenuate the estimate toward zero.

H3 Innovation-Related Updates as Outcome (Case ii)

Table H11 reports the same four configurations for case (ii). For each configuration, we report the change in the measured innovation share between periods and its distortion relative to the expert-labeled change of $-.06$ percentage points. The 95% bootstrap confidence intervals are obtained from 2,000 paired resamples and refer to this measurement distortion. The configurations differ substantially in their point estimates of distortion, but none of the four confidence intervals exclude zero.

----- Insert Table H11 about here -----

To locate the source of the measurement-induced changes, the remaining columns decompose each estimate into the contribution of baseline measurement error and drift in false-positive and false-negative rates between the periods. Because the true change is close to zero, the contribution of baseline error is negligible for all four configurations. Differences between the measured and expert-labeled changes arise almost entirely from changes in error rates across periods. For the default GPT-4.1 configuration, the distortion stems primarily from false-negative drift, whereas for the fine-tuned GPT-4.1 configuration both false-positive and false-negative drift contribute. The classical ML configuration produces a positive distortion driven mainly by false-negative drift, while BERT exhibits comparatively stable error rates and consequently almost no distortion. Notably, the fine-tuned GPT-4.1 configuration produces the largest distortion among the four configurations despite its higher aggregate performance. Whereas aggregate performance provides a useful indication of suitability in case (i), the suitability of a configuration for a comparative design depends more strongly on the stability of its error rates across the compared periods. Drift in error rates can arise when classification errors depend on text characteristics whose distribution shifts between periods, such as the length or composition of release notes. Because period-specific error rates are estimable on the holdout sample, such distortions can be detected before analyses on the full corpus.

Appendix I. Temporal Relevance Robustness Check

Because model capabilities and versions evolve rapidly, a relevant question is whether our framework’s performance depends on a specific model snapshot or whether it generalizes across model generations. We already implicitly tested for this concern in the analysis of the model selection design decision (Online Appendix E1). However, over the course of developing the framework and conducting the two demonstration studies, several LLM providers released new model versions (e.g., OpenAI’s GPT-5 series). To assess temporal robustness, we re-ran the baseline specification of both demonstration studies on the same validation data for the most recent generation of LLMs.

----- Insert Table I12 about here -----

Table I12 reports performance (macro-averaged F1-Scores) and the CR for each demonstration study. Column (1) shows the results for the multi-class version of the update classification; Column (2) shows the results for the review classification. Across both demonstrations, the results fall in the same performance range as the LLM baselines reported in Tables 3 and 4 of the main article, indicating that our framework’s effectiveness is not specific to a single model snapshot but generalizes to newer model generations.

In Demonstration Study 1, the frontier models reach Macro F1-Scores between .467 and .642 (Weighted F1-Scores between .516 and .822) with consistently high run-to-run agreement ($CR \geq .817$). GPT-5.2 and Claude Opus 4.6 (reasoning) achieve the strongest overall performance (Macro F1 = .642 and .636, respectively). These values are broadly comparable to the strongest non-fine-tuned LLM baselines in Table 3 of the main article, though they remain below the fine-tuned GPT-4.1 variants.

In Demonstration Study 2, the newest models achieve Macro F1-Scores between .605 and .676 (Weighted F1-Scores between .801 and .843), with high consistency overall ($CR \geq .829$). GPT-5.2

again performs at the top end (Macro F1-Score = .676; Weighted F1-Score = .841), closely matching the best non-fine-tuned models in Table 4 of the main article but not exceeding the strongest fine-tuned models.

Notably, CR is lower for GPT-5 Mini and GPT-5 Nano, aligning with the baseline constraint that these models were only available at temperature = 1.0, whereas all other models can use temperature = 0.0.

Appendix J. Summary of Conditions for Applying the Framework

----- Insert Table J13 about here -----

Appendix K. Handling Multi-Lingual Data

Our demonstration studies restrict attention to English-language text. This restriction serves baseline comparability because the dictionary and classical supervised ML methods against which we benchmark are developed for and validated on English text, so that only an English corpus provides the appropriate setting for comparing approaches. The framework itself is not language-specific. The scope conditions of the framework concern the relationship between text and construct and the availability of reliable expert judgment (see Section 5 in the main article for a detailed discussion). None depends on the language of the corpus.

What varies across languages is model performance. Multilingual evaluations document that frontier LLMs perform at near-English levels for high-resource languages while performance degrades for low-resource languages (Ahuja et al., 2023; Bandarkar et al., 2023). Researchers applying the framework to non-English corpora should therefore treat language capability as an empirical question to be settled within the framework’s own validation logic. As such, the holdout sample used for performance assessment should represent the languages of the target corpus, so that measurement quality is demonstrated in the language of application. Where fine-tuning is used, the same applies to the training data. For corpora spanning multiple languages, reporting performance separately by language guards against aggregate metrics masking weak performance in individual languages. Expert coders articulating and validating cues should likewise be proficient in the target languages, since the cues that carry a construct may be language- and culture-specific and not necessarily direct translations of their English counterparts.

References

- Ackley, D.H., Hinton, G.E., Sejnowski, T.J., 1985. A Learning Algorithm for Boltzmann Machines*. *Cognitive Science* 9, 147–169. https://doi.org/10.1207/s15516709cog0901_7
- Ahuja, K., Diddee, H., Hada, R., Ochieng, M., Ramesh, K., Jain, P., Nambi, A., Ganu, T., Segal, S., Ahmed, M., Bali, K., Sitaram, S., 2023. MEGA: Multilingual Evaluation of Generative AI, in: *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*. 4232–4267. <https://doi.org/10.18653/v1/2023.emnlp-main.258>
- Akiba, T., Sano, S., Yanase, T., Ohta, T., Koyama, M., 2019. Optuna: A Next-generation Hyperparameter Optimization Framework, in: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 2623–2631. <https://doi.org/10.1145/3292500.3330701>
- Bandarkar, L., Liang, D., Muller, B., Artetxe, M., Shukla, S.N., Husa, D., Goyal, N., Krishnan, A., Zettlemoyer, L., Khabsa, M., 2024. The Belebele Benchmark: a Parallel Reading Comprehension Dataset in 122 Language Variants, in: *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*. 749–775. <https://doi.org/10.18653/v1/2024.acl-long.44>
- Brown, T.B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., [...], Amodei, D., 2020. Language Models are Few-Shot Learners. <https://doi.org/10.48550/ARXIV.2005.14165>
- Carlson, N.A., Burbano, V., 2026. The use of LLMs to annotate data in management research: Foundational guidelines and warnings. *Strategic Management Journal* 47, 699–725. <https://doi.org/10.1002/smj.70023>
- Chen, T., Guestrin, C., 2016. XGBoost: A Scalable Tree Boosting System, in: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 785–794. <https://doi.org/10.1145/2939672.2939785>
- Cheng, X., Mayya, R., Sedoc, J., 2024. From Human Annotation to LLMs: SILICON Annotation Workflow for Management Research. <https://doi.org/10.48550/ARXIV.2412.14461>
- Chia, Y.K., Chen, G., Tuan, L.A., Poria, S., Bing, L., 2023. Contrastive Chain-of-Thought Prompting. <https://doi.org/10.48550/ARXIV.2311.09277>
- Clark, K., Luong, M.-T., Le, Q.V., Manning, C.D., 2020. ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators. <https://doi.org/10.48550/ARXIV.2003.10555>
- Comino, S., Manenti, F.M., Mariuzzo, F., 2019. Updates management in mobile applications: iTunes versus Google Play. *Journal of Economics & Management Strategy* 28, 392–419. <https://doi.org/10.1111/jems.12288>
- de Kok, T., 2025. ChatGPT for Textual Analysis? How to Use Generative LLMs in Accounting Research. *Management Science* 71, 7888–7906. <https://doi.org/10.1287/mnsc.2023.03253>
- Devlin, J., Chang, M.-W., Lee, K., Toutanova, K., 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, in: *Proceedings of the NAACL-HLT 2019. Association for Computational Linguistics*, 4171–4186. <https://doi.org/10.18653/v1/N19-1423>
- Durvasula, M.M., Eyuboglu, S., Ritzwoller, D.M., 2024. Counting Clinical Trials: New Evidence on Pharmaceutical Sector Productivity. <https://doi.org/10.48550/ARXIV.2405.08030>
- Einhorn, H.J., 1974. Expert judgment: Some necessary conditions and an example. *Journal of Applied Psychology* 59, 562–571. <https://doi.org/10.1037/h0037164>

- Foerderer, J., Kude, T., Mithas, S., Heinzl, A., 2018. Does Platform Owner's Entry Crowd Out Innovation? Evidence from Google Photos. *Information Systems Research* 29, 444–460. <https://doi.org/10.1287/isre.2018.0787>
- Franklin, J.M., Schneeweiss, S., Polinski, J.M., Rassen, J.A., 2014. Plasmode simulation for the evaluation of pharmacoepidemiologic methods in complex healthcare databases. *Computational Statistics & Data Analysis* 72, 219–226. <https://doi.org/10.1016/j.csda.2013.10.018>
- Gioia, D.A., Corley, K.G., Hamilton, A.L., 2013. Seeking Qualitative Rigor in Inductive Research: Notes on the Gioia Methodology. *Organizational Research Methods* 16, 15–31. <https://doi.org/10.1177/1094428112452151>
- Hannigan, T.R., Haans, R.F.J., Vakili, K., Tchalian, H., Glaser, V.L., Wang, M.S., Kaplan, S., Jennings, P.D., 2019. Topic Modeling in Management Research: Rendering New Theory from Textual Data. *Academy of Management Annals* 13, 586–632. <https://doi.org/10.5465/annals.2017.0099>
- Harris, Z.S., 1954. Distributional Structure. *Word* 10, 146–162. <https://doi.org/10.1080/00437956.1954.11659520>
- Johnson, J.M., Khoshgoftaar, T.M., 2019. Survey on deep learning with class imbalance. *Journal of Big Data* 6, 27. <https://doi.org/10.1186/s40537-019-0192-5>
- Just, J., Ströhle, T., Füller, J., Hutter, K., 2024. AI-based novelty detection in crowdsourced idea spaces. *Innovation: Organization & Management* 26, 359–386. <https://doi.org/10.1080/14479338.2023.2215740>
- Kim, Y., 2014. Convolutional Neural Networks for Sentence Classification, in: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 1746–1751. <https://doi.org/10.3115/v1/D14-1181>
- Kircher, T., Foerderer, J., 2024. Ban Targeted Advertising? An Empirical Investigation of the Consequences for App Development. *Management Science* 70, 1070–1092. <https://doi.org/10.1287/mnsc.2023.4726>
- Liu, P., Yuan, W., Fu, J., Jiang, Z., Hayashi, H., Neubig, G., 2023. Pre-train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing. *ACM Computer Surveys* 55, 1–35. <https://doi.org/10.1145/3560815>
- Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., Stoyanov, V., 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. <https://doi.org/10.48550/ARXIV.1907.11692>
- Manning, C.D., Raghavan, P., Schütze, H., 2008. *Introduction to Information Retrieval*, 1st ed. Cambridge University Press. <https://doi.org/10.1017/CBO9780511809071>
- Mikolov, T., Chen, K., Corrado, G., Dean, J., 2013. Efficient Estimation of Word Representations in Vector Space. <https://doi.org/10.48550/ARXIV.1301.3781>
- Miric, M., Jia, N., Huang, K.G., 2023. Using supervised machine learning for large-scale classification in management research: The case for identifying artificial intelligence patents. *Strategic Management Journal* 44, 491–519. <https://doi.org/10.1002/smj.3441>
- Mirzadeh, S.I., Farajtabar, M., Li, A., Levine, N., Matsukawa, A., Ghasemzadeh, H., 2019. Improved Knowledge Distillation via Teacher Assistant, in: *AAAI Conference on Artificial Intelligence* 34, 5191–5198. <https://doi.org/10.1609/aaai.v34i04.5963>
- Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Müller, A., Nothman, J., Louppe, G., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas,

- J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, É., 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 2825–2830.
- Peeperkorn, M., Kouwenhoven, T., Brown, D., Jordanous, A., 2024. Is Temperature the Creativity Parameter of Large Language Models? <https://doi.org/10.48550/ARXIV.2405.00492>
- Pennington, J., Socher, R., Manning, C., 2014. GloVe: Global Vectors for Word Representation, in: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 1532–1543. <https://doi.org/10.3115/v1/D14-1162>
- Petersen, A.M., 2022. Evolution of biomedical innovation quantified via billions of distinct article-level MeSH keyword combinations. *Advances in Complex Systems* 25, 2150016. <https://doi.org/10.1142/S0219525921500168>
- Sahoo, P., Singh, A.K., Saha, S., Jain, V., Mondal, S., Chadha, A., 2024. A Systematic Survey of Prompt Engineering in Large Language Models: Techniques and Applications. <https://doi.org/10.48550/ARXIV.2402.07927>
- Strauss, A.L., Corbin, J.M., 1998. *Basics of qualitative research: techniques and procedures for developing grounded theory*, 2nd ed. SAGE, Thousand Oaks, CA.
- Tietz, M., Fan, T.J., Nouri, D., Bossan, B., skorch Developers, 2017. skorch: A scikit-learn compatible neural network library that wraps PyTorch.
- Turney, P.D., Pantel, P., 2010. From Frequency to Meaning: Vector Space Models of Semantics. *Journal of Artificial Intelligence Research* 37, 141–188. <https://doi.org/10.1613/jair.2934>
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I., 2017. Attention Is All You Need, in: *Advances in Neural Information Processing Systems*. Curran Associates, Inc.
- Vaughan, L.K., Divers, J., Padilla, M.A., Redden, D.T., Tiwari, H.K., Pomp, D., Allison, D.B., 2009. The use of plasmodes as a supplement to simulations: A simple example evaluating individual admixture estimation methodologies. *Computational Statistics & Data Analysis* 53, 1755–1766. <https://doi.org/10.1016/j.csda.2008.02.032>
- Verhoeven, D., Bakker, J., Veugelers, R., 2016. Measuring technological novelty with patent-based indicators. *Research Policy* 45, 707–723. <https://doi.org/10.1016/j.respol.2015.11.010>
- Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E., Le, Q.V., Zhou, D., 2022. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. *Advances in Neural Information Processing Systems* 35, 24824–24837.
- Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Le Scao, T., Gugger, S., Drame, M., Lhoest, Q., Rush, A., 2020. Transformers: State-of-the-Art Natural Language Processing, in: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Association for Computational Linguistics, 38–45. <https://doi.org/10.18653/v1/2020.emnlp-demos.6>
- Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R., Le, Q.V., 2019. XLNet: Generalized Autoregressive Pretraining for Language Understanding, in: *Proceedings of the 33rd International Conference on Neural Information Processing Systems*. Curran Associates Inc., 5753–5763. <https://doi.org/10.5555/3454287.3454804>
- Zhang, Z., Zhang, A., Li, M., Smola, A., 2022. Automatic Chain of Thought Prompting in Large Language Models. <https://doi.org/10.48550/ARXIV.2210.03493>

Figure B1 Data structure for software application updates (Study 1)

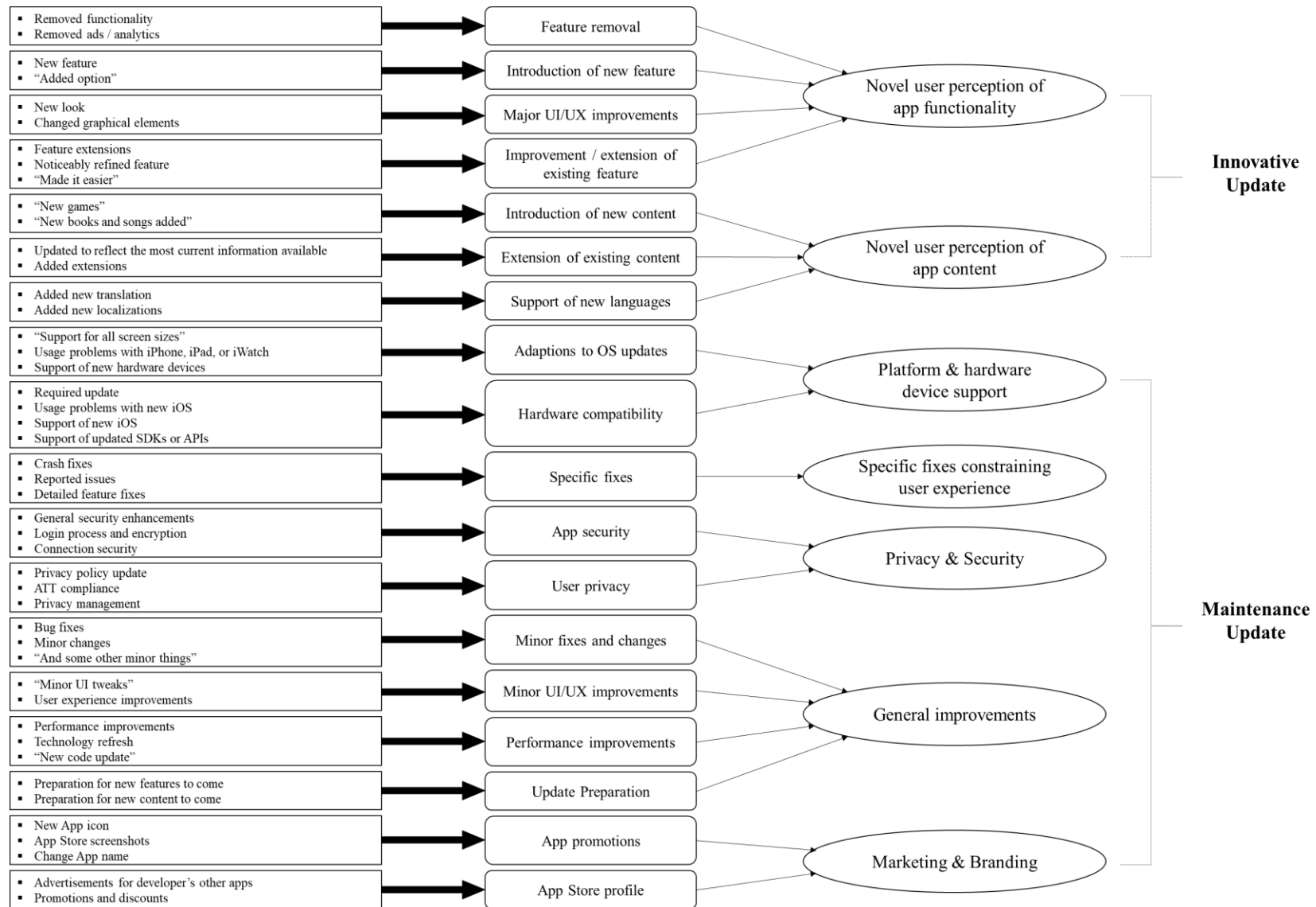


Figure B2 Data structure for product reviews (Study 2)

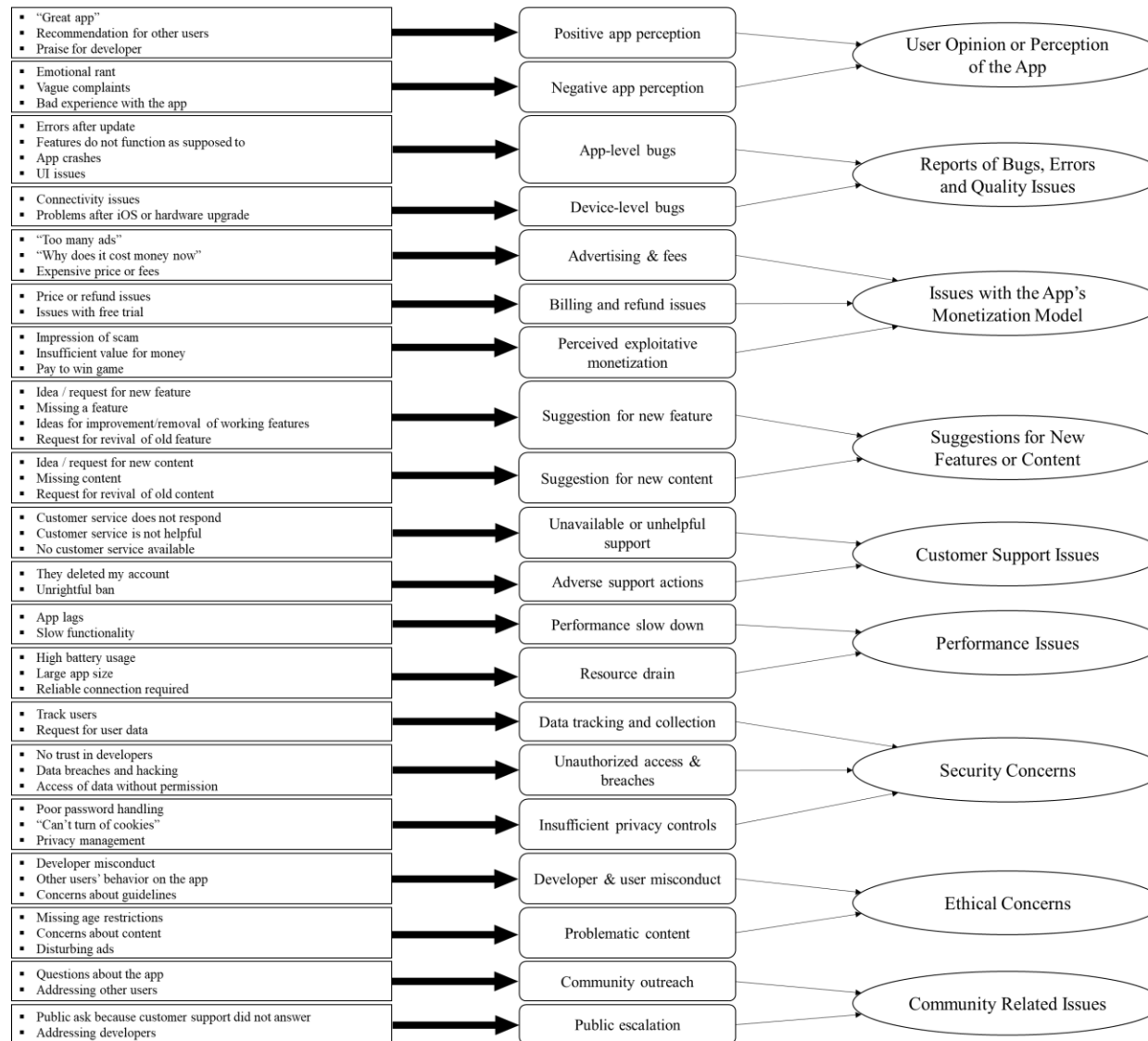
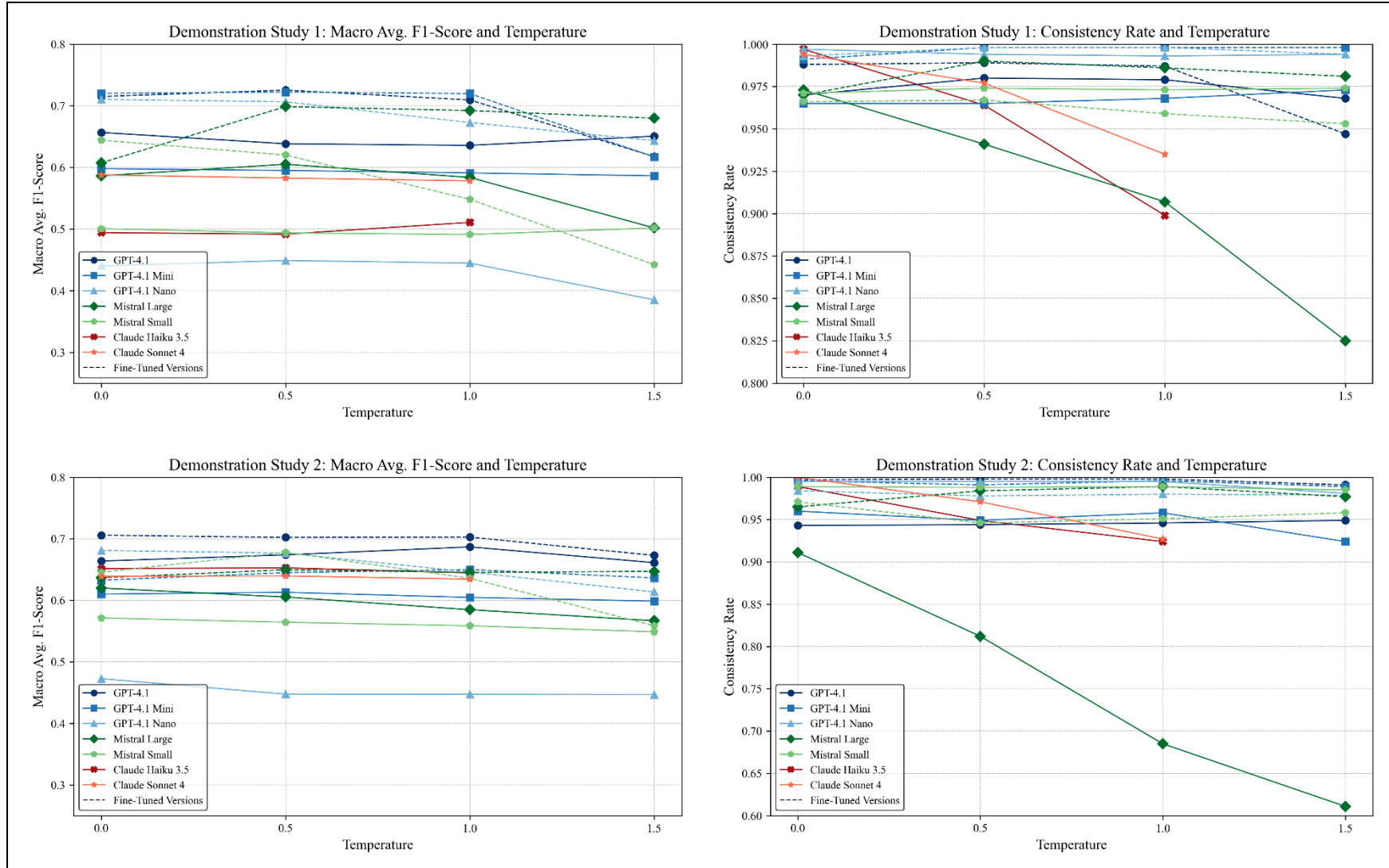
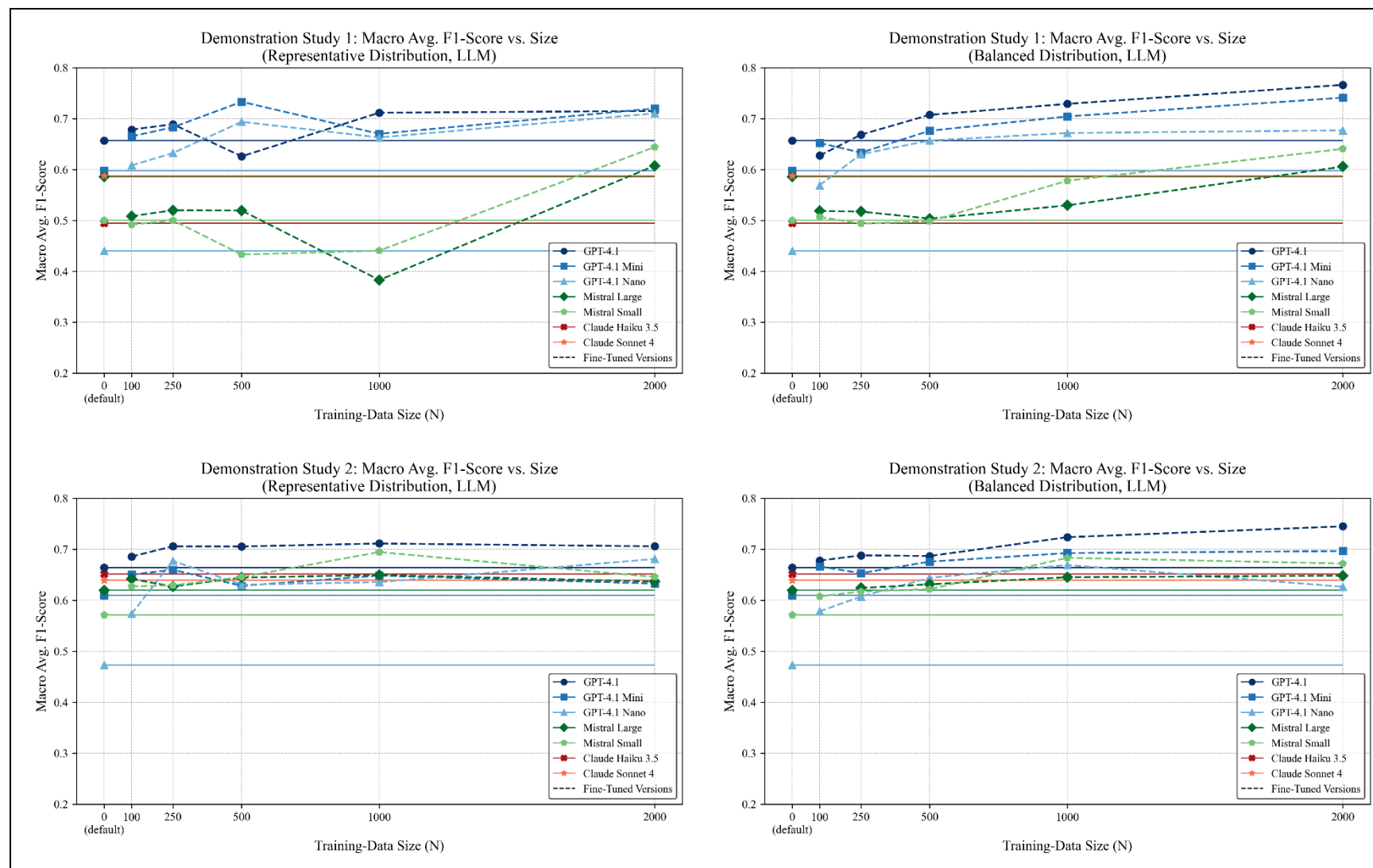


Figure E3 Model performance for classification with different temperatures



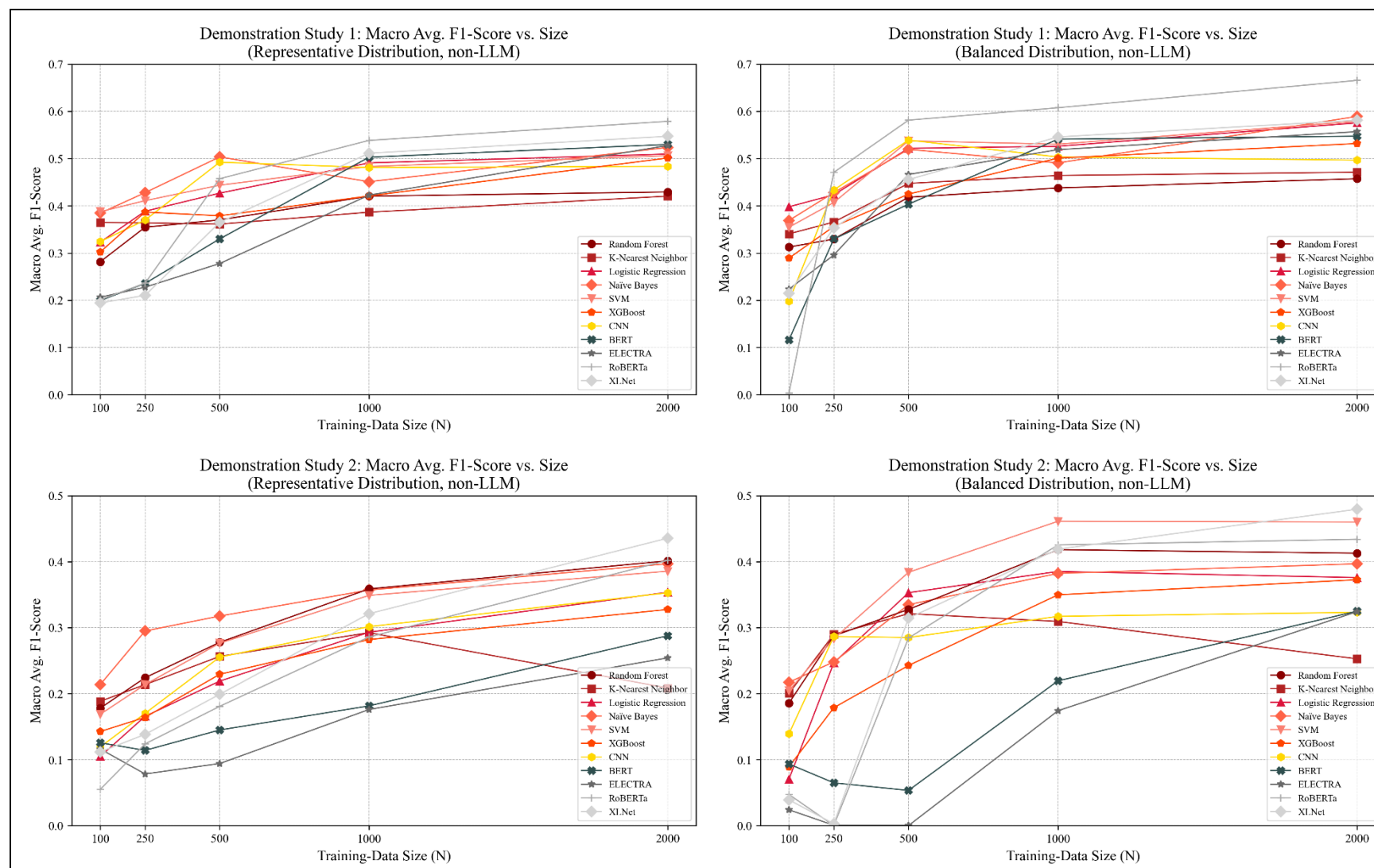
Note. LLMs were fine-tuned with a dataset $n = 2,000$, with categories distributed similarly to the validation data. F1-score is reported for the first run. Consistency Rate (CR) is based on three independent runs. Y-axes are scaled to fit best and worst model values. The maximum temperature for Claude models is limited to 1.0 per provider documentation.

Figure E4 Model performance for classification with different training data distribution and size



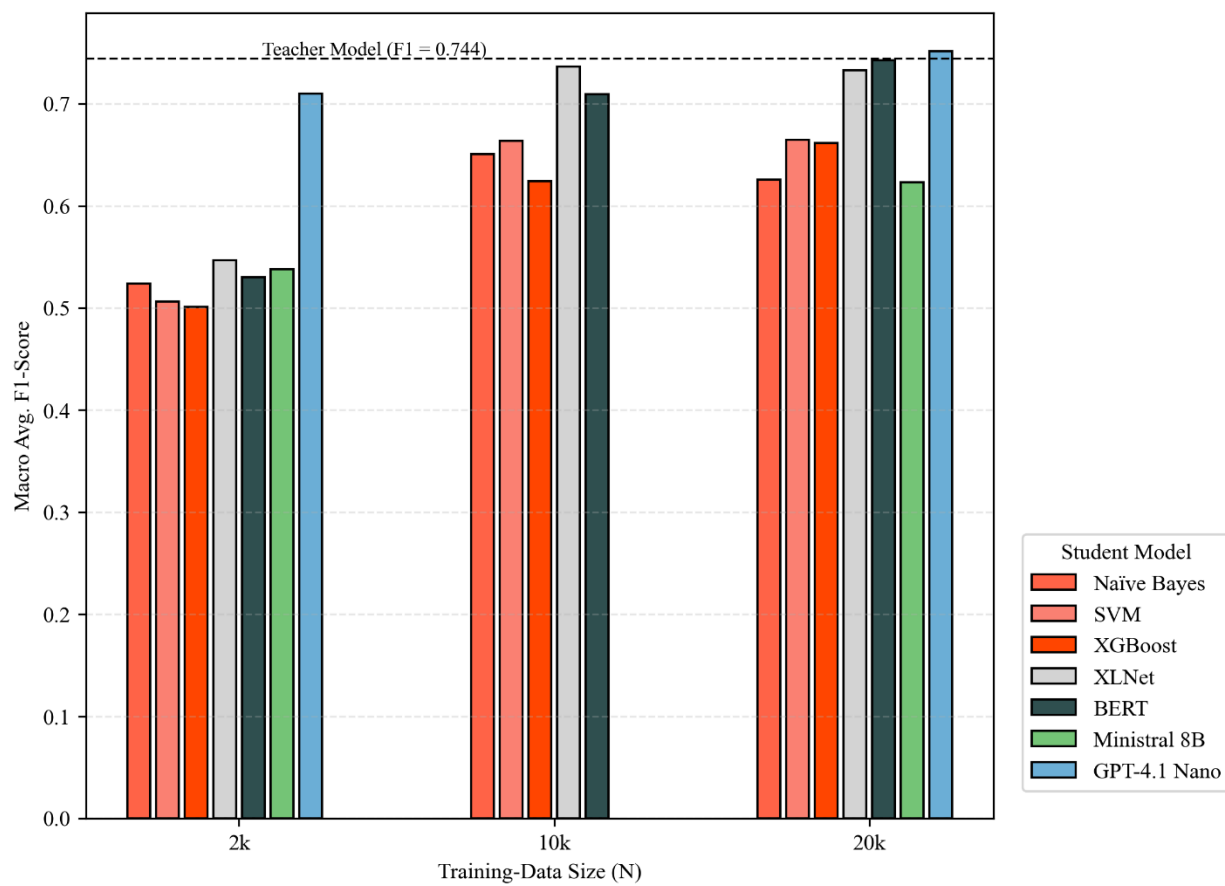
Note. F1-score is reported for the first run. Y-axes are scaled to fit best and worst model values.

Figure E5 Additional Analyses on the Role of Training Data Distribution and Data Size



Note. Embeddings for the ML classifiers were created using TF-IDF, and embeddings for the CNN were generated using glove-42b-300d. Y-axes are scaled to fit best and worst model values.

Figure F6 Model performance for teacher-student distillation setup



Note. Fine-Tuned GPT-4o Mini model is used as the teacher model. Sample sizes of $n = 10,000$ and $n = 20,000$ labels are labeled with the teacher model.

Table B1 Interviews with app developers

Interviewee	Duration
Professional Mobile App Developer (iOS & Android), Computer Scientist (M.Sc.)	31:13 min
Professional Mobile App Developer (iOS & Android), Computer Scientist (B.Sc.)	81:26 min
Professional iOS Developer, Data Scientist (M.A.)	Written survey
Professional Mobile & Web Developer, Software Engineer (Bachelor's Degree)	24:14 min
Professional Mobile App Developer (iOS & Android), Computer Scientist (B.Sc.)	18:17 min
Professional Unity Game Developer and Designer (for Mobile Games), Software Engineer (Bachelor's Degree)	26:32 min
Professional Unity Game Developer (for Mobile Games)	37:49 min

Table D2 Hyperparameter configurations and performance evaluation for ML classifiers for Demonstration Studies 1 & 2

(1)		(2)					
Best Performing Hyperparameter Configuration		Best CV Macro F1	HO Macro F1	Best Performing Hyperparameter Configuration		Best CV Macro F1	HO Macro F1
TF-IDF	K-Nearest-Neighbor	{“leaf_size”: 38, “n_neighbors”: 8, “weights”: “distance”, “max_df”: 0.9240475867068707, “max_features”: 8335, “min_df”: 1, “ngram_range”: [1, 3]}	.486	.420	{“leaf_size”: 37, “n_neighbors”: 18, “weights”: “uniform”, “max_df”: 0.9882818784859139, “max_features”: 7518, “min_df”: 1, “ngram_range”: [1, 1]}	.198	.208
	Logistic Regression	{“C”: 6.828621268586129, “penalty”: “l2”, “solver”: “lbfgs”, “max_df”: 0.996760773680571, “max_features”: 9801, “min_df”: 3, “ngram_range”: [1, 1]}	.543	.510	{“C”: 8.839465761662865, “penalty”: “l2”, “solver”: “saga”, “max_df”: 0.9958605089165147, “max_features”: 4093, “min_df”: 4, “ngram_range”: [1, 1]}	.302	.354
	Naïve Bayes	{“alpha”: 0.11917071876314952, “fit_prior”: false, “max_df”: 0.8326423978632586, “max_features”: 9392, “min_df”: 3, “ngram_range”: [1, 2]}	.543	.524	{“alpha”: 0.11917071876314952, “fit_prior”: false, “max_df”: 0.8326423978632586, “max_features”: 9392, “min_df”: 3, “ngram_range”: [1, 2]}	.380	.397
	Random Forest	{“bootstrap”: false, “criterion”: “entropy”, “max_depth”: 96, “max_features”: 0.6779687087336601, “min_samples_leaf”: 3, “min_samples_split”: 11, “n_estimators”: 448, “max_df”: 0.8260307570294546, “max_features”: 2630, “min_df”: 1, “ngram_range”: [1, 2]}	.477	.429	{“bootstrap”: false, “criterion”: “gini”, “max_depth”: 95, “max_features”: 0.2077430655332096, “min_samples_leaf”: 1, “min_samples_split”: 8, “n_estimators”: 198, “max_df”: 0.6627743230167918, “max_features”: 5848, “min_df”: 1, “ngram_range”: [1, 3]}	.344	.401
	SVM	{“C”: 8.73445728344814, “degree”: 2, “gamma”: “auto”, “kernel”: “linear”, “max_df”: 0.685093473488098, “max_features”: 5666, “min_df”: 1, “ngram_range”: [1, 2]}	.590	.506	{“C”: 6.918621268586129, “degree”: 2, “gamma”: “scale”, “kernel”: “linear”, “max_df”: 0.6041717425276485, “max_features”: 5270, “min_df”: 4, “ngram_range”: [1, 1]}	.393	.386
Glove	XGBoost	{“colsample_bytree”: 0.6965180821364413, “learning_rate”: 0.07583861356997272, “max_depth”: 3, “n_estimators”: 445, “subsample”: 0.6734468864139178, “max_df”: 0.6783234961893887, “max_features”: 8100, “min_df”: 3, “ngram_range”: [1, 3]}	.512	.501	{“colsample_bytree”: 0.767077736372738, “learning_rate”: 0.19276041730313465, “max_depth”: 3, “n_estimators”: 164, “subsample”: 0.7816961974601263, “max_df”: 0.6913515729152793, “max_features”: 3109, “min_df”: 4, “ngram_range”: [1, 2]}	.294	.328
	K-Nearest-Neighbor	{“vectorizer”: glove-840B-300d, “leaf_size”: 24, “n_neighbors”: 5, “weights”: “distance”}	.426	.406	{“vectorizer”: glove-42B-300d, “leaf_size”: 33, “n_neighbors”: 4, “weights”: “distance”}	.282	.337
	Logistic Regression	{“vectorizer”: glove-42B-300d, “C”: 6.999135822979566, “penalty”: “l2”, “solver”: “lbfgs”}	.572	.513	{“vectorizer”: glove-840B-300d, “C”: 9.76042118194638, “penalty”: “l2”, “solver”: “lbfgs”}	.340	.335
	Random Forest	{“vectorizer”: glove-42B-300d, “bootstrap”: false, “criterion”: “gini”, “max_depth”: 95, “max_features”: 0.2077430655332096, “min_samples_leaf”: 1, “min_samples_split”: 8, “n_estimators”: 198}	.423	.424	{“vectorizer”: glove-42B-300d, “bootstrap”: false, “criterion”: “entropy”, “max_depth”: 80, “max_features”: 0.9425728949509299, “min_samples_leaf”: 8, “min_samples_split”: 17, “n_estimators”: 326}	.263	.231
	SVM	{“vectorizer”: glove-42B-300d, “C”: 5.121773443923852, “degree”: 4, “gamma”: “auto”, “kernel”: “linear”}	.556	.500	{“vectorizer”: glove-840B-300d, “C”: 9.291710982638396, “degree”: 5, “gamma”: “auto”, “kernel”: “linear”}	.352	.358
N	XGBoost	{“vectorizer”: glove-840B-300d, “colsample_bytree”: 0.557957280470468, “learning_rate”: 0.23112312784449587, “max_depth”: 4, “n_estimators”: 455, “subsample”: 0.6505946735495828}	.537	.503	{“vectorizer”: glove-42B-300d, “colsample_bytree”: 0.594405164158504, “learning_rate”: 0.2845324775722167, “max_depth”: 6, “n_estimators”: 115, “subsample”: 0.6025027592743828}	.300	.283
		667	1,000		711	1,047	

Note. ML-based classifiers were trained with a dataset $n = 2,000$ with a representative label distribution. Observation count (N) of macro averaged F1-score of cross-validated samples can vary depending on the data splits. On average, they are about one-third of the training data size.

Table D3 Hyperparameter configurations and performance evaluation for convolutional neural network for Demonstration Studies 1 & 2

(1)			(2)			
	Best Performing Hyperparameter Configuration	Best CV Macro F1	HO Macro F1	Best Performing Hyperparameter Configuration	Best CV Macro F1	HO Macro F1
Convolutional Neural Network	{“batch_size”: 60, “max_epochs”: 12, “dense_units”: 72, “dropout”: 0.46777990469323594, “kernel_size”: 6, “n_filters”: 84, “trainable”: true, “num_words”: 16763, “seq_len”: 214}	.464	.484	{“batch_size”: 58, “max_epochs”: 10, “dense_units”: 72, “dropout”: 0.40989527282572635, “kernel_size”: 5, “n_filters”: 228, “trainable”: true, “num_words”: 26815, “seq_len”: 195}	.280	.353
N		667	1,000		711	1,047

Note. CNN was trained with a dataset $n = 2,000$ with a representative label distribution. Observation count (N) of macro averaged F1-score of cross-validated samples can vary depending on the data splits. On average, they are one-third of the training data size.

Table D4 Hyperparameter configurations and performance evaluation for pre-trained language models for Demonstration Studies 1 & 2

(1)			(2)	
	Best Performing Hyperparameter Configuration	HO Macro F1	Best Performing Hyperparameter Configuration	HO Macro F1
BERT	{"learning_rate": 2.3139649033257978e-05, "batch_size": 8, "num_train_epochs": 3, "weight_decay": 0.0072173948625224625}	.530	{"learning_rate": 1.3205102630174912e-05, "batch_size": 8, "num_train_epochs": 5, "weight_decay": 0.0069800190183135335}	.288
ELECTRA	{"learning_rate": 1.2868586517068187e-05, "batch_size": 8, "num_train_epochs": 5, "weight_decay": 0.004205974315203549}	.528	{"learning_rate": 2.3139649033257978e-05, "batch_size": 8, "num_train_epochs": 3, "weight_decay": 0.0072173948625224625}	.254
RoBERTa	{"learning_rate": 1.2868586517068187e-05, "batch_size": 8, "num_train_epochs": 5, "weight_decay": 0.004205974315203549}	.579	{"learning_rate": 1.2868586517068187e-05, "batch_size": 8, "num_train_epochs": 5, "weight_decay": 0.004205974315203549}	.402
XLNet	{"learning_rate": 1.2868586517068187e-05, "batch_size": 8, "num_train_epochs": 5, "weight_decay": 0.004205974315203549}	.547	{"learning_rate": 1.3205102630174912e-05, "batch_size": 8, "num_train_epochs": 5, "weight_decay": 0.0069800190183135335}	.436
N		1,000		1,047

Note. PLMs were fine-tuned with a dataset $n = 2,000$ with a representative label distribution.

Table E5 Model performance for classification with different model provider, size, age, and reasoning capability

		(1)			(2)		
		Macro F1	Weighted F1	CR	Macro F1	Weighted F1	CR
Claude Opus 3	Anthropic	.569	.740	.991	.596	.779	.982
Claude Opus 4 (reasoning)		.511	.787	.852	.650	.823	.996
Claude Sonnet 3.5		.644	.800	.994	.649	.814	.991
Claude Sonnet 3.7		.613	.771	.996	.638	.815	.987
Claude Haiku 3		.481	.563	1.000	.533	.732	1.000
Ministral 3B	Mistral	.399	.508	.970	.483	.703	.945
Ministral 8B		.287	.378	.978	.402	.674	.983
Open Mistral Nemo		.277	.245	.880	.491	.695	.862
Mistral Medium		.594	.690	.964	.622	.806	.941
Ministral 8B (fine-tuned)		.538	.756	.989	.571	.793	.994
Open Mistral Nemo (fine-tuned)		.485	.714	.945	.556	.788	.947
o3 (reasoning)	OpenAI	.684	.809	.932	.693	.845	.935
o4 Mini (reasoning)		.580	.734	.998	.693	.855	.978
GPT-3.5 Turbo		.439	.511	.916	.564	.780	.961
GPT-4o Mini		.484	.571	.976	.576	.797	.934
GPT-4o		.602	.768	.956	.640	.813	.948
GPT-3.5 Turbo (fine-tuned)		.751	.884	.997	.677	.854	1.000
GPT-4o Mini (fine-tuned)		.744	.881	.993	.662	.858	.993
GPT-4o (fine-tuned)		.741	.889	.990	.678	.861	.993
N		1,000	1,000	3,000	1,047	1,047	3,000

Note. Transformer-based models were fine-tuned with a dataset $n = 2,000$, with categories distributed similarly to the validation data. Claude models, Ministral 3B, Mistral Medium, o3, and o4 Mini are not open for supervised fine-tuning. LLM temperature = 0.0, LLM seed = 94032.

Table E6 Variations in prompt engineering for Demonstration Study 1

Prompt Engineering Technique	Prompt Text
Few-Shot	<p>ONLY provide the category number (1-7) in response. Determine the category for the following app update text. If multiple categories seem applicable, always choose the lowest category number (1<2<3<4<5<6<7):</p> <ol style="list-style-type: none"> (1) Novel features - Introducing or enhancing significant functionalities that modify user experience (2) Content extensions – Adding or expanding curated or user-facing content (3) Platform and device support – Enabling compatibility with new OS versions, development kits, device types, or hardware features (4) Specific fixes of bugs - Addressing distinct known issues (5) Privacy & Security - Strengthening user data protection, permissions, or security protocols (6) General Improvements – Tweaks addressing undisclosed bug fixes, performance improvements, minor changes, or UI/UX polish (7) Marketing & Branding – Changes purely about promotional or visual branding assets <p>EXAMPLES: Updates: ‘new style’, ‘new features’, ‘new interface’ Response: 1</p> <p>Updates: ‘optimized performance’, ‘new levels’, ‘new stage added’ Response: 2</p> <p>Updates: ‘updated for iOS 9’, ‘resolves iOS 9 issues’, ‘adapt to iOS 12 system’ Response: 3</p> <p>Updates: ‘fix for push-messaging’, ‘fixed game freeze bug’, ‘fixed a crash that could occur when starting the game’ Response: 4</p> <p>Updates: ‘security updates’, ‘terms of use updated’, ‘added prompt to ask permission for App Tracking Transparency’ Response: 5</p> <p>Updates: ‘bug fixes’, ‘stability improvements’, ‘improved feeds’ Response: 6</p> <p>Updates: ‘new app icon introduced’, ‘screenshots for App Store added’, ‘Spring has sprung and it's time to get floral! Collect as many blossoms as you can; the more you collect, the bigger the reward!’ Response: 7</p>
Automatic Chain-of-Thought	<p>ONLY provide the category number (1-7) in response. Determine the category for the following app update text. If multiple categories seem applicable, always choose the lowest category number (1<2<3<4<5<6<7):</p> <ol style="list-style-type: none"> (1) Novel features - Introducing or enhancing significant functionalities that modify user experience (2) Content extensions – Adding or expanding curated or user-facing content (3) Platform and device support – Enabling compatibility with new OS versions, development kits, device types, or hardware features (4) Specific fixes of bugs - Addressing distinct known issues (5) Privacy & Security – Strengthening user data protection, permissions, or security protocols (6) General Improvements – Tweaks addressing undisclosed bug fixes, performance improvements, minor changes, or UI/UX polish (7) Marketing & Branding – Changes purely about promotional or visual branding assets <p>Let’s think step by step.</p>
Manual Chain-of-Thought	<p>ONLY provide the category number (1-7) in response. Determine the category for the following app update text. If multiple categories seem applicable, always choose the lowest category number (1<2<3<4<5<6<7):</p> <ol style="list-style-type: none"> (1) Novel features - Introducing or enhancing significant functionalities that modify user experience (2) Content extensions – Adding or expanding curated or user-facing content (3) Platform and device support – Enabling compatibility with new OS versions, development kits, device types, or hardware features (4) Specific fixes of bugs – Addressing distinct known issues (5) Privacy & Security - Strengthening user data protection, permissions, or security protocols (6) General Improvements – Tweaks addressing undisclosed bug fixes, performance improvements, minor changes, or UI/UX polish

(7) Marketing & Branding – Changes purely about promotional or visual branding assets

Let's think step by step.

ANALYZE STEP BY STEP:

1. Read the app update text carefully
2. Identify the main purpose of the update
3. Check the different categories starting from 1 to 7 and determine which category applies best
4. If multiple categories apply, choose the category with the lowest number
5. Provide the number of the category as response

EXAMPLE:

Update text: 'game runs faster and more levels have been added'

1. Reading of update text 'game runs faster and more levels have been added'
2. Main purpose of 'game runs faster' is performance improvement and purpose of 'more levels have been added' is extensions of app content
3. Performance improvement aligns with category 6, extensions of app content aligns with category 2
4. Lower category number 2 is chosen
5. 2 is provided as response

ONLY provide the category number (1-7) in response. Determine the category for the following app update text. If multiple categories seem applicable, always choose the lowest category number (1<2<3<4<5<6<7):

- (1) Novel features - Introducing or enhancing significant functionalities that modify user experience
- (2) Content extensions – Adding or expanding curated or user-facing content
- (3) Platform and device support – Enabling compatibility with new OS versions, development kits, device types, or hardware features
- (4) Specific fixes of bugs – Addressing distinct known issues
- (5) Privacy & Security - Strengthening user data protection, permissions, or security protocols
- (6) General Improvements – Tweaks addressing undisclosed bug fixes, performance improvements, minor changes, or UI/UX polish
- (7) Marketing & Branding – Changes purely about promotional or visual branding assets

Let's think step by step.

ANALYZE STEP BY STEP:

1. Read the app update text carefully
2. Identify the main purpose of the update
3. Check the different categories starting from 1 to 7 and determine which category applies best
4. If multiple categories apply, choose the category with the lowest number
5. Provide the number of the category as response

Contrastive
Chain-of-
Thought

CORRECT EXAMPLE:

Update text: 'game runs faster and more levels have been added'

1. Reading of update text 'game runs faster and more levels have been added'
2. Main purpose of 'game runs faster' is performance improvement and purpose of 'more levels have been added' is extensions of app content
3. Performance improvement aligns with category 6, extensions of app content aligns with category 2
4. Lower category number 2 is chosen
5. 2 is provided as response"

WRONG EXAMPLE:

Update text: 'game runs faster and more levels have been added'

1. Reading of update text 'game runs faster and more levels have been added'
 2. Main purpose of 'game runs faster' is extensions/additions of app content and purpose of 'more levels have been added' is privacy & security
 3. Extensions/additions of app content aligns with category 6, privacy and security aligns with category 2
 4. Lower category number 2 is chosen
 5. 1 is provided as response
-

Table E7 Variations in prompt engineering for Demonstration Study 2

Prompt Engineering Technique	Prompt Text
Few-Shot	<p>ONLY provide a number (0-8) in response. Categorize the following app review text by assigning the most fitting category/categories out of the following nine categories. If the text contains elements from multiple categories, provide the categories separated by ; .</p> <ul style="list-style-type: none"> (0) User Opinion without specific reports, issues, suggestions - i.e., review only about good/bad perception of the app but nothing else (1) Reports of bugs, errors, or bad quality issues - i.e., something does not work in the app or is of bad quality (2) Issues of the app's monetization model - i.e., complaints or issues of how the app monetizes content (3) Suggestions for new features or content or revival of removed features (4) Customer support issues - i.e., problems or complaints regarding customer support (5) Performance issues - i.e., the app needs too much space, is too slow or similar (6) Security concerns - i.e., user is concerned about their data or privacy (7) Ethical concerns - i.e., user is concerned about practices in the app, fairness, discrimination (8) Community related issues - user asks openly for help (no feedback at customer support) <p>EXAMPLES:</p> <p>Review texts: 'I love this game', 'clear, friendly, effective', 'horrible app; if I could give these creators 0 stars, I would'</p> <p>Response: 0</p> <p>Review texts: 'I'm getting error messages when trying to use the app', 'makes my phone incredibly hot, then the game crashes', 'I am unable to access the app'</p> <p>Response: 1</p> <p>Review texts: 'the items should be at a lower price', 'the apps are unbelievably disturbing', 'the app hides the ability to cancel the subscription'</p> <p>Response: 2</p> <p>Review texts: 'add home screen widgets', 'please support on iPad', 'create stories'</p> <p>Response: 3</p> <p>Review texts: 'there is no solution even though I reported the issue already', 'nobody replies from support', 'I cannot contact the support'</p> <p>Response: 4</p> <p>Review texts: 'the app drains 50% battery overnight', 'updating the app takes too long', 'the game lagged a bit'</p> <p>Response: 5</p> <p>Review texts: 'do not trust anything about this app, it's a scam', 'they are selling our privacy', 'app reads and saves photos'</p> <p>Response: 6</p> <p>Review texts: 'the community tends to be very toxic towards each other', 'the app is debased by its selective censorship', 'this app clearly promotes sex trafficking'</p> <p>Response: 7</p> <p>Review texts: 'how can I change the language', 'hey guys, I have no idea what the color paints are for', 'please unfreeze my account and support me'</p> <p>Response: 8</p>
Automatic Chain-of-Thought	<p>Only provide a number (0-8) as response. Categorize the following app review text based on the most applicable out of the following 9 categories. If multiple categories apply, provide the categories separated by ; .</p> <ul style="list-style-type: none"> (0) User Opinion without specific reports, issues, suggestions - i.e., review only about good/bad perception of the app but nothing else (1) Reports of bugs, errors, or bad quality issues - i.e., something does not work in the app or is of bad quality (2) Issues of the app's monetization model - i.e., complaints or issues of how the app monetizes content (3) Suggestions for new features or content or revival of removed features (4) Customer support issues - i.e., problems or complaints regarding customer support (5) Performance issues - i.e., the app needs too much space, is too slow or similar (6) Security concerns - i.e., user is concerned about their data or privacy (7) Ethical concerns - i.e., user is concerned about practices in the app, fairness, discrimination (8) Community related issues - user asks openly for help (no feedback at customer support) <p>Let's think step by step.</p>
Manual Chain-of-Thought	<p>Only provide a number (0-8) as response. Categorize the following app review text based on the most applicable out of the following 9 categories. If multiple categories apply, provide the categories separated by ; .</p> <ul style="list-style-type: none"> (0) User Opinion without specific reports, issues, suggestions - i.e., review only about good/bad perception of the app but nothing else (1) Reports of bugs, errors, or bad quality issues - i.e., something does not work in the app or is of bad quality (2) Issues of the app's monetization model - i.e., complaints or issues of how the app monetizes content (3) Suggestions for new features or content or revival of removed features (4) Customer support issues - i.e., problems or complaints regarding customer support

	<p>(5) Performance issues - i.e., the app needs too much space, is too slow or similar</p> <p>(6) Security concerns - i.e., user is concerned about their data or privacy</p> <p>(7) Ethical concerns - i.e., user is concerned about practices in the app, fairness, discrimination</p> <p>(8) Community related issues - user asks openly for help (no feedback at customer support)</p> <p>ANALYZE STEP BY STEP:</p> <ol style="list-style-type: none"> 1. Read the app review text carefully 2. Identify the main purpose of the review 3. Check the different categories starting from 0 to 8 and determine which category/categories applies/apply best 4. Provide the number(s) of the category/categories as response <p>EXAMPLE:</p> <p>Review text: 'Trying to get an answer about anonymity. Can a winner stay anonymous? They have my name and address. Emailed for an answer- no response. Reading of update text 'game runs faster and more levels have been added'</p> <ol style="list-style-type: none"> 1. Reading of update text 'Trying to get an answer about anonymity. Can a winner stay anonymous? They have my name and address. Emailed for an answer- no response.' 2. Main purpose of 'trying to get an answer' and 'Emailed for an answer - no response' is customer support issue and purpose of 'anonymity' and 'They have my name and address' is security concerns 3. Customer support issue aligns with category 4, security concerns aligns with category 6 4. 4;6 are provided as response
Contrastive Chain-of-Thought	<p>Only provide a number (0-8) as response. Categorize the following app review text based on the most applicable out of the following 9 categories. If multiple categories apply, provide the categories separated by ; .</p> <ol style="list-style-type: none"> (0) User Opinion without specific reports, issues, suggestions - i.e., review only about good/bad perception of the app but nothing else (1) Reports of bugs, errors, or bad quality issues - i.e., something does not work in the app or is of bad quality (2) Issues of the app's monetization model - i.e., complaints or issues of how the app monetizes content (3) Suggestions for new features or content or revival of removed features (4) Customer support issues - i.e., problems or complaints regarding customer support (5) Performance issues - i.e., the app needs too much space, is too slow or similar (6) Security concerns - i.e., user is concerned about their data or privacy (7) Ethical concerns - i.e., user is concerned about practices in the app, fairness, discrimination (8) Community related issues - user asks openly for help (no feedback at customer support) <p>ANALYZE STEP BY STEP:</p> <ol style="list-style-type: none"> 1. Read the app review text carefully 2. Identify the main purpose of the review 3. Check the different categories starting from 0 to 8 and determine which category/categories applies/apply best 4. Provide the number(s) of the category/categories as response <p>CORRECT EXAMPLE:</p> <p>Review text: 'Trying to get an answer about anonymity. Can a winner stay anonymous? They have my name and address. Emailed for an answer- no response. Reading of update text 'game runs faster and more levels have been added'</p> <ol style="list-style-type: none"> 1. Reading of update text 'Trying to get an answer about anonymity. Can a winner stay anonymous? They have my name and address. Emailed for an answer- no response.' 2. Main purpose of 'trying to get an answer' and 'Emailed for an answer - no response' is customer support issue and purpose of 'anonymity' and 'They have my name and address' is security concerns 3. Customer support issue aligns with category 4, security concerns aligns with category 6 4. 4;6 are provided as response <p>WRONG EXAMPLE:</p> <p>Review text: 'Trying to get an answer about anonymity. Can a winner stay anonymous? They have my name and address. Emailed for an answer- no response. Reading of update text 'game runs faster and more levels have been added'</p> <ol style="list-style-type: none"> 1. Reading of update text 'Trying to get an answer about anonymity. Can a winner stay anonymous? They have my name and address. Emailed for an answer- no response.' 2. Main purpose of 'trying to get an answer' and 'Emailed for an answer - no response' is security concerns and purpose of 'anonymity' and 'They have my name and address' is ethical concerns 3. Security concerns aligns with category 4, ethical concerns aligns with category 6 4. 1;7 are provided as response

Table E8 Model performance for classification with different prompt designs

		(1)			(2)		
		Macro F1	Weighted F1	CR	Macro F1	Weighted F1	CR
GPT-4.1 Nano	Few-Shot	.445	.535	.997	.586	.784	.986
GPT-4.1 Mini		.621	.783	.954	.610	.813	.927
GPT-4.1		.658	.822	.975	.662	.831	.952
GPT-4.1 Nano (fine-tuned)		.650	.856	.995	.660	.842	.987
GPT-4.1 Mini (fine-tuned)		.724	.866	.999	.663	.856	.988
GPT-4.1 (fine-tuned)		.739	.881	.987	.695	.866	.985
GPT-4.1 Nano	Chain-of-Thought (Automatic)	.428	.485	.996	.482	.753	.991
GPT-4.1 Mini		.596	.751	.962	.621	.806	.946
GPT-4.1		.647	.821	.977	.672	.837	.961
GPT-4.1 Nano (fine-tuned)		.593	.858	.993	.653	.840	.984
GPT-4.1 Mini (fine-tuned)		.721	.869	.999	.645	.843	.992
GPT-4.1 (fine-tuned)		.710	.874	.991	.702	.860	.991
GPT-4.1 Nano	Chain-of-Thought (Manual)	.411	.415	.997	.470	.722	.980
GPT-4.1 Mini		.577	.723	.957	.621	.798	.958
GPT-4.1		.631	.808	.976	.674	.831	.926
GPT-4.1 Nano (fine-tuned)		.554	.841	.994	.662	.834	.982
GPT-4.1 Mini (fine-tuned)		.719	.872	.995	.647	.850	.993
GPT-4.1 (fine-tuned)		.722	.876	.983	.691	.863	.992
GPT-4.1 Nano	Chain-of-Thought (Contrastive)	.341	.355	.992	.468	.725	.978
GPT-4.1 Mini		.603	.737	.947	.616	.802	.956
GPT-4.1		.671	.823	.985	.684	.831	.940
GPT-4.1 Nano (fine-tuned)		.649	.840	.993	.676	.841	.982
GPT-4.1 Mini (fine-tuned)		.729	.873	.996	.645	.854	.994
GPT-4.1 (fine-tuned)		.748	.882	.987	.710	.867	.990
N		1,000	1,000	3,000	1,047	1,047	3,000

Note. Transformer-based models were fine-tuned with a dataset $n = 2,000$, with categories distributed similarly to the validation data. LLM temperature = 0.0, LLM seed = 94032. See Appendix E2 for prompts.

Table G9 When to fine-tune an LLM for framework application

Task or deployment characteristic	Ex ante question	Recommendation	Exemplary Demonstration Study Result
Classification with common, balanced classes	Does the default configuration already meet the required performance on the holdout sample?	Default suffices; fine-tuning yields marginal gains that are unlikely to alter substantive conclusions	Study 1, Table 2 (GPT-4.1 default F1 = .901 vs. .926 fine-tuned)
Limited expert-validated training data	Is the available training volume below the model-specific threshold at which fine-tuning pays off?	Do not fine-tune; below this volume, fine-tuned models can underperform their own default configurations	Appendix E4, Figure E4 (GPT-4.1 below its default at n = 500, Mistral models until n = 2,000)
Default configuration misses or under-recalls construct-relevant instances	Does the default configuration miss rare classes or under-recall relevant instances on the holdout sample?	Fine-tune; raises recall and per-class performance, verified per class on the holdout sample	Study 1, Table 2 (GPT-4.1 recall .879 to .961); Study 2, Table 3 (e.g., GPT-4.1 class 4 F1 .727 to .807)
Local, scalable, or API-independent deployment	Must inference run locally or at low recurring cost?	Fine-tune a student model on teacher-generated labels	Online Appendix F (teacher–student distillation)

Table H10 Downstream estimates with innovation-related updates as regressor (case i)

Configuration	F1	Estimated β	95% simulation interval	$\beta^{\text{FN only}}$	$\beta^{\text{FP only}}$
GPT-4.1 default	.901	.416	[.341, .483]	.457	.468
GPT-4.1 fine-tuned	.926	.430	[.359, .502]	.484	.449
Classical ML (best)	.864	.373	[.298, .443]	.470	.415
PLM (BERT)	.887	.395	[.317, .467]	.474	.429

Note. Estimated coefficients of innovation-related updates on the synthetic outcome ($\beta = .50$). The simulation interval spans the 2.5th to 97.5th percentile of the coefficient distribution across 1,000 simulated outcome realizations. F1 refers to the binary innovation indicator on the holdout sample. $\beta^{\text{FN only}}$ retains a configuration's false negatives and corrects its false positives; $\beta^{\text{FP only}}$ retains its false positives and corrects its false negatives.

Table H11 Measured pre-post change with innovation-related updates as outcome (case ii)

Configuration	Measured Δ (pp)	Distortion (pp)	95% paired bootstrap CI	Baseline error (pp)	FPR drift (pp)	FNR drift (pp)
GPT-4.1 default	-1.58	-1.52	[-5.20, 2.18]	.01	-.05	-1.48
GPT-4.1 fine-tuned	-2.72	-2.66	[-5.94, .49]	.01	-1.65	-1.02
Classical ML (best)	1.13	1.20	[-3.37, 5.47]	.02	.07	1.11
PLM (BERT)	-.14	-.08	[-4.27, 4.06]	.01	-.13	.04

Note. Change in the measured share of innovation-related updates between the earlier and later holdout period, in percentage points (pp). Distortion is the measured change minus the expert-labeled change of $-.06$ pp. The confidence interval spans the 2.5th to 97.5th percentile of the distortion across 2,000 paired bootstrap resamples stratified by period. Baseline error, false-positive rate (FPR) drift, and false-negative rate (FNR) drift decompose the distortion into attenuation from period-invariant error rates and the contributions of changes in each error rate between periods.

Table I12 Model performance for classification with most recent model generation

		(1)			(2)		
		Macro F1	Weighted F1	CR	Macro F1	Weighted F1	CR
Claude Opus 4.6 (reasoning)	Anthropic	.636	.809	.993	.605	.813	.966
Claude Sonnet 4.5		.612	.751	.991	.636	.811	.980
Claude Haiku 4.5		.576	.742	.997	.605	.801	.984
Mistral Large 3	M	.512	.667	.967	.609	.802	.920
GPT-5.2	OpenAI	.642	.822	.970	.676	.841	.979
GPT-5 Mini		.520	.635	.817	.667	.843	.901
GPT-5 Nano		.467	.516	.879	.675	.838	.829
N		1,000	1,000	3,000	1,047	1,047	3,000

Note. Recent frontier models are not open for supervised fine-tuning. LLM temperature = 0.0, except for GPT-5 Mini and GPT-5 Nano which only allowed temperature = 1.0. LLM seed = 94032.

Table J13 Scope conditions for the LLM-based framework

Task characteristic	Ex ante question	Recommended approach	Example settings
<i>A. Is the construct manifest in the text?</i>			
Textual cues carry the construct	Would a domain expert reach a judgment reading only the focal text?	The presented framework is well suited.	Release notes, product reviews, idea submissions, changelogs, grant abstracts
Construct requires verification beyond the focal text	Does the judgment require comparison against prior art or an external corpus?	Combine with corpus-comparison methods (e.g., Verhoeven et al., 2016)	Patent-style prior-art assessment
Construct is defined relationally across documents	Is the construct a property of the focal text relative to the full corpus rather than of the focal text alone?	Combine with embedding- or similarity-based methods (e.g., Just et al., 2024)	Novelty operationalized as distance to all other contributions
Construct requires non-linguistic evidence	Do experts rely on visual, auditory, or physical artifacts to judge the construct?	Currently unsuitable; multimodal extensions are a question for future research	Visual design novelty, aesthetic quality
Text producers strategically misrepresent	Do text producers have incentives and the ability to game descriptions in ways indistinguishable within the text?	Unsuitable where misrepresentation leaves no textual trace (Comino et al., 2019); monitor drift when the measure becomes a target	Promotional release notes; competitive funding pitches
<i>B. Can reliable expert ground truth be established?</i>			
Experts converge in their judgments	Do experts identify similar cues?	Unsuitable if reliability is low; no benchmark exists for validation (Einhorn, 1974)	Genuinely contested constructs without expert consensus
Domain experts are accessible	Can a small group of experts validate the data structure?	Required input to the framework	All applications
Target text corpus is temporally, culturally, or linguistically distant from the majority of the model’s training data	Can an expert-validated holdout sample be established for the target period and domain?	Suitable with holdout validation; validity does not transfer from contemporary text corpora	Historical documents; non-Western text corpora
<i>C. Is expert-level judgment required at all?</i>			
Construct is latent, language	Do dictionaries and classical ML fail to reach acceptable	Comparative advantage of the framework	Unstandardized descriptive text

heterogeneous, labeled data scarce	performance at feasible labeling cost?		
Construct resides in surface features, labels are abundant	Is terminology standardized, or is a large labeled dataset already available?	Suitable but dominated on cost by dictionaries or classical ML	Well-defined terminology, established labeled benchmarks

AI-Based Measurement of Innovation: Mapping Expert Insight into Large Language Model Applications

Abstract

Measuring innovation often relies on context-specific proxies and expert evaluation. Hence, empirical innovation research is often limited to settings where such data is available. We investigate how large language models (LLMs) can be leveraged to overcome the constraints of manual expert evaluations and assist researchers in measuring innovation. We first design an LLM framework that reliably approximates domain experts' assessment of innovation from unstructured text data. We then evaluate the performance and broad applicability of this framework through two demonstration studies in different contexts: (1) innovation in software application updates and (2) user-generated ideas in product reviews. We compared the performance (F1-score) and reliability (consistency rate) of our LLM framework against alternative measures used in prior innovation studies, and to state-of-the-art machine learning- and deep learning-based models. The LLM framework achieved higher F1-scores than the other approaches, and its results are highly consistent (i.e., results do not change across runs). This article equips R&D personnel in firms, as well as researchers, reviewers, and editors, with the knowledge and tools to effectively use LLMs for measuring innovation and evaluating the performance of LLM-based innovation measures. In doing so, we discuss the impact of important design decisions—including model selection, prompt engineering, training data size, training data distribution, and parameter settings—on performance and reliability. Given the challenges inherent in using human expert evaluation and existing text-based measures, our framework has important implications for harnessing LLMs as reliable, increasingly accessible, and broadly applicable research tools for measuring innovation.

Keywords: innovation measurement, expert evaluation, large language models, artificial intelligence, text analysis

1. Introduction

Innovation scholars have long been interested in measuring to what extent an activity or outcome is ‘innovative’ (Bellstam et al., 2021; Gault, 2018; Griliches, 1990; Miric et al., 2023b; Smith, 2005; Verhoeven et al., 2016). This is inherently difficult as innovative activity spans a broad spectrum, from developing new products, business models, and processes to adopting novel organizational structures or reconfiguring existing systems to create value in new ways (Schumpeter, 1934). Within a given activity and context, the degree of innovativeness can vary substantially and these innovation-related constructs are often not directly observable. Discerning, for instance, how novel or how technologically distant a given activity is typically requires the specialized knowledge of domain experts (Rosenkopf and McGrath, 2011). To illustrate, recognizing whether a user-generated idea for a new product feature departs substantially from what already exists, or merely varies it, requires experts’ contextual judgment of what the product already offers and how the idea relates to these existing features.

As a result, quantitative studies focus on contexts where expert-assessed data is publicly available. For instance, patent-related proxies are widely used because the patent application process involves a rigorous examination of an invention’s novelty by patent examiners at patent offices (Griliches, 1990). Similarly, in pharmaceutical contexts, clinical trials and new drug approvals offer a rich setting to capture expert-evaluated output from innovative activity (e.g., Haeussler and Assmus, 2021; Krieger, 2021). However, in many other domains, accessing such data or expert evaluators is challenging, limiting possibilities to capture innovative activity and risking a biased perspective that favors contexts where expert-based measures are readily available (e.g., Criscuolo et al., 2017; Moser, 2012). Alternatively, innovation researchers have to rely on

the support of experts recruited to assist in a specific project to manually assess and code innovation activities and outcomes, which severely limits sample size.

In response, scholars have begun to adopt natural language processing (NLP) and machine learning (ML) techniques to create alternatives for constructing quantitative variables based on textual data (Gentzkow et al., 2019; Miric et al., 2023a). While often and successfully implemented (e.g., Bellstam et al., 2021; Miric et al., 2023b), these techniques come with substantial limitations. First, many NLP techniques depend on unambiguous, structured language for accurate performance (e.g., Loughran and McDonald, 2011). Second, while ‘classical’ ML approaches can tolerate linguistic noise, they generally rely on large sets of expert-labeled training data to yield reliable outcomes (Hirschberg and Manning, 2015). In contexts where the nature of the data and the construct to be measured involves understanding nuanced, multi-dimensional information, these approaches struggle to perform on par with human expert evaluation (Gentzkow et al., 2019).

Large language models (LLMs) may enable us to overcome some of these limitations. Built upon the transformer architecture (Vaswani et al., 2017) and pre-trained on vast text corpora, LLMs are able to accurately capture many subtleties of human language. The latest generation of LLMs bring enhanced predictive capabilities while requiring little to no additional data to perform tasks (Brown et al., 2020). They can be instructed by written human language, so-called prompts, that determine models’ output generation. Outputs are often remarkably human-like across scenarios (Jones and Bergen, 2024), with LLMs even capable of adopting different viewpoints in experimental settings (Boussiou et al., 2024; Doshi et al., 2025). These advancements might allow the analysis of text data in novel ways: LLMs may be able to capture and mimic the experts’ evaluation process, holding the potential to scale expert evaluation with unprecedented performance. This way, they can democratize the analysis of unstructured text data to a wider

audience of scholars by lowering barriers to use, e.g., by having little data requirements and reducing the need for repeated manual assessment.

In this paper, we introduce and empirically evaluate a framework for how to develop an LLM application to measure innovation-related constructs based on unstructured text data that mimics how an expert would have evaluated these constructs. Recognizing that evaluations yield the most accurate results when informed by specialized expertise (Böttcher and Klingebiel, 2025; Csaszar and Eggers, 2013), our LLM framework involves a structured process to capture the evaluation resulting from experts’ reasoning and decision-making steps. First, we inductively identify content dimensions that indicate innovation in our contexts, which we then validate with domain experts to establish the foundation for prompt design. Second, we move beyond the default capabilities of the pre-trained model by fine-tuning it with expert-labeled training data. Third, we address the reliability of results in repeated analyses by testing variations in design decisions including prompting, model selection, fine-tuning data, and parameters.

We evaluate the performance and broad applicability of this framework empirically through two demonstration studies. In Study 1, we measure the innovation-related outcomes from product development activities in the context of software application updates. In Study 2, we identify user-generated ideas for product development activities in the context of product reviews. Both demonstration studies traditionally require methods relying heavily on human-coded labels due to the inherent complexity of the constructs being analyzed. Moreover, in both studies, the textual data serving as the basis for evaluation is highly nuanced, context-specific, and lacks clear structures or standardized guidelines, making outcomes such as updates or reviews subject to subtle linguistic variations.

In both demonstration studies, our LLM application outperformed (in terms of F1-scores) innovation measures recently used in scholarly work as well as established NLP techniques. When

fine-tuning with expert-labeled training data is possible, LLMs achieve their highest performance, as this data allows for targeted mitigation of biases—such as those stemming from structural misclassifications or failed classifications of texts—and ensures high reliability independent of design decisions. But even without expert-labeled training data, LLMs still achieve strong performance and consistency. Overall, our findings reveal that our LLM framework is highly adjustable to both the measurement objective and context, provided that specialized human experts are available to define measurement criteria and assess its accuracy on validation data.

Our work makes two main contributions. First, we add to an emerging line of research that applies LLMs as research tools to construct variables based on text data (Carlson and Burbano, 2026; Cheng et al., 2024; de Kok, 2025; Fyffe et al., 2024; Just et al., 2024). Our framework presents a low-barrier approach for researchers to access reliable expert-level evaluations in various innovation domains. Specifically, we provide a systematic process that guides researchers and R&D personnel in firms through a structured prompt design, fine-tuning, and validation of a pre-trained LLM informed by validated domain-specific content dimensions to create measures of interest. Researchers wishing to adapt our framework to their own context need only to define the construct of interest by identifying key indicators, collaborate with a few specialized experts for validation, and then follow our guidelines. Second, our demonstration studies introduce novel, high-quality measures that enable more precise estimations of innovation-related constructs in software application development and product review contexts. They offer researchers and R&D personnel in firms a robust alternative to traditional proxies and human expert assessments. The respective code, local models, training and validation datasets, and fine-tuning approaches are all made public in an online repository¹ to support replication and adaptation for future research.

¹ All data, code, and supplementary materials needed to reproduce our results are openly accessible in the project's GitHub repository (https://github.com/robi979/AI-Innovation-Measurement_Extension). We also provide a

2. Expert Evaluation in Innovation Research

Measuring innovation-related constructs requires clarifying which aspect of innovative activity is meant to be captured. Prior literature has differentiated innovative activity according to the type and degree of change involved. For instance, innovations may preserve or disrupt existing technological and market linkages (Abernathy and Clark, 1985), differ in whether they alter a product's components or the architecture linking them (Henderson and Clark, 1990), or emerge from searching for unprecedented combinations of existing knowledge (Fleming, 2001). Reviews of this literature stream synthesize these perspectives by emphasizing that innovation can be conceptualized along multiple dimensions, including whether it concerns a process or an outcome, whether it occurs at the product, process, organizational, or market level, and whether it represents a minor improvement or a more substantial departure from existing practice (Crossan and Apaydin, 2010; Garcia and Calantone, 2002). These perspectives illustrate that innovation-related constructs are context-specific operationalizations of particular aspects of innovative activity. Accordingly, the measurement of such constructs rests on identifying the aspect of innovative activity that is theoretically relevant in a specific setting and the observable cues through which this aspect manifests in the available data.

Because innovation-related constructs are rarely directly observable and their cues are not self-evident, their measurement often depends on expert judgment. This helps explain why scholars frequently rely on measures whose institutional characteristics already embed elements of such judgment. Patents, in particular, are by legal definition related to inventiveness and undergo a rigorous examination process to ensure novelty, non-obviousness, and industrial applicability

lightweight 'application package' containing the essential code and example notebooks that show how to run our framework directly through common LLM-provider APIs.

(Title 35 United States Code), relying on specialized, highly skilled professionals such as patent examiners of patent offices to evaluate technical details and legal requirements. Extending this level of rigor to other spheres of innovation is challenging. Unlike patents, many forms of innovation lack formalized frameworks, legal structures, or dedicated expert evaluators. Still, reliable construct measurement often demands a clear understanding of the construct, deep domain knowledge and well-developed cognitive frameworks to comprehend, organize, and interpret construct-relevant information—expertise that is both scarce and difficult to scale.

This constraint has become increasingly salient as the growing availability of text data unlocks new opportunities to gain insights into innovative activities. On the one hand, organizations can now easily access broad pools of ideas from several sources, which allow them to source novel solutions to innovation challenges from both internal and external contributors (Boudreau et al., 2011; Jeppesen and Lakhani, 2010). On the other hand, such organizations themselves generate much text data, providing opportunities to detect patterns in their innovative activities beyond what is captured by traditional measures like R&D intensity or patent-related outputs (Bellstam et al., 2021). Exploiting these new data sources, however, requires expert judgment to identify construct-relevant information and evaluate it consistently.

Even when experts are available, scaling their evaluation is inherently constrained by time or budget constraints and human cognitive limits. Manually processing large volumes of complex information and identifying meaningful construct-relevant patterns is tedious and time-intensive. Evaluators often struggle to manage the complexity of vast (multi-dimensional) datasets, particularly when faced with uncertain aspects that cannot be easily connected to existing knowledge structures. Psychology research highlights that when overwhelmed with too much information at once, humans fail to organize and interpret the data thematically, leading to

suboptimal decision-making (Tversky and Kahneman, 1974). These cognitive limits hinder the ability to process complex information effectively and impede innovation evaluation at scale.

The difficulty of processing large datasets not only constrains evaluators’ ability to assess information but also narrows their scope of attention (Samuelson and Zeckhauser, 1988). Faced with complex or uncertain information at scale, evaluators often resort to heuristics, reinforcing existing biases or adhering to prevailing concepts (Tversky and Kahneman, 1974). Their attention space then tends to align with established knowledge structures, causing them to overlook unconventional information that falls outside their expertise (Boudreau et al., 2016; Piezunka and Dahlander, 2015). For instance, evaluators may judge the value of new unfamiliar information based on prior decisions rather than its intrinsic merits, potentially explaining why established technologies persist even when superior alternatives exist (Abrahamson, 1991). Taken to the extreme, this tendency to favor the status quo may result in the outright rejection of potential innovations when increased workload overwhelms evaluators’ cognitive resources.

However, thorough evaluation is essential for measuring valid constructs, particularly when leveraging the growing availability of large datasets to explore hitherto less researched contexts. Advances in NLP techniques, driven by progress in AI research, now present an opportunity to overcome constraints imposed by human cognitive limits in processing information (Korinek, 2023). And in fact, first studies have demonstrated how these novel techniques can complement or even substitute human capabilities in generating measures (e.g., Choudhury et al., 2019; de Kok, 2025; Fyffe et al., 2024; Miric et al., 2023a), decision-making (Doshi et al., 2025), and idea selection (Just et al., 2024), paving the way for a deeper understanding of how these tools can be utilized to enhance both research and practice. An overview of prior text-based approaches for approximating expert evaluation is provided in Online Appendix A.

3. Mapping Expert Insight into Large Language Model Applications

Evaluations in innovation contexts often place a strong emphasis on qualitative judgments such as assessing an idea’s novelty (Boudreau et al., 2016; Lane et al., 2024). Such judgments are inherently interpretive, drawing upon individual perceptions (Kim and DellaPosta, 2022; Lane et al., 2022), intuition (Huang and Pearce, 2015), and expertise (Boudreau et al., 2016; Li, 2017), and typically lack universally accepted benchmarks at the time of evaluation. Hence, selecting appropriate evaluators is essential for high evaluation quality and reliability.

Research on innovation evaluation indicates that uncertainty in evaluations is minimized when domain experts can be involved (Böttcher and Klingebiel, 2025; Boudreau et al., 2016; Csaszar and Eggers, 2013). Through years of learning and deliberate practice, experts develop distinctive know-how that enables them to, often intuitively, recognize intricate patterns and extract meaningful insights from complex information (Ericsson and Smith, 1991; Johnson et al., 1982; Kahneman et al., 1982). As a result, they exhibit better judgment, allowing them to assess tasks and contextual cues more accurately than nonexperts and ultimately make more informed evaluations in their domain.

At the same time, this raises the question of what qualifies an evaluator as a domain expert for evaluation tasks. Research on experts suggests that the validity of expert judgment is a function of the fit between the evaluator’s domain of practice and the characteristics of the judgment task and not a general attribute of credentials or seniority (Shanteau, 1992). Expert judgment is most reliable when the task environment exhibits learnable regularities and the evaluator has accumulated repeated, feedback-rich experience with those regularities (Ericsson and Smith, 1991; Kahneman et al., 1982). For text-based construct measurement, the relevant expertise therefore concerns the data-generating process behind the analyzed texts, meaning that evaluators need to

command the conventions and vocabulary of those who produce these texts, while familiarity with the theoretical construct alone is not sufficient. Conversely, when an evaluator’s domain of practice and the judgment task are misaligned, evaluation quality can degrade below that of generalist evaluators (Böttcher and Klingebiel, 2025; Csaszar and Eggers, 2013). Selecting evaluators thus involves three considerations: (i) defining *ex ante* which judgment task the construct of interest requires, (ii) choosing evaluators whose demonstrable, current practice aligns with the data-generating process underlying the text data, and (iii) documenting the recruitment and qualification criteria as part of the research design.

In this study, we build on these considerations and aim to integrate the distinctive know-how of appropriately selected domain experts into LLM-based applications, so that researchers can apply them as reliable tools for approximating expert evaluation. Although LLMs excel at extracting insights from large datasets and can generate assessments that are consistent with human judgment (Brown et al., 2020), it is essential to recognize that they operate on statistical pattern recognition and not genuine logical reasoning or expertise-based intuition (Mitchell and Krakauer, 2023). This reliance on probabilistic modeling often results in variability in outputs across runs. To address this, we suggest a systematic mapping of the cues and implicit criteria that experts draw upon in their judgments and evaluations. By explicitly codifying these aspects in the input options of an LLM, it becomes possible to steer its internal probabilities, enabling outputs to more closely mirror expert-level reasoning and assessments (Liu et al., 2023).

We begin with outlining a stylized framework to discern cues indicative of expert judgment (Einhorn, 1974). This involves initially *identifying relevant cues and information* by clarifying the evaluation objective and criteria and engaging with the available data to isolate evaluation-relevant cues. Next, the identified cues are *organized into clusters or dimensions* by contextualizing each through its connection to existing knowledge. This allows us to *assess the amount of each cue* as

it is reflected in the data element (e.g., a certain text) being evaluated. Finally, the organized and assessed *cues are weighed* according to their importance relative to the evaluation criteria, ultimately informing the final judgment. It is important to recognize that expert judgment often appears automatic and even instantaneous due to experts’ deeply internalized knowledge structures (Chase and Simon, 1973; Dane and Pratt, 2007; Ericsson and Smith, 1991; Sukhov et al., 2021). Such judgments may draw on learned, tacit conceptions of similarity that enable experts to recognize patterns that are difficult to reduce to explicit information-processing steps (Nightingale, 1998). Hence, it is unrealistic to assume that all experts consciously and explicitly follow the outlined steps. Still, this stylized framework assists in a structured identification of relevant cues indicative of expert judgment, enabling researchers to instruct an LLM to generate outputs that better reflect expert reasoning.

To map this framework into LLM applications, we create a prompt-data-output combination that mirrors the results of expert reasoning. In other words, rather than trying to replicate the complex and often implicit nature of expert reasoning, we embed relevant evaluation cues within the LLMs’ input options. LLMs operate by generating text token by token, with each token selected from a probability distribution that is conditioned on the input prompt, model tuning parameters (such as temperature settings), and, when available, fine-tuning data. When we craft prompts that mirror the described stylized LLM framework for an evaluation task—starting with a clear task definition, moving to the identification, clustering, and assessment of relevant cues, and finally weighing these cues relative to evaluation criteria—we can effectively steer the model’s internal probabilities toward outputs that reflect those from expert-level reasoning (Liu et al., 2023). Furthermore, fine-tuning the model with domain-specific, labeled data can further reinforce this alignment.

This mapping implies a division of roles between human experts and an LLM. Expert judgment informs the framework at the design stage, where experts define the construct and its indicative cues, validate the inductively derived data structure, weigh the relevance of its dimensions, and adjudicate the labels against which model outputs are validated. The LLM in turn executes this codified judgment by repeatedly applying the expert-defined evaluation scheme to large volumes of text. The division of roles also shapes the implementation effort of the framework. Human effort is concentrated in the iterative design and validation stages, with the required effort depending on the iterations needed to reach saturation and expert agreement and, where fine-tuning is used, on the amount of labeled training data required. Once a validated configuration is established, scaling the measure to additional observations requires little additional human involvement. Notably, the large-scale application of expert-developed evaluation schemes has traditionally rarely been performed by experts themselves. For instance, in quantitative content analysis, applying an expert-developed coding scheme at scale is routinely delegated to trained human coders whose reliability is monitored through inter-coder agreement statistics (Krippendorff, 2019).

In substituting for such non-expert coding work, the LLM takes on a role closely related to research that employs LLMs as simulacra of humans, for instance to augment survey data (Brynjolfsson et al., 2026). Operationally, these applications proceed much like our framework, conditioning a model on structured prompts and human-provided information so that its outputs approximate the judgments of individuals. Our framework also relies on this capability at the execution stage. What converts it into an expert-aligned measure is the apparatus built around it, as the execution is embedded in the codified evaluation scheme and the labeled fine-tuning data that reflect expert reasoning.

----- Insert Table 1 about here -----

Table 1 illustrates the framework for designing LLM inputs to measure constructs from unstructured text data, mimicking how an expert would have processed the evaluation. The framework begins with the researcher explicitly defining the task and desired output in the prompt’s opening section, aligning it with the specific measurement objectives.

Next, the LLM framework leverages inductive coding of data (Gioia et al., 2013; Strauss and Corbin, 1998) until thematic and meaning saturation are reached. This approach ensures that the prompt captures the critical cues that experts use to filter and prioritize information. In addition, relevance weights can be integrated directly into the prompt or applied during post-processing. This step is crucial because it aligns the model’s output with expert-informed priorities and reflects the key elements of the data structure relevant for the measured construct. It is imperative that experts are directly involved or iteratively consulted during this phase to validate the final data structure, thereby laying the groundwork for a robust coding scheme—especially if fine-tuning with further training data is planned.

The final stages of the LLM framework involve validating the LLM’s output against a holdout sample and, whenever possible, other established metrics. Based on performance outcomes, fine-tuning with labeled training data can be applied and design decisions may have to be adjusted. Researchers and R&D personnel need to systematically document decisions such as model selection², prompt engineering techniques (Wei et al., 2022), temperature settings (Peeperkorn et al., 2024), and fine-tuning configurations (Gao et al., 2020) to ensure reliability.

² Since 2024, reasoning models (e.g., OpenAI’s o-series) have emerged as a class of LLMs that generate intermediate reasoning tokens before producing an output. These models internalize a multi-step procedure similar to the one our framework specifies explicitly, with the difference that the generated reasoning is model-derived, typically acquired through reinforcement learning on tasks with verifiable solutions such as mathematics and code, and thus does not necessarily represent the expert reasoning required for the task. Because providers expose this reasoning only in truncated form, it cannot itself be verified, and validation must rest on the outputs alone. As fine-tuning access is currently unavailable for these models, they enter our framework as an alternative model choice in default configuration, subject to the same validation standards as any other model. We discuss the implications in our Demonstration Studies in Section 4 and the corresponding Online Appendix.

For instance, tuning parameters like temperature allow control over the randomness of token selection: lower temperatures favor high-probability tokens for more deterministic outputs, while higher temperatures facilitate more varied outputs.

Understanding the implications of such specific decisions is crucial for optimizing both performance and consistency, ensuring that the model’s outputs remain reliable across repeated runs. Similar to human expert evaluation, reliability measures such as intra-rater reliability (consistency within individual repeated assessments) and inter-rater reliability (consistency among multiple evaluators) are prerequisites for valid judgments, confirming the robustness of the evaluation process and derived conclusions (DeVellis and Thorpe, 2022; Einhorn, 1974).

Because expert judgments serve as the standard against which the LLM application is calibrated and validated, their reliability requires the same systematic treatment as any other design decision. We recommend that the labeling of training and validation data involve at least two raters and that inter-rater reliability be reported against conventional benchmarks (Krippendorff, 2019; Landis and Koch, 1977). Disagreement between raters should not be treated as noise to be resolved ad hoc; it is diagnostic information about ambiguity in the construct definition or the data structure. Recurring disagreement therefore feeds back into the refinement of the coding scheme, which is iterated until meaning saturation and stable inter-rater agreement are jointly reached (Gioia et al., 2013), with a documented adjudication rule governing residual cases.

Two implications follow. First, human inter-rater agreement constitutes the ceiling on validated model performance, since an LLM application cannot be meaningfully validated beyond the reliability of the benchmark against which it is assessed. Second, the expert-adjudicated holdout sample provides the standard that disciplines evaluator-related researcher degrees of freedom. A different selection of experts could, in principle, yield a different coding scheme. The framework

makes this choice testable as any alternative evaluator configuration constitutes a rival specification whose adequacy can be assessed against the same validation logic.

4. Demonstration Studies

4.1. Demonstration Study 1: Innovation in Software Application Updates

In the first demonstration study, we apply our systematic framework for innovation measurement to analyze and classify software application updates. Software application development is inherently generative, allowing for continuous product evolution through optimization, reuse, and novel combination of digital elements even after initial launch (Zittrain, 2006). Digital platforms enhance this process by providing infrastructure and tools that reduce costs and accelerate development, fostering rapid experimentation and sequential innovation (Parker and Van Alstyne, 2018). Low barriers to entry cultivate active developer communities and intense competition, leading to fast innovation cycles and early product releases (James et al., 2023; Lee and Raghu, 2014). However, these same dynamics also incentivize the frequent release of low-quality updates or updates with little improvement to the software application and without introducing novel elements (Comino et al., 2019). When studying innovation behavior in many digital contexts, it is therefore crucial to differentiate between updates that merely refine existing products or “fix bugs”, and those that contribute genuine innovation.

Recent methods for measuring innovation in software application updates include analyzing version numbers (Boudreau, 2012; James et al., 2023; Wen and Zhu, 2019) and release notes (Foerderer et al., 2018; Kircher and Foerderer, 2024; Leyden, 2022). Relying on version numbers assumes that significant numerical changes indicate major updates, but developers often have incentives not to adhere to versioning protocol conventions³, potentially distorting such

³ See <https://semver.org/> for a popular example of such conventions.

metrics. Text analysis of release notes improves upon this issue but is mostly conducted using dictionary-based approaches or ‘classical’ machine learning methods. However, the vast heterogeneity in how developers describe updates and the absence of standardized terminology make it challenging to capture all variations accurately. Since datasets in such settings can be extremely large—often involving up to millions of descriptive texts—relying on human experts becomes impractical, albeit necessary, underscoring the need for advanced methods that can accurately assess innovation in software updates.

4.1.1. Data

An important aim of this demonstration study is to evaluate our proposed LLM framework and compare it to methods used in previous studies. To achieve this, we require a coded sample of software application updates, specifically requiring knowledge about the update and classification of its content. For comparability with previous studies, our data must include update information reported in release notes as well as the corresponding version numbers.

We obtained the data for this demonstration study from MixRank, a provider of app data. From a large dataset covering the complete update history of every app active on the Apple App Store between 2018 and 2021, we randomly sampled 4,000 update descriptions⁴ to form a pool for creating training and validation datasets. To ensure consistent analysis, we filtered the update descriptions to include only those written in English.⁵ Each update description contains a corresponding release note and version number.

⁴ On the App Store, these release notes are displayed in a section titled “What’s New” and limited to 4,000 characters. Since 2018, it is mandatory for developers to specify update details and avoid generic texts.

⁵ Our demonstrations use English-language text to ensure comparability with dictionary and classical ML baselines. The framework itself is not language-specific as LLMs show increasing performance on multilingual datasets (e.g., OpenAI, 2023). Online Appendix K discusses its application to multilingual corpora.

To establish ground truth for the sample pool, a co-author and a research assistant coded the release notes in the pool according to a coding scheme developed by the research team through inductive coding. This scheme comprises seven content dimensions (classes) and was validated in interviews with software application developers. A subset of $n=1,000$ update descriptions was independently coded by both raters to ensure high inter-rater reliability (observed agreement = .91, $\kappa = .87$). Any discrepancies were discussed to be consistent across the remaining sample pool.

Subsequently, we generated two random subsamples from the labeled sample pool of $n=4,000$ release notes: a validation set of $n=1,000$ release notes serving as a holdout and a training set of $n=2,000$ release notes. Both subsamples were constructed to reflect a representative class distribution of the overall pool. The remaining $n=1,000$ release notes are used in additional analyses to allow variations with class distributions.

4.1.2. Research design

We anchored our measurement objective in the literature’s conceptualization of an innovative update in mobile software applications. In this context, innovative updates are defined as those that introduce additional, user-salient functionality or content to the app, rather than “under the hood” changes made primarily for internal purposes and imperceptible to consumers (Foerderer et al., 2018; Leyden, 2022). In other words, the primary task of our prompt is to extract and analyze the content of release notes which we then use for an informed decision whether an update is innovative according to this definition.⁶

In a next step, we conduct inductive coding of release notes until reaching meaning saturation. This iterative process yielded seven content dimensions that characterize the data

⁶ We acknowledge that back-end modifications such as performance improvements, security hardening, or code refactoring may also be labeled “innovative” when viewed as process innovations. Our study, however, focuses explicitly on consumer-visible product innovations that add new functionality or content.

structure of release notes and enable us to distinguish between updates reflecting innovative activity (dimensions 1 and 2) and those intended primarily to fix bugs, introduce minor improvements, or promote the app (dimensions 3-7; cf. Online Appendix B1). To validate our measurement objective and data structure, we conducted several interviews with software application developers (cf. Online Appendix B1).

----- Insert Figure 1 about here -----

Following our proposed framework, both inform the prompt that maps the process of expert reasoning into an LLM-compatible format (cf. Figure 1). The validated content dimensions build the foundation for assigning human-coded labels to the release notes in our sample pool. Assigning these labels also serves as an additional check on data structure saturation, confirming that no further coding dimensions are necessary.

Our framework is applied to LLMs of various sizes, versions, and providers—both in their default state and after fine-tuning—with all models accessed via provider APIs. For fine-tuning, the training data is prepared by integrating the release note text with its corresponding labeled output via our designed prompt. The generated outputs are evaluated against our validation dataset and benchmarked against the performance of methods from previous studies as well as ‘classical’ machine learning models, deep-learning approaches, and transformer-based pre-trained language models on the same validation dataset. The inference workflow and the output processing procedure are documented in Online Appendix C.

To replicate prior work, we incorporated the preprocessing steps, dictionaries and filters described in Agarwal and Kapoor (2023) and Kircher and Foerderer (2024) and we applied version logic similar to that used in Wen and Zhu (2019).⁷ For benchmarking the other models, we trained

⁷ Kircher and Foerderer (2024) classify an update as innovative (feature update) if the release note contains at least one of the keywords “new”, “added”, “upgrade”, or “major”. Agarwal and Kapoor (2023) classify an update as

each with the same labeled training data as for our LLM fine-tuning, albeit without the prompts. To identify the best-performing model setup, we conducted several iterations with different embedding approaches and hyperparameter settings. For each combination of embedding technique and classifier, we systematically optimized model hyperparameters via randomized search and stratified cross-validation across multiple dataset partitions. This entailed testing a range of embedding and model-specific configurations to robustly identify optimal setups for each method (Goodfellow et al., 2016). The full hyperparameter grids, search procedures, and rationale for model selection are described in detail in our Online Appendix D.

4.1.3. Results

Table 2 reports performance (Accuracy, F1-score, Precision, Recall) and reliability metrics (Consistency Rate) for identifying innovative updates, comparing (1) rule-based and dictionary-based approaches from prior literature, (2) ‘classical’ machine learning (ML) models, (3) neural architectures such as convolutional neural networks (CNN), (4) earlier transformer-based pre-trained language models (e.g., BERT), and (5) our framework applied to several large language models (LLMs) from different providers, with varied parameter sizes, and fine-tuning state. Accuracy measures the overall proportion of correct predictions, providing a straightforward assessment of model performance. However, to account for class imbalances and differentiate between false predictions, we use the F1-score, a standard composite metric calculated as the harmonic mean of Precision and Recall. Precision indicates the proportion of predicted innovative

innovative (new functionality) if the release note exceeds 200 characters or contains at least one of the keywords “introduce”, “feature”, “support”, “performance”, “improve”, “enable”, “update”, “enhance”, “modify”, “optimize”, “fast”, “adjust”, or “multitask”. To account for differences in text representation and ensure consistency in the keyword matching process, we applied several preprocessing steps. We converted each release note to lowercase, tokenized it into individual words, and each word was reduced to its stem. The keywords in each dictionary were processed similarly. In Wen and Zhu (2019), the baseline treats every update release as an innovative update. A version number rule is applied in robustness checks, classifying an update as innovative (major update) if the <major> field in the app’s version number (first digit) changed. When replicating this approach, we additionally controlled for changes in the <minor> field in the app’s version number (second digit).

updates that are actually correct, while Recall measures the proportion of actual innovative updates correctly identified. Additionally, we evaluate model stability using the Consistency Rate (CR), which quantifies how reliably a model produces the same predictions across different runs.

----- Insert Table 2 about here -----

Relying on version numbers (i.e., distinguishing whether the “first digit” or the “second digit” of the version changed) alone yields low F1-scores (below .30 for the “Version First Digit” heuristic), confirming prior concerns that app developers do not consistently follow versioning protocols. Dictionary-based methods show better performance (F1-score up to .67), but remain noticeably behind more sophisticated approaches, reflecting the limited ability of fixed keyword lists to capture the richness of how new features are described in release notes.

Among ‘classical’ ML approaches (e.g., logistic regression, random forest, and support vector machines), F1-scores range from approximately .70 to .85, with logistic regression and support vector machine performing best within this category. The convolutional neural network performs similarly with an F1-score of .828, and early transformer-based pre-trained language models (PLM) perform even better with F1-scores close to .90. Crucially, all three categories of models can be run locally and reproduced under identical settings and training data, ensuring full replicability. However, each requires a substantial volume of labeled data to achieve their upper performance bounds (in this study, $n = 2,000$).

Even more compelling is the performance of several LLMs integrated into our framework. Without any additional training data (i.e., in their default configuration), and based on our framework, several LLMs achieve F1-scores on par with or better than PLMs, with GPT-4.1 reaching the highest F1-score of .901. Still, other default configurations such as GPT-4.1 Nano or Claude 3.5 Haiku fall below the performance of ‘classical’ ML models, suggesting that not all LLMs excel ‘out of the box’ and that models with fewer parameters perform worse than their larger

counterparts. However, once fine-tuned with corresponding input-output pairs, performance improves across the board, with OpenAI’s GPT models edging F1-scores up to .926.

Notably, even in their default configuration, LLMs maintain high precision, meaning they rarely mislabel non-innovative updates as innovative. In contrast, recall (i.e., the ability to correctly identify the full range of innovative updates) improves more markedly with fine-tuning. This suggests that, much like ‘classical’ ML models or earlier PLMs, additional training data helps LLMs to learn the varied and sometimes subtle language used to describe new features, thus better capturing the ‘breadth’ of innovative updates.

In practice, many LLMs are accessible only via an external API, making local deployment impractical or impossible. To nonetheless ensure reliability in these scenarios, we standardize our setup by setting the temperature parameter to 0 and by defining a model seed⁸. This adjustment produces deterministic outputs from the same model versions under identical prompts, with CRs above .98. The implications of variations in model selection, temperature settings, and training data for performance and reliability are further discussed in Online Appendix E. To still enable scalable and reproducible deployment beyond external APIs, we additionally implement a teacher-student distillation setup described in Online Appendix F. In this setup, a high-performing API-based “teacher” LLM generates labels that are used to train smaller (open or local) “student” models, enabling low-cost, locally deployable inference with performance close to the teacher.

----- Insert Table 3 about here -----

Table 3 presents a more fine-grained, multi-class version of the update classification. Columns 1-7 reflect dimensions from the data structure and prompt, with per-class F1-scores,

⁸ Defining a seed parameter fixes the pseudo-random sampling state, so identical prompts and parameters return the same output when the model (*fingerprint*) is unchanged. Omitting the seed—even with temperature=0 calls—lets stochastic tie-breaks and sampling noise vary across runs and can reduce consistency across runs.

macro- and weighted-average results, and the CR across three independent runs. The Macro F1-score averages performance equally across all classes, ensuring both common and rare updates are considered, whereas the weighted F1-score uses sample-based weights to reflect the dataset’s true distribution. Comparing macro- and weighted-averages helps clarify whether models struggle on underrepresented classes (via macro-average) or show strong overall performance on the more frequent classes (via weighted-average). Large discrepancies between these averages can signal that certain classes are being overlooked.

Earlier dictionary- or version-based methods are not included in this analysis because they rely on fixed heuristics or keywords that cannot differentiate between subtle update types. ‘Classical’ ML and early PLMs (e.g., ELECTRA) perform robustly overall but fail to detect rare classes when training data are scarce. Although misclassifying classes that serve a similar measurement objective may not drastically affect the overall innovation score, failing to detect a class altogether can skew outcomes and introduce bias.

By contrast, default LLMs demonstrate more balanced performance across all classes. When fine-tuned with the same training data used by ML or PLM approaches, LLMs (e.g., GPT-4.1 or GPT-4.1 Mini) can achieve a macro-averaged F1-score of up to .720, showcasing the advantages of our proposed framework and the capacity of novel LLMs to precisely classify content. Consistency rates (CR) for both default and fine-tuned LLMs remain very similar and consistently far exceed typical human inter-annotator agreement standards (Artstein and Poesio, 2008; Carletta, 1996; Landis and Koch, 1977). Although the models are accessed via an API and not in a local setup, with the specified temperature and seed settings, both default and fine-tuned versions produce identical outputs across independent runs, with only rare discrepancies.

4.2. Demonstration Study 2: User Ideas in Product Reviews

In our second demonstration study, we apply and evaluate our framework to classify user-generated product reviews. External sources of innovation such as user ideas are invaluable for organizational innovation activities, providing access to knowledge and insights that lie beyond organizational boundaries (Chesbrough, 2003; Von Hippel, 1978). However, research on crowdsourcing and idea evaluation highlights that identifying ideas that are both original and practically relevant, and that genuinely add value to organizations' innovation activities, remains challenging due to the overwhelming number of suggestions, their often unstructured nature, and evaluators' limited attention in idea contests and review platforms (e.g., Piezunka and Dahlander, 2015).

Previous methods that analyze customer ideas and needs in product reviews often rely on expert committees or advanced NLP techniques backed by large training datasets (e.g., Timoshenko and Hauser, 2019; Zhang et al., 2021). While effective, these approaches can easily become resource-intensive, requiring extensive training data, and significant involvement from many domain experts. We illustrate LLMs' potential to generate valid classifications of user ideas with less reliance on extensive data or expert involvement.

4.2.1. Data

Our dataset, sampling, and preprocessing procedures are comparable to those used in Demonstration Study 1. From a large dataset covering the complete review history of all apps that have been listed at least once in the top-100 overall or category-specific rankings, we randomly sampled 4,000 reviews written in English.

To establish ground truth for the sample pool, a co-author and a research assistant independently coded the reviews using a coding scheme developed through inductive coding by our team. This scheme comprises nine content dimensions. A subset of $n=1,000$ review

descriptions was independently coded by both raters to assess inter-rater reliability (observed agreement = .89, macro-average κ = .77), and any discrepancies were discussed.

In line with Demonstration Study 1, we generated two distinct random subsamples from the labeled pool of $n=4,000$ reviews: a validation set of $n=1,000$ reviews serving as a holdout, and a training set of $n=2,000$ reviews. Both subsamples were constructed to reflect a representative class distribution of the overall pool. The remaining $n=1,000$ reviews of the labeled pool serve to back additional analyses to explore variations in class distributions.

4.2.2. Research design

Our measurement objective builds on the premise that product reviews often contain valuable suggestions, ideas, and expressions of customer needs (Timoshenko and Hauser, 2019; Zhang et al., 2021). These reviews are multifaceted, ranging from brief comments to extensive narratives where users may suggest or report several issues at once. As such, our prompt is aimed at distinguishing between different types of content and identifying instances where multiple topics are addressed simultaneously.

----- Insert Figure 2 about here -----

To further inform prompt design, we conducted inductive coding of reviews until meaning saturation was reached. This process resulted in nine content dimensions that characterize the reviews' data structure and provide a means to differentiate them at a highly granular level (cf. Online Appendix B2). User-generated ideas are most directly captured by dimension 3 (suggestions for new features or content or revival of removed features), while the remaining dimensions distinguish other feedback types, such as bug reports or general opinion.⁹ Both our measurement

⁹ Innovation-related input is not confined to explicit suggestions. For instance, reports of bugs or monetization issues can likewise reveal unmet customer needs (Timoshenko and Hauser, 2019). The granular data structure allows researchers to include or exclude such content depending on the construct of interest in subsequent analyses.

objective and the data structure are then integrated in the final design of our prompt (see Figure 2) and the content dimensions serve as the foundation for labeling reviews in our sample pool.

Our framework is applied to the same LLMs used in Demonstration Study 1. To prepare training data for fine-tuning, we created input-output pairs by combining every merged prompt and review text with its corresponding manual label. The LLM-generated outputs are then evaluated against our validation dataset and benchmarked against the performance of ‘classical’ machine learning models, deep-learning approaches, and transformer-based pre-trained language models on the same validation dataset. For benchmarking against the other models, we used the same labeled training data as our LLM fine-tuning, though without the prompt. Given the multi-label nature of our task, we employed a one-vs-rest strategy, where each label is encoded and classified as an independent binary feature to capture the presence of multiple labels per review. For instance, if a review’s labels were 8 and 4, this method converted it into a binary vector [0 0 0 0 1 0 0 0 1] representing the presence of labels from “0” through “8”. Similar to Demonstration Study 1, we experimented with different embedding techniques and hyperparameter settings to identify the best-performing setup. The rationale behind our configuration for each model is detailed in Online Appendix D.

Although product reviews, and particularly app reviews, are a common subject in NLP tasks and have been extensively explored for feature extraction and content classification (e.g., Gunathilaka and de Silva, 2025; Khlifi et al., 2024), no study has, to our knowledge, tested such fine-grained content differentiation or demonstrated LLMs’ capabilities in a multi-label setting.

4.2.3. Results

Table 4 shows the results of applying our framework to an analysis of reviews, reporting comparable performance and reliability metrics as in Demonstration Study 1. The nine columns (0-8) correspond to the content dimensions derived from the data structure and included prompt.

Because each review can receive multiple labels in the one-vs-rest approach, the total number of labeled instances (n) exceeds 1,000, even though the holdout sample contains 1,000 reviews.

----- Insert Table 4 about here -----

The overall observations from Demonstration Study 1 are corroborated here. ‘Classical’ machine learning models vary considerably, with random forest showing the highest Macro F1 score (.401), and K-Nearest-Neighbor (Macro F1 score = .208) yielding the lowest performance. Applying pre-trained language models (PLMs) results in modest improvements, with XLNet achieving a Macro F1 score of .436. Notably, applying our framework in LLMs outperforms these approaches, with even default configurations achieving superior results. The best performance is obtained through fine-tuning, with OpenAI’s GPT models achieving top Macro F1 scores (e.g., GPT-4.1 = .706).

Models that leverage additional training data or benefit from fine-tuning particularly excel in classes with larger sample sizes. Interestingly, default versions of LLMs can achieve F1 scores comparable to their fine-tuned counterparts and deliver moderate performance in scenarios where traditional methods struggle, such as with underrepresented samples or linguistically complex cases (e.g., class 6). Further performance enhancements are achievable by fine-tuning labeled prompt-review-output combinations. Consistent with Demonstration Study 1, the temperature parameter is set at 0, and the model seed is fixed to systematically assess reliability metrics. Also in this multi-label setting, outputs across three independent runs remain highly consistent, with consistency rates persistently above .91. We discuss variations in model selection, temperature settings, and training data in Online Appendix E.

4.3. The Relevance of Model Fine-Tuning

Both demonstration studies compare default and fine-tuned model configurations. The performance results reveal different gains from fine-tuning across settings, indicating that the value

that fine-tuning adds should be assessed with regard to a study's classification task and measurement objectives. The decision whether to include model fine-tuning is consequential: fine-tuning requires expert-validated training data that is time-intensive to create and entails further choices regarding the size and distribution of this data.

For classifications with common and balanced classes, default configurations approach the performance ceiling and the gains from fine-tuning are unlikely to alter substantive conclusions (Table 2; e.g., GPT-4.1 default F1 = .901 vs. .926 fine-tuned). Small increases in the F1-score should be weighed carefully against the costs of fine-tuning. In some cases, fine-tuning can even reduce performance. With limited training data under a representative class distribution, the performance of fine-tuned models may fall below the performance of their own default configurations. The training volume beyond which fine-tuning pays off varies by model and task and is only revealed through holdout validation (Online Appendix Figure E4; in our setting, a fine-tuned GPT-4.1 stays below its default performance until $n = 500$, Mistral models until $n = 2,000$).

Fine-tuning is most valuable when default configurations miss or under-recall construct-relevant instances. This applies when the construct resides in rare classes (Table 4; e.g., GPT-4.1, class 4 F1 from .727 to .807). Fine-tuning may also add value, even when performance gains are small, when a research design requires maximizing the identification of relevant instances, as default LLMs already classify with high precision while fine-tuning primarily raises recall (Table 2; e.g., GPT-4.1 recall from .879 to .961). Fine-tuning thereby changes the error structure. When a class is missed entirely without fine-tuning, all of its instances are measured incorrectly, which constitutes systematic measurement error and can bias downstream estimates. After fine-tuning, these classes are detected, and the remaining misclassifications are fewer and no longer concentrated in single classes. Because these gains are not uniform across models and classes

(Table 4; e.g., GPT-4.1 Mini, class 6 F1 from .629 to .435), per-class validation on the holdout sample should accompany any fine-tuning decision.

Finally, fine-tuning can serve deployment. In the teacher-student configuration (Online Appendix F), a high-performing, often proprietary and API-based, LLM labels training data to fine-tune a smaller, locally deployable student model. Fine-tuning on these labels brings the student model close to the teacher’s performance, while inference runs locally at a fraction of the teacher’s (recurring) costs and independent of provider-side changes.

Online Appendix G provides a structured summary of these contingencies. The overarching criterion is the structure of the remaining errors. Where default configurations already err in largely unsystematic ways, fine-tuning is less effective; where errors concentrate systematically in construct-relevant classes, fine-tuning may reduce bias in downstream estimates.

4.4. Measurement Performance and Implications for Downstream Estimates

The performance results presented above assess classification quality against human expert judgment. In empirical research, however, classifications typically serve as variables in subsequent analyses, and their measurement error may affect coefficient estimates. Recent work cautions that estimates based on LLM-generated variables can shift markedly with seemingly innocuous design choices (Camuffo et al., 2026; Ludwig et al., 2025).

We illustrate these consequences with a simulation on the Demonstration Study 1 holdout sample. Consider a researcher who uses the framework’s classifications (i) as an explanatory variable, for instance to estimate whether innovative updates increase subsequent user engagement, or (ii) as an outcome variable, for instance, to examine whether a specific policy changed the share of innovative updates. For case (i), we assume that innovative updates increase engagement and simulate an engagement outcome accordingly with the coefficient set at $\beta = .50$ and noise calibrated such that innovation explains a modest share of the variance ($R^2 = .15$). We estimate the regression

with each model configuration's holdout classifications in place of the expert labels. The gap to the set $\beta = .50$ then shows how strongly each configuration's errors distort the estimate. Online Appendix H1 details the design.

In case (i), Figure 3, Panel (a) reports the estimates for all configurations evaluated in Demonstration Study 1 for which the binary innovation variable exhibits sufficient variation to estimate a coefficient. Against the set $\beta = .50$, estimates range from approximately $-.02$ to $.45$. Fine-tuned LLM configurations produce estimates closest to the known coefficient (median $\beta = .42$), followed by default LLM (.37) and PLM configurations (.35), whereas classical ML configurations show stronger deviation (.31). More generally, classification performance is highly informative about the extent of downstream bias in this setting (see Online Appendix H2 for additional analyses). Across configurations, F1-scores are strongly negatively associated with absolute coefficient bias (Spearman $\rho = -.93$), so that better classification performance translates into less deviation from the true coefficient value.

Case (ii) yields a different pattern. We use the classifications as the outcome and compare the change in the measured innovation share between an earlier and later period with the expert-labeled change of $-.06$ percentage points (Figure 3, Panel b). Individual configurations generate both positive and negative deviations from this benchmark, in some cases exceeding several percentage points. Importantly, aggregate classification performance provides little guidance about the magnitude of this distortion. Among the same configurations used in case (i), the association between F1 and absolute pre-post distortion is weak (Spearman $\rho = .14$; see Online Appendix H3 for additional analyses).

----- Insert Figure 3 about here -----

Which error type is most consequential hence depends on the research design. As in our simulated case (i), when a binary measure serves as a regressor, false positives and false negatives

both attenuate the estimate toward zero. Aggregate performance measures such as the F1-score can therefore provide a useful indication of a configuration's suitability for subsequent analyses, although the exact bias depends on the underlying error structure. As in our case (ii), when the measure serves as the outcome in, for instance, a pre-post design, stable error rates do not fabricate a change when the true outcome remains unchanged, although they may attenuate an existing change. Yet, if either error rate differs across the compared periods, the measured share can change independently of the underlying construct and we may observe an effect that does not exist with an expert-labeled outcome. A configuration therefore has to be validated not only in aggregate but also within the cells which a research design compares. Beyond both cases, when the construct resides in rare classes, false negatives concentrated in a single class (i.e., systematic misses) can make corresponding effects difficult or impossible to estimate.

When applying the framework outside a simulation, the true coefficient is unknown *ex ante*. The holdout sample can nevertheless reveal a configuration's error structure (Ludwig et al., 2025). Our simulation illustrates what this structure can imply for downstream estimates. In particular, the observed pattern of false-positive and false-negative errors provides an indication of the expected magnitude of bias and, in comparative designs, the potential risk of fabricated effects when error rates vary across design cells. A valuable complement is thus to re-estimate the focal model under alternative validated configurations, which reveals how strongly substantive conclusions depend on design choices and the range of estimates that different plausible specifications produce (Carlson and Burbano, 2026). Researchers should accordingly document the selected configuration and its aggregate validation performance, as well as the error structure observed on the holdout sample for the empirical design of interest.

4.5. Framework Relevance Under LLM Evolution

Given the rapid pace of LLM development and the prevalence of silent model updates in commercial APIs, a natural concern is whether LLM-based measurement approaches become obsolete as model versions and capabilities evolve. For instance, over the course of developing the framework and conducting the two demonstration studies, several LLM providers released new model versions (e.g., OpenAI’s GPT-5 series). Such model updates can affect instruction-following, reasoning behavior, and sensitivity to linguistic cues, which may affect performance metrics or shift decision thresholds (e.g., Chen et al., 2024).

We argue that our framework is designed to be agnostic across model versions because it anchors the measurement task in expert-validated content dimensions and a structured evaluation procedure rather than model-specific heuristics. By mapping domain experts’ cues into explicit decision criteria (and, when feasible, incorporating expert-labeled examples via fine-tuning), the framework specifies what should be measured and why, while allowing the underlying model to improve in how it recognizes and integrates these cues. In this sense, newer models are expected to maintain or even enhance the framework’s effectiveness rather than invalidate it. To assess temporal robustness, we re-ran the baseline specification of both demonstration studies on the same validation data. Overall, we found strong robustness in performance metrics and no differences in the resulting patterns or interpretations. Full details are provided in Online Appendix I.

5. Conditions for Applying the Framework

Abstracting from the features that the two demonstration studies share, we identify three conditions under which the framework is applicable. First, the construct must manifest in the text itself. The text has to contain the cues a domain expert would rely on, such that an expert reading only the focal text could reach a judgment. Second, a small number of domain experts must be available to

articulate and validate these cues and to label a holdout sample. Third, these expert judgments of the construct must be sufficiently reliable to serve as ground truth (Einhorn, 1974).

These conditions also clarify which empirical settings the framework can be applied to. The availability of text data is necessary but has rarely been the binding constraint as text documenting innovative activity has become widely available (Bellstam et al., 2021). The main constraint was the need for expert-assessed labels at scale. Our framework relaxes this constraint considerably. The contexts that become more tractable are therefore those with construct-bearing text but no institutionalized expert evaluation of the kind that, for instance, patents provide. Examples include idea submissions on crowdsourcing platforms (Zaggl et al., 2023), crowdfunding pitches (Mollick and Nanda, 2016), and startup value propositions (Guzman and Li, 2023).

Within these conditions, the framework is not equally valuable for every classification task. Its comparative advantage lies in tasks where the construct is latent and judgment-intensive, the language describing it is heterogeneous and lacks standardized terminology, and expert-labeled training data is scarce or costly to produce. Where the construct resides in surface features of text, established alternatives remain competitive at lower cost. For instance, dictionary methods perform well when terminology is unambiguous and broadly shared (Miric et al., 2023a), particularly where fields maintain curated ontologies that encode expert judgment in the vocabulary itself (Petersen, 2022), and classical supervised ML performs well when large, labeled datasets exist and linguistic variation is more limited (Miric et al., 2023a). In such settings the framework works but is less competitive on cost.

Constructs that experts evaluate from non-linguistic evidence lie outside our demonstrations altogether. While current LLMs accept multimodal input, extending the framework’s expert-cue validation to such data remains a task for future research. Finally, strategic misrepresentation constrains any text-based method, not LLM-based measurement specifically.

Mild promotional language can be absorbed by the coding scheme, as the marketing and branding dimension in Demonstration Study 1 illustrates, but systematic misrepresentation that leaves no trace within the text cannot be corrected by a classifier. This concern intensifies once a measure itself becomes a target of strategic behavior, and it compounds as linguistic conventions drift over time, so that cue sets and fine-tuned models require periodic revalidation.

Relatedly, the framework’s validity is local to the corpus on which it is established. The training corpora of LLMs overrepresent recent, English-language, and Western material, and model outputs may by default align most closely with the judgments of the corresponding populations (Bender et al., 2021; Tao et al., 2024).¹⁰ Expert-validated cues and fine-tuning anchor the framework’s evaluations to the designated evaluators, yet applying these cues still draws on the model’s pre-trained language understanding. Performance can therefore degrade with temporal, linguistic, or cultural distance from this distribution, and validation on a contemporary corpus does not certify applications to distant ones.¹¹ Such applications remain possible in principle, but they require establishing an expert-validated holdout for the target period and domain, which in turn presupposes experts for that period and domain.

The framework relies on a ground truth. Its performance criterion is the approximation of expert judgment. For genuinely contested constructs on which domain experts do not converge there is nothing to approximate and no benchmark against which validation is meaningful (Einhorn,

¹⁰ This concern relates to the broader notion of WEIRD bias, which describes how evidence and data disproportionately drawn from Western, Educated, Industrialized, Rich, and Democratic populations may not generalize to populations outside these contexts (Henrich et al., 2010). In the context of LLMs, the overrepresentation of Western and English-language material in training data raises an analogous concern about whose language use, knowledge, and judgments are most strongly represented.

¹¹ Historical applications illustrate two failure modes. A present-day LLM assessing the innovativeness of the incandescent lightbulb knows its eventual significance, so that ex ante novelty is conflated with ex post importance. At the same time, nineteenth-century language and genre conventions are likely sparse in training data, which degrades comprehension of ordinary period documents (Lazaridou et al., 2021).

1974). The appropriate response in such cases is conceptual work on the construct or the aggregation of judgments across evaluator groups.

Table J13 in Online Appendix J summarizes these conditions as questions that researchers can apply to their own settings before adopting the framework.

6. Discussion

6.1. Contributions to Innovation Management Research

The measurement of innovation-related phenomena is often constrained by the challenges inherent in identifying and assessing contextual information (Boudreau et al., 2016). In many cases, such evaluation relies on expertise-based judgment, yet access to qualified experts is often limited. Moreover, analyzing large datasets is impeded by the inherent cognitive limits of human evaluators. As the volume of the data increases, the cognitive load on experts can lead to greater reliance on heuristics and biases, which may impair their ability to evaluate effectively (Tversky and Kahneman, 1974). Empirical research in innovation has therefore mainly been conducted in contexts where reliable measures are readily available. Our research addresses these challenges by developing an LLM-based framework. Our framework is based on the premise that certain contextual cues are highly indicative of expert judgment (Einhorn, 1974). We apply qualitative methods and collaborate with experts to identify these cues, assess their relevance, and integrate these insights systematically into prompt design and, where useful, fine-tuning for the LLM. Because expert judgments may draw on learned, tacit conceptions of similarity that are difficult to articulate explicitly (Nightingale, 1998), this process can also help make otherwise implicit construct boundaries and evaluation criteria more explicit. A key strength of our framework is that it reduces the need for repeated human expert evaluation at scale while still achieving comparable evaluations. The scalability enabled by this approach simplifies empirical innovation research

across a broader range of contexts, and also provides opportunities for additional robustness checks and analytical perspectives.

We successfully illustrate the applicability and utility of our framework in two demonstration studies conducted in different innovation contexts. Although the validation exercises are specific to innovation in digital platform ecosystems, they capture two central concepts in innovation research: the outputs of innovative activity and the generation of ideas, holding promise for extension to other innovation contexts as well. Our results further show that fine-tuning LLMs is not a prerequisite for such applications of our framework, but a design choice whose value depends on the structure of the remaining classification errors and the requirements of the intended empirical analysis. At the same time, even the best-performing models in our demonstration studies still misclassify roughly one in ten (positive) instances (precision $\approx .90$). Any high-stakes deployment should retain a human-in-the-loop review and undertake a careful error-impact analysis tailored to the application context.

Furthermore, the application of our framework makes specific contributions to the research areas of these two demonstration studies by enhancing existing metrics. First, in studying innovation in software application development in digital platform ecosystems (e.g., Agarwal and Kapoor, 2023; Kircher and Foerderer, 2024; Wen and Zhu, 2019), we improve measurement performance, assessed by F1-scores, of the innovation variable by 39% relative to the best-performing measure from previous studies (see Table 2), and additionally enable a more fine-grained differentiation of update content. Second, in identifying innovation-related user-generated ideas in product reviews (e.g., Timoshenko and Hauser, 2019; Zhang et al., 2021), our framework allows for a more detailed, multi-class screening of ideas.

The implications of these improvements are twofold. First, they introduce new levels of analysis and promising directions for future research. Specifically, the capability to examine update

release notes and product reviews at such detailed levels allows researchers to accurately trace how user innovations are adopted and integrated over time. Consequently, researchers can gain deeper insights into when and how user-suggested ideas become part of products and assess the extent to which these incorporations contribute to outcomes like sustained competitive advantage. For instance, products with larger and more engaged user communities may disproportionately benefit from a greater influx of user-generated ideas, potentially accelerating innovation cycles or enhancing adaptability to market demands. Moreover, such detailed analyses facilitate better evaluation of the impact of external events that influence development and user behavior. For example, regulators and platform owners of digital platforms are increasingly emphasizing user privacy by introducing privacy-preserving regulations. The refined metrics demonstrated in our studies allow researchers to quantify the effects of these regulations, such as their influence on application development practices and user perceptions regarding the regulations and corresponding implementations by developers. Second, the improvements may increase the validity of empirical findings and conclusions such as in studies on complementor innovation on app platforms, by reducing potential systematic measurement errors that could result in substantial bias in coefficient estimates (e.g., deHaan et al., 2025; Millimet and Parmeter, 2022). Our additional analyses further show that the consequences of such errors depend on the empirical design and the underlying error structure, highlighting the importance of validating measures with the intended downstream application in mind. Future research, through replication studies and further exploration, is needed to fully assess the implications of these enhancements.

6.2. The Role of the Expert Evaluator

Our framework is centered on relevant domain expertise being available for cue identification and weighting. The underlying assumption is that bounded rationality and imperfect judgment in evaluation tasks affect all possible evaluators, while those whose expertise aligns most closely with

the evaluation context exhibit lower uncertainty in their judgments (Böttcher and Klingebiel, 2025; Boudreau et al., 2016; Csaszar and Eggers, 2013). This is because domain experts are able to disproportionately identify and interpret more relevant information compared to less specialized evaluators (Ericsson and Smith, 1991; Johnson et al., 1982; Kahneman et al., 1982).

However, we also recognize that identifying the most suitable expert *ex ante* is challenging. There exists a risk of cognitive entrenchment, where experts may be less inclined to identify and integrate new information that challenges their existing views (Almandoz and Tilcsik, 2016; Dane, 2010; Furr et al., 2012). As a result, they may inadvertently extrapolate from their established knowledge into domains where their expertise is less applicable. This misalignment can degrade evaluation performance, even to a point that their judgment becomes inferior to that of generalist evaluators (Böttcher and Klingebiel, 2025; Csaszar and Eggers, 2013).

Although the framework is designed to enforce robust evaluation processes, it is not immune to the risk of replicating or even reinforcing such evaluator bias. These risks are partially mitigated by the framework’s data-centric approach, where the extraction of the data structure is carried out by the researcher, and domain experts are engaged primarily for validation and adjustment purposes. Still, there remains a potential for bias, particularly when fine-tuning with biased data is not carefully managed or when experts and the model are subject to the same bias. Where expert coders hold, for instance, a WEIRD-normed conception of what constitutes innovation (Henrich et al., 2010), both deviate in the same direction, and the error would survive holdout validation undetected.

Given these limitations, alternative evaluation scenarios are worth considering. In contexts where identifying the most relevant domain experts is challenging, integrating insights, for instance, from crowds (e.g., Mollick and Nanda, 2016; Wimbauer et al., 2019) or evaluator panels (e.g., Criscuolo et al., 2017) may yield an increase in evaluation performance. The primary

advantage of the proposed framework lies in its data-centric approach, where expert validation is confined to the data structure. This design makes the framework agnostic to the type and format of expert input. Future research could explore how integrating different evaluator groups and formats (e.g., ranking, voting, aggregation) in our framework impacts performance, providing deeper insights into the trade-offs and benefits of each approach. This exploration is particularly relevant as evaluation methods for LLMs continue to evolve, with hybrid strategies combining human and automated evaluations emerging as a promising direction.

6.3. Ethical and Practical Considerations

LLMs are often considered “black boxes” due to their complexity and opacity. They are pre-trained on exceptionally large datasets and build on advanced deep learning architectures with billions of parameters. Despite their impressive performance, our current understanding of their internal mechanisms is limited (Bommasani et al., 2021), and independently constructing comparable models at similar performance scales is typically beyond the capabilities of individual researchers or smaller institutions. Moreover, the processes by which LLMs generate outputs and the biases introduced during pre-training, particularly with datasets undisclosed, are not yet fully understood. Variations between different models, or even between different versions of the same model, are both inevitable and expected. When applying the framework, it is therefore essential to systematically validate the outputs against a holdout sample or established ground truth to accurately assess performance and detect potential systematic measurement errors.

Practically, using LLMs involves navigating operational considerations related to both proprietary and open-weight models. Proprietary models, typically accessed via APIs, often provide minimal and non-specific information about their internal configurations and are subject to potential modifications or discontinuation by providers. Therefore, detailed documentation of the exact model version used and transparent reporting of all design decisions is essential for

ensuring reproducibility. While proprietary models still offer advantages in efficiency and effectiveness, API costs, despite decreasing over time, may accumulate significantly when working with large datasets. Open-weight models, on the other hand, reduce the risks associated with proprietary models, such as unexpected accessibility changes or alterations in performance. However, using medium or large-sized open-weight models locally demands substantial hardware infrastructure. Alternatively, cloud-based deployment can also incur significant scaling costs. Despite platforms like Hugging Face considerably reducing entry barriers, substantial technical expertise remains essential. A practical middle ground is to use proprietary, high-performing models as teachers to generate labels for training smaller, open or locally deployable student models, thereby reducing recurring API costs and dependence on provider-side changes while retaining much of the predictive quality.

Given the rapid advancements in LLM technology, questions may arise regarding the long-term relevance of our proposed framework. The proposed framework is designed to be model-agnostic, allowing it to leverage ongoing advancements in LLM capabilities and adapt to future improvements in the field. By focusing on mapping expert cues into the evaluation process, the framework provides a robust and adaptable foundation for scaling the assessment of innovation-related constructs in diverse scenarios.

References

- Abernathy, W.J., Clark, K.B., 1985. Innovation: Mapping the winds of creative destruction. *Research Policy* 14, 3–22. [https://doi.org/10.1016/0048-7333\(85\)90021-6](https://doi.org/10.1016/0048-7333(85)90021-6)
- Abrahamson, E., 1991. Managerial Fads and Fashions: The Diffusion and Rejection of Innovations. *The Academy of Management Review* 16, 586–612. <https://doi.org/10.2307/258919>
- Agarwal, S., Kapoor, R., 2023. Value Creation Tradeoff in Business Ecosystems: Leveraging Complementarities While Managing Interdependencies. *Organization Science* 34, 1216–1242. <https://doi.org/10.1287/orsc.2022.1615>
- Almandoz, J., Tilcsik, A., 2016. When Experts Become Liabilities: Domain Experts on Boards and Organizational Failure. *AMJ* 59, 1124–1149. <https://doi.org/10.5465/amj.2013.1211>

- Artstein, R., Poesio, M., 2008. Inter-Coder Agreement for Computational Linguistics. *Computational Linguistics* 34, 555–596. <https://doi.org/10.1162/coli.07-034-R2>
- Bellstam, G., Bhagat, S., Cookson, J.A., 2021. A Text-Based Analysis of Corporate Innovation. *Management Science* 67, 4004–4031. <https://doi.org/10.1287/mnsc.2020.3682>
- Bender, E.M., Gebru, T., McMillan-Major, A., Shmitchell, S., 2021. On the Dangers of Stochastic Parrots: Can Language Models Be Too Big?, in: *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, ACM, 610–623. <https://doi.org/10.1145/3442188.3445922>
- Bommasani, R., Hudson, D.A., Adeli, E., Altman, R., Arora, S., [...], K., Liang, P., 2021. On the Opportunities and Risks of Foundation Models. <https://doi.org/10.48550/ARXIV.2108.07258>
- Böttcher, L., Klingebiel, R., 2025. Organizational Selection of Innovation. *Organization Science* 36, 387–410. <https://doi.org/10.1287/orsc.2023.17357>
- Boudreau, K.J., 2012. Let a Thousand Flowers Bloom? An Early Look at Large Numbers of Software App Developers and Patterns of Innovation. *Organization Science* 23, 1409–1427. <https://doi.org/10.1287/orsc.1110.0678>
- Boudreau, K.J., Guinan, E.C., Lakhani, K.R., Riedl, C., 2016. Looking Across and Looking Beyond the Knowledge Frontier: Intellectual Distance, Novelty, and Resource Allocation in Science. *Management Science* 62, 2765–2783. <https://doi.org/10.1287/mnsc.2015.2285>
- Boudreau, K.J., Lacetera, N., Lakhani, K.R., 2011. Incentives and Problem Uncertainty in Innovation Contests: An Empirical Analysis. *Management Science* 57, 843–863. <https://doi.org/10.1287/mnsc.1110.1322>
- Boussiou, L., Lane, J.N., Zhang, M., Jacimovic, V., Lakhani, K.R., 2024. The Crowdless Future? Generative AI and Creative Problem-Solving. *Organization Science* 35, 1589–1607. <https://doi.org/10.1287/orsc.2023.18430>
- Brown, T.B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., [...], Amodei, D., 2020. Language Models are Few-Shot Learners. <https://doi.org/10.48550/ARXIV.2005.14165>
- Brynjolfsson, E., Enríquez, J.R., Kazinnik, S., Nguyen, D., 2026. Augmenting Survey Data with Generative AI: An Application to Economic Research. <https://doi.org/10.2139/ssrn.6343598>
- Camuffo, A., Gambardella, A., Kazemi, S., Malachowski, J., Pandey, A., 2026. Variance-Aware LLM Annotation for Strategy Research: Sources, Diagnostics, and a Protocol for Reliable Measurement. <https://doi.org/10.48550/arXiv.2601.02370>
- Carletta, J., 1996. Assessing Agreement on Classification Tasks: The Kappa Statistic. *Computational Linguistics* 22, 249–254.
- Carlson, N.A., Burbano, V., 2026. The use of LLMs to annotate data in management research: Foundational guidelines and warnings. *Strategic Management Journal* 47, 699–725. <https://doi.org/10.1002/smj.70023>
- Chase, W.G., Simon, H.A., 1973. Perception in chess. *Cognitive Psychology* 4, 55–81. [https://doi.org/10.1016/0010-0285\(73\)90004-2](https://doi.org/10.1016/0010-0285(73)90004-2)
- Chen, L., Zaharia, M., Zou, J., 2024. How Is ChatGPT’s Behavior Changing Over Time? *Harvard Data Science Review* 6. <https://doi.org/10.1162/99608f92.5317da47>
- Cheng, X., Mayya, R., Sedoc, J., 2024. From Human Annotation to LLMs: SILICON Annotation Workflow for Management Research. <https://doi.org/10.48550/ARXIV.2412.14461>
- Chesbrough, H.W., 2003. *Open innovation: the new imperative for creating and profiting from technology*. Harvard Business School Press, Boston, MA.

- Choudhury, P., Wang, D., Carlson, N.A., Khanna, T., 2019. Machine learning approaches to facial and text analysis: Discovering CEO oral communication styles. *Strategic Management Journal* 40, 1705–1732. <https://doi.org/10.1002/smj.3067>
- Comino, S., Manenti, F.M., Mariuzzo, F., 2019. Updates management in mobile applications: iTunes versus Google Play. *Journal of Economics & Management Strategy* 28, 392–419. <https://doi.org/10.1111/jems.12288>
- Criscuolo, P., Dahlander, L., Grohsjean, T., Salter, A., 2017. Evaluating Novelty: The Role of Panels in the Selection of R&D Projects. *Academy of Management Journal* 60, 433–460. <https://doi.org/10.5465/amj.2014.0861>
- Crossan, M.M., Apaydin, M., 2010. A Multi-Dimensional Framework of Organizational Innovation: A Systematic Review of the Literature. *Journal of Management Studies* 47, 1154–1191. <https://doi.org/10.1111/j.1467-6486.2009.00880.x>
- Csaszar, F.A., Eggers, J.P., 2013. Organizational Decision Making: An Information Aggregation View. *Management Science* 59, 2257–2277. <https://doi.org/10.1287/mnsc.1120.1698>
- Dane, E., 2010. Reconsidering the Trade-off Between Expertise and Flexibility: a Cognitive Entrenchment Perspective. *Academy of Management Review* 35, 579–603. <https://doi.org/10.5465/amr.35.4.zok579>
- Dane, E., Pratt, M.G., 2007. Exploring Intuition and its Role in Managerial Decision Making. *Academy of Management Review* 32, 33–54. <https://doi.org/10.5465/amr.2007.23463682>
- de Kok, T., 2025. ChatGPT for Textual Analysis? How to Use Generative LLMs in Accounting Research. *Management Science* 71, 7888–7906. <https://doi.org/10.1287/mnsc.2023.03253>
- deHaan, E., Lawrence, A., Litjens, R., 2025. Measuring Investor Attention Using Google Search. *Management Science* 71, 6275–6297. <https://doi.org/10.1287/mnsc.2022.02174>
- DeVellis, R.F., Thorpe, C.T., 2022. Scale development: theory and applications, 5th ed. SAGE, Thousand Oaks, CA.
- Doshi, A.R., Bell, J.J., Mirzayev, E., Vanneste, B.S., 2025. Generative artificial intelligence and evaluating strategic decisions. *Strategic Management Journal* 46, 583–610. <https://doi.org/10.1002/smj.3677>
- Einhorn, H.J., 1974. Expert judgment: Some necessary conditions and an example. *Journal of Applied Psychology* 59, 562–571. <https://doi.org/10.1037/h0037164>
- Ericsson, K.A., Smith, J. (Eds.), 1991. *Toward a general theory of expertise: prospects and limits*. Cambridge University Press, Cambridge.
- Fleming, L., 2001. Recombinant Uncertainty in Technological Search. *Management Science* 47, 117–132. <https://doi.org/10.1287/mnsc.47.1.117.10671>
- Foerderer, J., Kude, T., Mithas, S., Heinzl, A., 2018. Does Platform Owner’s Entry Crowd Out Innovation? Evidence from Google Photos. *Information Systems Research* 29, 444–460. <https://doi.org/10.1287/isre.2018.0787>
- Furr, N.R., Cavarretta, F., Garg, S., 2012. Who Changes Course? The Role of Domain Knowledge and Novel Framing in Making Technology Changes. *Strategic Entrepreneurship Journal* 6, 236–256. <https://doi.org/10.1002/sej.1137>
- Fyffe, S., Lee, P., Kaplan, S., 2024. “Transforming” Personality Scale Development: Illustrating the Potential of State-of-the-Art Natural Language Processing. *Organizational Research Methods* 27, 265–300. <https://doi.org/10.1177/10944281231155771>
- Gao, T., Fisch, A., Chen, D., 2020. Making Pre-trained Language Models Better Few-shot Learners. <https://doi.org/10.48550/ARXIV.2012.15723>

- Garcia, R., Calantone, R., 2002. A critical look at technological innovation typology and innovativeness terminology: a literature review. *Journal of Product Innovation Management* 19, 110–132. <https://doi.org/10.1111/1540-5885.1920110>
- Gault, F., 2018. Defining and measuring innovation in all sectors of the economy. *Research Policy* 47, 617–622. <https://doi.org/10.1016/j.respol.2018.01.007>
- Gentzkow, M., Kelly, B., Taddy, M., 2019. Text as Data. *Journal of Economic Literature* 57, 535–574. <https://doi.org/10.1257/jel.20181020>
- Gioia, D.A., Corley, K.G., Hamilton, A.L., 2013. Seeking Qualitative Rigor in Inductive Research: Notes on the Gioia Methodology. *Organizational Research Methods* 16, 15–31. <https://doi.org/10.1177/1094428112452151>
- Goodfellow, I., Bengio, Y., Courville, A., 2016. *Deep learning*, Adaptive computation and machine learning. MIT press, Cambridge, MA.
- Griliches, Z., 1990. Patent Statistics as Economic Indicators: A Survey. *Journal of Economic Literature* 28, 1661–1707.
- Gunathilaka, S., de Silva, N., 2025. Automatic Analysis of App Reviews Using LLMs, in: *Proceedings of the 17th International Conference on Agents and Artificial Intelligence*, 828–839. <https://doi.org/10.5220/0013375600003890>
- Guzman, J., Li, A., 2023. Measuring Founding Strategy. *Management Science* 69, 101–118. <https://doi.org/10.1287/mnsc.2022.4369>
- Haeussler, C., Assmus, A., 2021. Bridging the gap between invention and innovation: Increasing success rates in publicly and industry-funded clinical trials. *Research Policy* 50, 104155. <https://doi.org/10.1016/j.respol.2020.104155>
- Henderson, R.M., Clark, K.B., 1990. Architectural Innovation: The Reconfiguration of Existing Product Technologies and the Failure of Established Firms. *Administrative Science Quarterly* 35, 9–30. <https://doi.org/10.2307/2393549>
- Henrich, J., Heine, S.J., Norenzayan, A., 2010. The weirdest people in the world? *Behavioral and Brain Sciences* 33, 61–83. <https://doi.org/10.1017/S0140525X0999152X>
- Hirschberg, J., Manning, C.D., 2015. Advances in natural language processing. *Science* 349, 261–266. <https://doi.org/10.1126/science.aaa8685>
- Huang, L., Pearce, J.L., 2015. Managing the Unknowable: The Effectiveness of Early-stage Investor Gut Feel in Entrepreneurial Investment Decisions. *Administrative Science Quarterly* 60, 634–670. <https://doi.org/10.1177/0001839215597270>
- James, T., Qiao, Z., Shen, W., Wang, G.A., Fan, W., 2023. Competing for Temporary Advantage in a Hypercompetitive Mobile App Market. *MIS Quarterly* 47, 1177–1212. <https://doi.org/10.25300/MISQ/2022/15079>
- Jeppesen, L.B., Lakhani, K.R., 2010. Marginality and Problem-Solving Effectiveness in Broadcast Search. *Organization Science* 21, 1016–1033. <https://doi.org/10.1287/orsc.1090.0491>
- Johnson, P.E., Hassebrock, F., Durán, A.S., Moller, J.H., 1982. Multimethod study of clinical judgment. *Organizational Behavior and Human Performance* 30, 201–230. [https://doi.org/10.1016/0030-5073\(82\)90218-5](https://doi.org/10.1016/0030-5073(82)90218-5)
- Jones, C.R., Bergen, B.K., 2024. People cannot distinguish GPT-4 from a human in a Turing test. <https://doi.org/10.48550/ARXIV.2405.08007>
- Just, J., Ströhle, T., Füller, J., Hutter, K., 2024. AI-based novelty detection in crowdsourced idea spaces. *Innovation: Organization & Management* 26, 359–386. <https://doi.org/10.1080/14479338.2023.2215740>
- Kahneman, D., Slovic, P., Tversky, A., 1982. *Judgment under uncertainty: heuristics and biases*. Cambridge University Press, Cambridge.

- Khelifi, G., Jenhani, I., Messaoud, M.B., Mkaouer, M.W., 2024. Multi-label Classification of Mobile Application User Reviews Using Neural Language Models, in: Bouraoui, Z., Vesic, S. (Eds.), *Symbolic and Quantitative Approaches to Reasoning with Uncertainty*, Lecture Notes in Computer Science. Springer Nature Switzerland, Cham, pp. 417–426. https://doi.org/10.1007/978-3-031-45608-4_31
- Kim, M., DellaPosta, D., 2022. The Fickle Crowd: Reinforcement and Contradiction of Quality Evaluations in Cultural Markets. *Organization Science* 33, 2496–2518. <https://doi.org/10.1287/orsc.2021.1556>
- Kircher, T., Foerderer, J., 2024. Ban Targeted Advertising? An Empirical Investigation of the Consequences for App Development. *Management Science* 70, 1070–1092. <https://doi.org/10.1287/mnsc.2023.4726>
- Korinek, A., 2023. Generative AI for Economic Research: Use Cases and Implications for Economists. *Journal of Economic Literature* 61, 1281–1317. <https://doi.org/10.1257/jel.20231736>
- Krieger, J.L., 2021. Trials and Terminations: Learning from Competitors' R&D Failures. *Management Science* 67, 5525–5548. <https://doi.org/10.1287/mnsc.2020.3775>
- Krippendorff, K., 2019. *Content Analysis: An Introduction to Its Methodology*. SAGE, Thousand Oaks, CA. <https://doi.org/10.4135/9781071878781>
- Landis, J.R., Koch, G.G., 1977. The Measurement of Observer Agreement for Categorical Data. *Biometrics* 33, 159–174. <https://doi.org/10.2307/2529310>
- Lane, J.N., Boussioux, L., Ayoubi, C., Chen, Y.H., Lin, C., Spens, R., Wagh, P., Wang, P.-H., 2024. The Narrative AI Advantage? A Field Experiment on Generative AI-Augmented Evaluations of Early-Stage Innovations. Harvard Business School Working Paper, No. 25-001.
- Lane, J.N., Teplitskiy, M., Gray, G., Ranu, H., Menietti, M., Guinan, E.C., Lakhani, K.R., 2022. Conservatism Gets Funded? A Field Experiment on the Role of Negative Information in Novel Project Evaluation. *Management Science* 68, 4478–4495. <https://doi.org/10.1287/mnsc.2021.4107>
- Lazaridou, A., Kuncoro, A., Gribovskaya, E., Agrawal, D., Liska, A., Terzi, T., Gimenez, M., d'Autume, C. de M., Kocisky, T., Ruder, S., Yogatama, D., Cao, K., Young, S., Blunsom, P., 2021. Mind the Gap: Assessing Temporal Generalization in Neural Language Models, in: *Proceedings of the 35th International Conference on Neural Information Processing Systems*. pp. 29348–29363.
- Lee, G., Raghu, T.S., 2014. Determinants of Mobile Apps' Success: Evidence from the App Store Market. *Journal of Management Information Systems* 31, 133–170. <https://doi.org/10.2753/MIS0742-1222310206>
- Leyden, B.T., 2022. There's an App (Update) for That: Understanding Product Updating Under Digitization.
- Li, D., 2017. Expertise versus Bias in Evaluation: Evidence from the NIH. *American Economic Journal: Applied Economics* 9, 60–92. <https://doi.org/10.1257/app.20150421>
- Liu, P., Yuan, W., Fu, J., Jiang, Z., Hayashi, H., Neubig, G., 2023. Pre-train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing. *ACM Comput. Surv.* 55, 1–35. <https://doi.org/10.1145/3560815>
- Loughran, T., McDonald, B., 2011. When Is a Liability Not a Liability? Textual Analysis, Dictionaries, and 10-Ks. *The Journal of Finance* 66, 35–65. <https://doi.org/10.1111/j.1540-6261.2010.01625.x>

- Ludwig, J., Mullainathan, S., Rambachan, A., 2025. Large Language Models: An Applied Econometric Framework (No. w33344). National Bureau of Economic Research, Cambridge, MA. <https://doi.org/10.3386/w33344>
- Millimet, D.L., Parmeter, C.F., 2022. Accounting for Skewed or One-Sided Measurement Error in the Dependent Variable. *Political Analysis* 30, 66–88. <https://doi.org/10.1017/pan.2020.45>
- Miric, M., Jia, N., Huang, K.G., 2023a. Using supervised machine learning for large-scale classification in management research: The case for identifying artificial intelligence patents. *Strategic Management Journal* 44, 491–519. <https://doi.org/10.1002/smj.3441>
- Miric, M., Ozalp, H., Yilmaz, E.D., 2023b. Trade-offs to using standardized tools: Innovation enablers or creativity constraints? *Strategic Management Journal* 44, 909–942. <https://doi.org/10.1002/smj.3457>
- Mitchell, M., Krakauer, D.C., 2023. The debate over understanding in AI's large language models. *Proceedings of the National Academy of Sciences* 120, e2215907120. <https://doi.org/10.1073/pnas.2215907120>
- Mollick, E., Nanda, R., 2016. Wisdom or Madness? Comparing Crowds with Expert Evaluation in Funding the Arts. *Management Science* 62, 1533–1553. <https://doi.org/10.1287/mnsc.2015.2207>
- Moser, P., 2012. Innovation without Patents: Evidence from World's Fairs. *The Journal of Law and Economics* 55, 43–74. <https://doi.org/10.1086/663631>
- Nightingale, P., 1998. A cognitive model of innovation. *Research Policy* 27, 689–709. [https://doi.org/10.1016/S0048-7333\(98\)00078-X](https://doi.org/10.1016/S0048-7333(98)00078-X)
- OpenAI, 2023. GPT-4 Technical Report. <https://doi.org/10.48550/ARXIV.2303.08774>
- Parker, G., Van Alstyne, M., 2018. Innovation, Openness, and Platform Control. *Management Science* 64, 3015–3032. <https://doi.org/10.1287/mnsc.2017.2757>
- Peeperkorn, M., Kouwenhoven, T., Brown, D., Jordanous, A., 2024. Is Temperature the Creativity Parameter of Large Language Models? <https://doi.org/10.48550/ARXIV.2405.00492>
- Petersen, A.M., 2022. Evolution of biomedical innovation quantified via billions of distinct article-level MeSH keyword combinations. *Advances in Complex Systems* 25, 2150016. <https://doi.org/10.1142/S0219525921500168>
- Piezunka, H., Dahlander, L., 2015. Distant Search, Narrow Attention: How Crowding Alters Organizations' Filtering of Suggestions in Crowdsourcing. *Academy of Management Journal* 58, 856–880. <https://doi.org/10.5465/amj.2012.0458>
- Rosenkopf, L., McGrath, P., 2011. Advancing the Conceptualization and Operationalization of Novelty in Organizational Research. *Organization Science* 22, 1297–1311. <https://doi.org/10.1287/orsc.1100.0637>
- Samuelson, W., Zeckhauser, R., 1988. Status quo bias in decision making. *Journal of Risk and Uncertainty* 1, 7–59. <https://doi.org/10.1007/BF00055564>
- Schumpeter, J.A., 1934. *The Theory of Economic Development. An Inquiry into Profits, Capital, Credit, Interest, and the Business Cycle*. Harvard University Press, Cambridge, MA.
- Shanteau, J., 1992. Competence in experts: The role of task characteristics. *Organizational Behavior and Human Decision Processes* 53, 252–266. [https://doi.org/10.1016/0749-5978\(92\)90064-E](https://doi.org/10.1016/0749-5978(92)90064-E)
- Smith, K., 2005. Measuring Innovation, in: Fagerberg, J., Mowery, D.C., Nelson, R.R. (Eds.), *The Oxford Handbook of Innovation*. Oxford University Press, Oxford, pp. 148–177.
- Strauss, A.L., Corbin, J.M., 1998. *Basics of qualitative research: techniques and procedures for developing grounded theory*, 2nd ed. SAGE, Thousand Oaks, CA.

- Sukhov, A., Sihvonen, A., Netz, J., Magnusson, P., Olsson, L.E., 2021. How experts screen ideas: The complex interplay of intuition, analysis and sensemaking. *Journal of Product Innovation Management* 38, 248–270. <https://doi.org/10.1111/jpim.12559>
- Tao, Y., Viberg, O., Baker, R.S., Kizilcec, R.F., 2024. Cultural bias and cultural alignment of large language models. *PNAS Nexus* 3, pgae346. <https://doi.org/10.1093/pnasnexus/pgae346>
- Timoshenko, A., Hauser, J.R., 2019. Identifying Customer Needs from User-Generated Content. *Marketing Science* 38, 1–20. <https://doi.org/10.1287/mksc.2018.1123>
- Tversky, A., Kahneman, D., 1974. Judgment under Uncertainty: Heuristics and Biases: Biases in judgments reveal some heuristics of thinking under uncertainty. *Science* 185, 1124–1131. <https://doi.org/10.1126/science.185.4157.1124>
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I., 2017. Attention Is All You Need, in: *Advances in Neural Information Processing Systems*. Curran Associates, Inc.
- Verhoeven, D., Bakker, J., Veugelers, R., 2016. Measuring technological novelty with patent-based indicators. *Research Policy* 45, 707–723. <https://doi.org/10.1016/j.respol.2015.11.010>
- Von Hippel, E., 1978. A customer-active paradigm for industrial product idea generation. *Research Policy* 7, 240–266. [https://doi.org/10.1016/0048-7333\(78\)90019-7](https://doi.org/10.1016/0048-7333(78)90019-7)
- Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E., Le, Q.V., Zhou, D., 2022. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. *Advances in Neural Information Processing Systems* 35, 24824–24837.
- Wen, W., Zhu, F., 2019. Threat of platform-owner entry and complementor responses: Evidence from the mobile app market. *Strategic Management Journal* 40, 1336–1367. <https://doi.org/10.1002/smj.3031>
- Wimbauer, L., Figge, P., Haeussler, C., 2019. Distant Search, but Local Implementation? Using the Crowd's Evaluation to Overcome Organizational Limitations in the Selection of Crowdsourced Ideas, in: *ICIS 2019 Proceedings*.
- Zaggl, M.A., Malhotra, A., Alexy, O., Majchrzak, A., 2023. Governing crowdsourcing for unconstrained innovation problems. *Strategic Management Journal* 44, 2783–2817. <https://doi.org/10.1002/smj.3505>
- Zhang, M., Fan, B., Zhang, N., Wang, W., Fan, W., 2021. Mining product innovation ideas from online reviews. *Information Processing & Management* 58, 102389. <https://doi.org/10.1016/j.ipm.2020.102389>
- Zittrain, J., 2006. The Generative Internet. *Harvard Law Review* 119, 1974–2040.

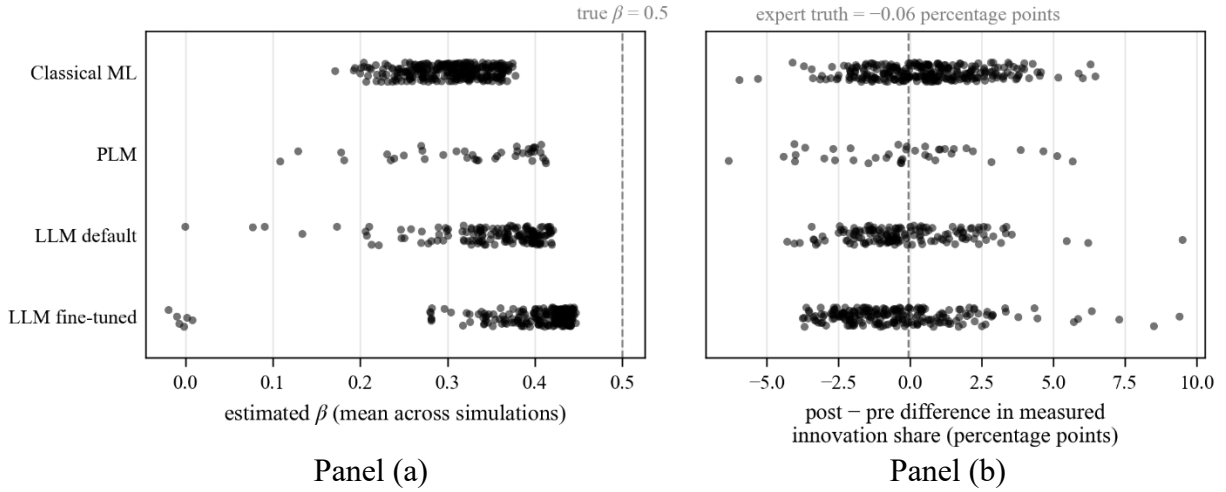
Figure 1 System prompt and prompt breakdown for software application update classification (Study 1)

<p>ONLY provide the category number (1-7) in response. Determine the category for the following app update text. If multiple categories seem applicable, always choose the lowest category number (1<2<3<4<5<6<7):</p> <ol style="list-style-type: none"> (1) Novel features – Introducing or enhancing significant functionalities that modify user experience (2) Content extensions – Adding or expanding curated or user-facing content (3) Platform and device support – Enabling compatibility with new OS versions, development kits, device types, or hardware features (4) Specific fixes of bugs – Addressing distinct known issues (5) Privacy & Security – Strengthening user data protection, permissions, or security protocols (6) General Improvements – Tweaks addressing undisclosed bug fixes, performance improvements, minor changes, or UI/UX polish (7) Marketing & Branding – Changes purely about promotional or visual branding assets <p>App Update: [RELEASE NOTE]</p>	<p>Section 1 Output definition and task description according to measurement objective</p> <p>Section 2 Expert validated data structure</p> <p>Section 3 Text data for evaluation</p>
---	--

Figure 2 System prompt and prompt breakdown for product review classification (Study 2)

<p>ONLY provide a number (0-8) in response. Categorize the following app review text by assigning the most fitting category/categories out of the following nine categories. If the text contains elements from multiple categories, provide the categories separated by ;</p> <ol style="list-style-type: none"> (0) User Opinion without specific reports, issues, suggestions – i.e., review only about good/bad perception of the app but nothing else (1) Reports of bugs, errors, or bad quality issues – i.e., something does not work in the app or is of bad quality (2) Issues of the app's monetization model – i.e., complaints or issues of how the app monetizes content (3) Suggestions for new features or content or revival of removed features (4) Customer support issues – i.e., problems or complaints regarding customer support (5) Performance issues – i.e., the app needs too much space, is too slow or similar (6) Security concerns – i.e., user is concerned about their data or privacy (7) Ethical concerns – i.e., user is concerned about practices in the app, fairness, discrimination (8) Community related issues – user asks openly for help (no feedback at customer support) <p>Review text: [REVIEW TEXT]</p>	<p>Section 1 Output definition and task description according to measurement objective</p> <p>Section 2 Expert validated data structure</p> <p>Section 3 Text data for evaluation</p>
---	--

Figure 3 Downstream estimates based on the classified innovation variable across model configurations (Study 1)



Note. Expert-labeled holdout sample of Demonstration Study 1 ($n = 1,000$). Each dot is one model configuration, grouped by model family. Panel (a): classified innovation variable serves as explanatory variable in a simulated outcome with true $\beta = .50$ (dashed line); estimates are averaged across simulation draws. Panel (b): classified innovation variable serves as outcome; dashed line marks the expert-labeled post-pre difference of -0.06 percentage points.

Table 1 Framework for mapping expert insights into LLM applications

Expert Evaluation	LLM Process Mapping	Design Decisions
<i>Task Definition</i>	Prompt Section 1: <ul style="list-style-type: none"> - Researcher describes task and defines desired output according to the measurement objective. - Task description and output definition are included in the prompt. 	<ul style="list-style-type: none"> - Expert selection criteria (number of experts, task-expertise alignment) - Prompt engineering
<i>Data Engagement</i>	Inductive Coding AND Prompt Section 2: <ul style="list-style-type: none"> - Researcher conducts inductive three-pronged (open, axial, selective) coding procedure with the data until code/thematic saturation is reached. - Data structure is validated with expert(s) to assure meaning saturation and adequate structuring of cues. - Data dimensions (or categories) are included in the prompt. 	<ul style="list-style-type: none"> - Prompt engineering
<i>Relate to Prior Knowledge</i>	Prompt Section 3 OR Manual Output Interpretation: <ul style="list-style-type: none"> - Expert assesses relevance and weight of each data dimension (or category) for the measurement. - Relevance and weights are included in the prompt or researcher interprets model output based on the relevance of dimensions for the defined task 	<ul style="list-style-type: none"> - Prompt engineering - Weight elicitation format (in-prompt vs. post-processing)
<i>Judgment Formation</i>	Output validation AND Fine-Tuning: <ul style="list-style-type: none"> - Researcher and expert validate model output and measurement - Researcher labels training and validation data for model fine-tuning on desired outcome (input: prompt + text data, output: classification/measure in expected format) 	General: <ul style="list-style-type: none"> - Model selection - Temperature setting Fine-Tuning: <ul style="list-style-type: none"> - Training data size - Data distribution
<i>Reliability Assurance</i>	Design Decisions: <ul style="list-style-type: none"> - Researcher analyzes how model selection, prompt engineering, training data, and temperature setting affect consistency rate of repeated runs - Researcher optimizes and documents design decisions for reliability 	

Table 2 Model performance for identification of innovative updates

		F1-score	Precision	Recall	Accuracy	CR
Version First Digit (1.0 → 2.0)	Literature	.211	.559	.130	.574	
Version Second Digit (1.1 → 1.2)		.472	.475	.470	.540	
Dictionary-Based		.666	.799	.571	.749	
Dictionary-Character-Based		.615	.477	.863	.526	
K-Nearest-Neighbor	Classical Machine Learning Models	.713	.828	.626	.779	
Logistic Regression		.850	.810	.895	.862	
Naïve Bayes		.835	.839	.831	.856	
Random Forest		.789	.740	.845	.802	
Support Vector Machine (SVM)		.849	.832	.868	.865	
Extreme Gradient Boosting (XGBoost)		.834	.808	.863	.850	
Convolutional Neural Network (CNN)	Deep Learning	.828	.817	.838	.847	
BERT	Transformer-Based PLMs	.899	.865	.936	.908	
ELECTRA		.889	.849	.934	.898	
RoBERTa		.902	.879	.927	.912	
XLNet		.887	.838	.943	.895	
Claude 3.5 Haiku	Large Language Models	.746	.944	.616	.816	1.000
Claude Sonnet 4		.896	.938	.858	.913	.999
Mistral Small		.765	.934	.648	.826	.992
Mistral Large		.851	.930	.785	.880	.984
Mistral Small (fine-tuned)		.887	.888	.886	.901	.984
Mistral Large (fine-tuned)		.909	.891	.929	.919	.985
GPT-4.1 Nano		.760	.952	.632	.825	.999
GPT-4.1 Mini		.869	.894	.845	.888	.987
GPT-4.1		.901	.923	.879	.915	.982
GPT-4.1 Nano (fine-tuned)		.919	.907	.932	.928	.998
GPT-4.1 Mini (fine-tuned)		.914	.888	.941	.922	.992
GPT-4.1 (fine-tuned)		.926	.894	.961	.933	.996
N		1,000	1,000	1,000	1,000	3,000

Note. The version approach follows Wen and Zhu (2019), the dictionary approach follows Kircher and Foerderer (2024), and the dictionary-character approach follows Agarwal and Kapoor (2023). For the ML- and transformer-based classifications, updates classified as 1 or 2 were considered innovative. ML-based classifiers were trained and transformer-based models were fine-tuned with a dataset $n=2,000$, with categories representative of the validation data. Embeddings for the ML classifiers were created using TF-IDF, and embeddings for the CNN were generated using glove-42b-300d. LLM temperature = 0.0, LLM seed = 94032. F1-score is reported for the first run. Consistency Rate (CR) is calculated based on three independent runs.

Table 3 Model performance for update classification

	1	2	3	4	5	6	7	Macro F1	Weighted F1	CR
K-Nearest-Neighbor	.665	.217	.629	.372	.118	.776	.167	.420	.639	
Logistic Regression	.793	.449	.673	.569	.211	.873	.000	.510	.759	
Naïve Bayes	.788	.496	.621	.563	.320	.878	.000	.524	.760	
Random Forest	.702	.232	.542	.607	.105	.816	.000	.429	.682	
SVM	.788	.463	.647	.583	.190	.873	.000	.506	.757	
XGBoost	.764	.353	.686	.605	.105	.853	.143	.501	.737	
CNN	.774	.349	.667	.457	.286	.854	.000	.484	.729	
BERT	.833	.500	.721	.750	.000	.906	.000	.530	.804	
ELECTRA	.850	.458	.691	.776	.000	.919	.000	.528	.813	
RoBERTa	.864	.647	.776	.840	.000	.926	.000	.579	.844	
XLNet	.842	.531	.739	.803	.000	.916	.000	.547	.819	
Claude 3.5 Haiku	.642	.471	.475	.498	.444	.756	.174	.494	.646	.997
Claude Sonnet 4	.865	.504	.556	.524	.571	.799	.296	.588	.758	.994
Mistral Small	.587	.441	.527	.466	.545	.738	.200	.501	.619	.971
Mistral Large	.772	.582	.637	.558	.389	.818	.348	.586	.742	.973
Mistral Small (fine-tuned)	.821	.511	.752	.745	.500	.904	.276	.644	.812	.966
Mistral Large (fine-tuned)	.863	.563	.764	.758	.583	.929	.400	.607	.845	.970
GPT-4.1 Nano	.615	.439	.463	.315	.462	.455	.333	.440	.501	.997
GPT-4.1 Mini	.798	.595	.614	.552	.522	.819	.286	.598	.753	.965
GPT-4.1	.849	.523	.741	.738	.435	.911	.400	.657	.825	.970
GPT-4.1 Nano (fine-tuned)	.876	.681	.764	.797	.538	.940	.375	.710	.865	.993
GPT-4.1 Mini (fine-tuned)	.882	.682	.779	.838	.522	.938	.400	.720	.870	.991
GPT-4.1 (fine-tuned)	.888	.602	.800	.877	.455	.949	.435	.715	.875	.988
N	368	70	58	78	15	400	11	1,000	1,000	3,000

Note. ML-based classifiers were trained and transformer-based models were fine-tuned with a dataset $n=2,000$, with categories distributed similarly to the validation data. Embeddings for the ML classifiers were created using TF-IDF, and embeddings for the CNN were generated using glove-42b-300d. LLM temperature = 0.0, LLM seed = 94032. F1-score is reported for the first run. Consistency Rate (CR) is based on three independent runs.

Table 4 Model performance for review classification

	0	1	2	3	4	5	6	7	8	M F1	W F1	CR
K-Nearest-Neighbor	.826	.431	.465	.092	.063	.000	.000	.000	.000	.208	.529	
Logistic Regression	.858	.644	.610	.481	.359	.087	.143	.000	.000	.354	.672	
Naïve Bayes	.850	.677	.553	.553	.407	.204	.129	.098	.100	.397	.687	
Random Forest	.813	.600	.602	.424	.511	.387	.000	.174	.100	.401	.645	
SVM	.821	.660	.609	.483	.356	.250	.118	.174	.000	.386	.664	
XGBoost	.846	.640	.573	.445	.356	.087	.000	.000	.000	.328	.655	
CNN	.800	.571	.620	.406	.379	.174	.091	.133	.000	.353	.623	
BERT	.895	.733	.410	.551	.000	.000	.000	.000	.000	.288	.687	
ELECTRA	.909	.767	.000	.611	.000	.000	.000	.000	.000	.254	.674	
RoBERTa	.923	.758	.697	.700	.439	.000	.000	.100	.000	.402	.766	
XLNet	.917	.764	.651	.674	.642	.000	.000	.273	.000	.436	.766	
Claude 3.5 Haiku	.924	.829	.753	.741	.622	.609	.625	.429	.333	.652	.827	.989
Claude Sonnet 4	.914	.791	.780	.732	.621	.427	.632	.411	.444	.639	.813	1.000
Mistral Small	.909	.776	.766	.690	.686	.161	.615	.386	.154	.571	.793	.989
Mistral Large	.900	.779	.745	.724	.641	.415	.667	.346	.364	.620	.798	.911
Mistral Small (fine-tuned)	.927	.814	.822	.703	.583	.519	.667	.526	.250	.646	.825	.971
Mistral Large (fine-tuned)	.937	.837	.814	.763	.476	.429	.643	.333	.500	.637	.836	.965
GPT-4.1 Nano	.898	.764	.675	.524	.300	.261	.519	.160	.154	.473	.740	.995
GPT-4.1 Mini	.905	.835	.739	.731	.651	.373	.629	.371	.256	.610	.812	.960
GPT-4.1	.916	.828	.749	.731	.727	.516	.647	.491	.370	.664	.825	.943
GPT-4.1 Nano (fine-tuned)	.938	.856	.785	.729	.781	.500	.727	.438	.375	.681	.846	.984
GPT-4.1 Mini (fine-tuned)	.944	.850	.844	.770	.793	.471	.435	.323	.267	.633	.851	.996
GPT-4.1 (fine-tuned)	.948	.872	.853	.769	.807	.500	.667	.467	.471	.706	.867	.997
N	475	240	92	146	31	19	13	19	12	1,047	1,047	3,000

Note. ML-based classifiers were trained and transformer-based models were fine-tuned with a dataset n=2,000, with categories distributed similarly to the validation data. Embeddings for the ML classifiers were created using TF-IDF, and embeddings for the CNN were generated using glove-42b-300d. LLM temperature = 0.0, LLM seed = 94032. F1-score is reported for the first run. Consistency Rate (CR) is based on three independent runs.

Online Appendix

AI-Based Measurement of Innovation: Mapping Expert Insight into Large Language Model Applications

Appendix A. Towards Large Language Models for Expert Evaluation	2
Appendix B. Data Structure and Coding Dimensions.....	6
Appendix C. Large Language Model Inference via Batch API	7
Appendix D. Hyperparameter Tuning.....	10
Appendix E. Implications of Design Decisions	15
Appendix F. Teacher-Student Distillation Setup	25
Appendix G. Summary of Fine-Tuning Decisions.....	28
Appendix H. Measurement Performance and Implications for Downstream Estimates.....	29
Appendix I. Temporal Relevance Robustness Check	33
Appendix J. Summary of Conditions for Applying the Framework	35
Appendix K. Handling Multi-Lingual Data	36
References	37

Paper Companion Repository:

https://github.com/robi979/AI-Innovation-Measurement_Extension

Appendix A. Towards Large Language Models for Expert Evaluation

Traditionally, scholars have operationalized variables at scale by using keyword-based methods to approximate the conceptual judgments that domain experts otherwise would have to make (e.g., Foerderer et al., 2018; Kircher and Foerderer, 2024). This method involves the use of predefined dictionaries that map specific terms to theoretically grounded constructs.

The approach is particularly effective when theoretical concepts and the terminology used to communicate them are well-defined and broadly recognized. However, keyword-based methods come with limitations. First, constructing comprehensive and reliable dictionaries can be challenging, especially when no related dictionary exists. The process often relies on subjective judgments, and validating dictionaries is non-trivial. Second, keyword-based approaches struggle to capture the context in which a word appears. For instance, they may fail to identify nuances such as negations or different semantic meanings depending on the surrounding text. While these approaches are computationally efficient and easily interpretable, their ability to comprehensively capture complex constructs is very limited (see Miric et al., 2023 for a detailed discussion).¹

To address these issues, scholars have increasingly turned to machine learning (ML) models. ML encompasses statistical algorithms that refine their performance through iterative “learning” from data. Depending on the research objective, scholars can choose between supervised and unsupervised techniques. Supervised algorithms classify observations into predefined categories, making them well-suited for tasks where evaluators assess content according to specific patterns or develop categorical variables (e.g., Miric et al., 2023). In contrast, unsupervised

¹ A notable recent development is curated ontologies, which organize expert-maintained vocabularies in multi-level hierarchies. For instance, the Medical Subject Headings in biomedicine are continuously revised by expert indexers and have been used to quantify innovation at scale (Petersen, 2022). Still, sustaining such ontologies requires an institutionalized curation effort which few research settings can draw on.

methods discover emergent themes in text data without relying on prior frameworks, making them useful for exploratory analyses (e.g., Hannigan et al., 2019).

Effective application of ML to textual data requires transforming language into numerical representations that algorithms can process. Early approaches, such as bag-of-words models, convert text into numeric representations by counting term frequencies, disregarding order and contextual information (Manning et al., 2008). For instance, popular models like TF-IDF (i.e., Term Frequency-Inverse Document Frequency) assign weights to words based on their frequency across documents, or use co-occurrence patterns to group words into latent themes as in latent topic distributions. However, these techniques often fail to capture deeper semantic relationships and struggle with comparing large numbers of documents due to their sparse, context-free representations (Turney and Pantel, 2010).

Embedding-based approaches have improved on these limitations by creating numerical representations (embeddings) of words that encode their relationships in a given text corpus. Based on the idea that words appearing in similar contexts tend to have related meanings (Harris, 1954), approaches (continuous models) like Word2Vec (Mikolov et al., 2013) and GloVe (Pennington et al., 2014) build on large text corpora and neural architectures to infer an embedding for every vocabulary item, analyzing how frequently they co-occur in a text corpus. Each word can then be represented as a vector based on all surrounding words (or vice versa). This results in vectors where semantically similar words are geometrically close, while unrelated words are farther apart. However, the embeddings generated by continuous models are static, meaning each word is assigned the same numerical representation regardless of the context in which it appears in downstream tasks. Simply put, the word “launch” would always have the same vector based on the initial text corpus, regardless of whether it now refers to “releasing a new product” or “propelling a rocket into space”.

Dynamically adjusting a word’s representation based on its meaning and context has become possible with the revolutionary transformer architectures (transformers). Transformers are deep neural networks that include positional encodings of each word’s relative position in the text document and a self-attention mechanism to produce rich contextual embeddings (Vaswani et al., 2017). In essence, self-attention allows the model to weigh surrounding words differently when updating a word’s representation, so the same word can have varying vector representations depending on its context. This breakthrough has enabled universal text representations across massive corpora, opening unprecedented avenues for analyzing the semantic content in text data.

Transformer-based models (transformer models) have excelled in a wide range of NLP tasks that previously required human expertise to achieve sufficient performance. Early pre-trained language models (PLM) such as BERT (Devlin et al., 2019) and its derivatives rely on an encoder-based transformer architecture. The encoder processes the input text holistically, capturing intricate relationships between words by producing dynamic, context-aware representations. This process results in high-quality embeddings that form the foundation for tasks such as text classification, sentiment analysis, named entity recognition, and text similarity—often achieving or even surpassing human expert-level performance.

More recently, LLMs, which are typically built on decoder-based transformer architectures, have attracted much attention. Models like GPT-3 (Brown et al., 2020) and its successors generate text by predicting the next (sub)word (token) from its preceding tokens. This generative capability makes LLMs highly effective for tasks such as text generation, summarization, and conversational artificial intelligence, while also performing well in other NLP tasks. Unlike encoder-based models, which typically require further task-specific fine-tuning with labeled datasets, decoder-based models can be instructed via natural language prompts. In this context, a prompt conditions the model’s response by providing contextual and directive information. Technically, the prompt

is concatenated with the input sequence, and through the self-attention mechanism, the model weights different parts of the prompt and context accordingly, thereby guiding the generation process (Brown et al., 2020; Liu et al., 2023). ‘Reacting’ to prompts, they can perform tasks directly, often without task-specific fine-tuning, significantly reducing the barriers to their use.

The scaling of model size and pre-training data has driven significant performance improvements (Brown et al., 2020), enabling these models to exhibit emergent abilities such as answering unseen questions, performing arithmetic, and reasoning over multiple steps. This evolution not only makes fine-tuning an option for many use cases (de Kok, 2025) but also provides researchers with the ability to incorporate expert knowledge into models for various applications with minimal prerequisites.

Appendix B. Data Structure and Coding Dimensions

B1 Demonstration Study 1

Figure B1 presents the data structure derived from an inductive, multi-stage analysis of software application release notes. Drawing on our sample pool of $n = 4,000$ release notes, we first conducted open coding to identify categories of informant terms. Then, we clustered these categories into broader themes and even further to aggregate dimensions, forming the data structure of this context. We coded sequentially and iterated across these stages until no new categories and relationships emerged (Gioia et al., 2013; Strauss and Corbin, 1998).

To assess validity and refine the data structure, we reviewed release-note content in light of Apple's developer guidelines and discussed typical software development workflows, update reporting, versioning logic, as well as our preliminary data structure in interviews with several domain experts (software application developers recruited via Fiverr, see Table B1). The final data structure comprises 18 second-order themes that formed seven aggregate dimensions.

----- Insert Figure B1 and Table B1 about here -----

B2 Demonstration Study 2

Figure B2 presents the data structure derived from an inductive, multi-stage analysis of user-generated product reviews of software applications. Our coding followed the procedure used in Demonstration Study 1. The final data structure comprises 20 second-order themes that formed nine aggregate dimensions.

----- Insert Figure B2 about here -----

Appendix C. Large Language Model Inference via Batch API

C1 Batch Processing

To evaluate large language model (LLM) performance at scale in both Demonstration Studies, we leveraged batch inference APIs from all three model providers included in the experiments—OpenAI, Mistral, and Anthropic—across all design decision variations.

For each provider, a systematic procedure was implemented to generate, upload, and execute batch requests for both base and fine-tuned models, adhering to the respective provider’s API documentation.² The set of evaluated models included:

- OpenAI: *GPT-3.5-turbo-0125* (default & fine-tuned), *GPT-4o-mini-2024-07-18* (default & fine-tuned), *GPT-4o-2024-08-06* (default & fine-tuned), *GPT-4.1-2025-04-14* (default & fine-tuned), *GPT-4.1-mini-2025-04-14* (default & fine-tuned), *GPT-4.1-nano-2025-04-14* (default & fine-tuned), *o3-2025-04-16* (default), *o4-mini-2025-04-16* (default)
- Mistral: *mistral-large-2411* (default & fine-tuned), *mistral-medium-2505* (default), *mistral-small-2503* (default & fine-tuned), *open-mistral-nemo-2407* (default & fine-tuned), *ministral-8b-2410* (default & fine-tuned), *ministral-3b-2410* (default)
- Anthropic: *claude-sonnet-4-20250514* (default), *claude-3-7-sonnet-20250219* (default), *claude-3-5-haiku-20241022* (default), *claude-3-5-sonnet-20241022* (default), *claude-3-haiku-20240307* (default), *claude-opus-4-20250514* (default), *claude-3-opus-20240229* (default)

Fine-tuned models were trained on datasets (distribution and size) similar to those in the main article above but included the respective prompts. Further design decisions reported in Online Appendix E included *prompt* engineering and *temperature* setting. As such we iterated each model over the *prompt* templates (default, few-shot, automatic-chain-of-thought, manual-chain-of-thought, contrastive-chain-of-thought) and *temperature* settings (0, 0.5, 1.0, 1.5). For Anthropic models, the maximum *temperature* was limited to 1.0 per provider documentation. For each model-

² See <https://platform.openai.com/docs/guides/batch> (Retrieved February 18, 2026), <https://docs.mistral.ai/capabilities/batch/> (Retrieved February 18, 2026), and <https://docs.anthropic.com/en/docs/build-with-claude/batch-processing> (Retrieved February 18, 2026) for details.

prompt-temperature combination, three repeated runs were performed on the full validation dataset, using a fixed *seed* (OpenAI and Anthropic) or *random_seed* (Mistral) parameter set to 94032 for reproducibility. For all batch jobs, *max_completion_tokens* (OpenAI) or *max_tokens* (Mistral/Anthropic) was set to 1000.

For OpenAI’s reasoning models (i.e., *o3-2025-04-16* and *o4-mini-2025-04-16*), the *temperature* parameter was omitted in all batch requests, as these models do not accept temperature control. Additionally, for these models, the *reasoning_effort* parameter was set to ‘low’.

After all batch files were constructed, they were uploaded using the respective provider’s API. For OpenAI and Mistral, each uploaded batch file produced a file identifier, which was then referenced when creating a batch classification job, specifying the model and endpoint (*/v1/chat/completions*), along with metadata for job tracking. For Anthropic, batch submission was performed via the *messages.batches.create* endpoint, directly uploading a list of parameterized requests. Unique identifiers (*custom_id*) were assigned to every request for traceability and later result matching.

C2 Post-processing and Output Cleaning

To evaluate large language model (LLM) performance at scale in both Demonstration Studies, we leveraged batch inference APIs from all three model providers included in the experiments—OpenAI, Mistral, and Anthropic—across all design decision configurations.

To ensure comparability and accurate evaluation of LLM predictions, a standardized post-processing pipeline was applied to all raw outputs from the batch API runs across providers. For each provider (OpenAI, Mistral, Anthropic), provider-specific parsing functions were implemented to extract the predicted content and align it with the corresponding sample identifiers. Within each output file, the *custom_id* field was parsed to reconstruct the relevant design configuration including the model name, prompt template, temperature setting, run index, and sample identifier.

Further the raw batch outputs (*message*) were parsed to extract the predicted content. The extracted predictions were mapped back to the holdout sample.

Given the potential variability in free-form LLM outputs, especially under more complex prompts or increased temperature values, we implemented an output cleaning procedure to reliably extract class labels. In particular, we used *regular expressions* to accommodate a wide range of output formats, correctly extracting single integer predictions, multiple class outputs separated by semicolons (as in Demonstration Study 2), as well as labels embedded in extra text and further non-numeric characters such as parentheses or stray punctuation. Any invalid, missing, or out-of-range predictions (i.e., anything not mapping to the defined target labels of the Demonstration Study) were assigned a fallback class label to ensure these were penalized in downstream performance evaluation.

Appendix D. Hyperparameter Tuning

D1 Tuning Numerical Text Representations and Hyperparameters for Classical Machine Learning Classifiers

To identify the optimal model configuration for update classification, we systematically tuned both feature representations and classifier hyperparameters using an automated and reproducible pipeline via the *scikit-learn* (Pedregosa et al., 2011) and *xgboost* (Chen and Guestrin, 2016) libraries. For the analyses in the main article, we used representative training sets consisting of $n = 2,000$ release notes and labels for Demonstration Study 1, and $n = 2,000$ reviews (including title and body) and labels for Demonstration Study 2. For the additional analysis in Online Appendix E4 of the paper, we varied the training data distribution (balanced) and size ($n = 100, 250, 500, 1,000$) in both studies. To ensure compatibility with *scikit-learn*, labels in Study 1 were zero-indexed. For the multi-label classification task in Study 2, labels were transformed into multi-hot vectors. A fixed holdout sample ($n = 1,000$) was reserved for final evaluation.

Two feature representation approaches were evaluated. First, *TF-IDF* vectorization was applied, with the following hyperparameters included in the search: *max_features* (random integers between 2,000 and 10,000), *ngram_range* ((1,1), (1,2), or (1,3)), *min_df* (1-4), and *max_df* (uniformly between 0.6 and 1.0). Second, distributed representations were obtained by averaging pre-trained *GloVe* word embeddings, considering four variants: *6B-100d*, *6B-300d*, *42B-300d*, and *840B-300d* (Pennington et al., 2014). Document vectors were precomputed and cached for each embedding.

For each representation, we evaluated *Random Forest*, *Logistic Regression*, *K-Nearest Neighbors*, *Support Vector Machine*, and *XGBoost* classifiers. *Multinomial Naive Bayes* was included for *TF-IDF* features but excluded for *GloVe* representations as the algorithm is designed specifically for discrete feature counts (such as those generated by *TF-IDF*) and does not perform

reliably on continuous or dense embeddings like *GloVe*. The hyperparameter ranges were as follows:

- Random Forest: *n_estimators* (50-500), *max_depth* (5-100), *min_samples_split* (2-20), *min_samples_leaf* (1-10), *max_features* (0.1-1.0), *criterion* ('gini', 'entropy'), *bootstrap* (True, False)
- Logistic Regression: *C* (0.01-10), *penalty* ('l2'), *solver* ('lbfgs', 'saga')
- K-Nearest Neighbors: *n_neighbors* (1-30), *weights* ('uniform', 'distance'), *leaf_size* (10-50)
- Multinomial Naive Bayes: *alpha* (1e-6-1.0), *fit_prior* (True, False)
- Support Vector Machine: *C* (0.1-10), *kernel* ('linear', 'rbf', 'poly'), *gamma* ('scale', 'auto'), *degree* (2-5)
- XGBoost: *n_estimators* (50-500), *max_depth* (3-10), *learning_rate* (0.01-0.3), *subsample* (0.5-1.0), *colsample_bytree* (0.5-1.0)

Hyperparameter optimization was conducted using scikit-learn’s *RandomizedSearchCV* with 50 randomly sampled parameter sets per classifier. For *TF-IDF* pipelines, both vectorizer and classifier parameters were tuned jointly. For *GloVe*, only classifier parameters were tuned using the precomputed document vectors. All searches used 3-fold stratified cross-validation and optimized the macro-averaged F1 score, with a fixed random seed (`random_state=94032`) for full reproducibility. Computation was parallelized for efficiency.

For Demonstration Study 1 (single-label), standard scikit-learn classifiers were used. For Demonstration Study 2 (multi-label), all classifiers were wrapped with *OneVsRestClassifier*, and multi-hot encoded targets were used as labels. The best configuration for each representation, embedding, classifier, and training set was selected based on cross-validation performance, retrained on the full training data, and evaluated on the holdout sample.

----- Insert Table D2 about here -----

Table D2 summarizes the best-performing hyperparameter configurations identified via randomized search for the machine learning classifiers used in the analyses in the main article. For each classifier, we report the macro-averaged F1 scores from both cross-validated training and evaluation on the holdout sample. Comprehensive details including all tested configurations, cross-

validation results, search parameters, predictions on the holdout sample, and summary metrics are provided in the online repository to ensure full reproducibility.

D2 Tuning Hyperparameters for Convolutional Neural Network Classifier

To identify optimal configurations for neural network-based text classification, we performed automated hyperparameter tuning of a convolutional neural network for sentence classification (*TextCNN*) with pre-trained embeddings (Kim, 2014). Data preparation was not repeated; the same training and validation splits as for the machine learning classifiers were used for the *TextCNN* runs.

For each dataset, the pipeline consisted of a custom tokenizer and the CNN implemented via the *skorch scikit-learn* library (Tietz et al., 2017). Tokens were mapped to 300-dimensional *GloVe* embeddings (42B-300d), which were preloaded and assigned to each vocabulary item. The following parameter ranges were included in the search:

- Tokenizer: *num_words* (10,000-30,000), *seq_len* (100-300)
- CNN architecture: *n_filters* (64-256), *kernel_size* (3-8), *dense_units* (32-256), *dropout* (0.1-0.5), *trainable* (True, False)
- Training: *batch_size* (32-128), *max_epochs* (5-15)

The hyperparameters were jointly optimized using scikit-learn’s *RandomizedSearchCV*, sampling 30 parameter configurations per run.³ All searches used 3-fold stratified cross-validation and optimized the macro-averaged F1 score, with a fixed random seed (*random_state*=94032) for full reproducibility. Computation was parallelized for efficiency.

In Demonstration Study 1, the network was trained using categorical cross-entropy loss for single-label classification. In Demonstration Study 2, a multi-label extension of CNN with a binary cross-entropy loss function (*BCEWithLogitsLoss*) was employed to accommodate multi-hot target

³ Given the increased computational effort associated with neural model training, 30 randomized hyperparameter configurations were evaluated for each CNN pipeline, compared to 50 for classical models.

vectors. The configuration achieving the best mean cross-validated macro-averaged F1 score was retrained on the full training data and evaluated on the holdout sample.

----- Insert Table D3 about here -----

Table D3 summarizes the best-performing hyperparameter configuration identified via *RandomizedSearchCV* for the Convolutional Neural Network used in the analyses in the main article. Macro-averaged F1 scores from both cross-validated training and evaluation on the holdout sample are reported. Comprehensive details including all tested configurations, cross-validation results, search parameters, predictions on the holdout sample, and summary metrics are provided in the online repository to ensure full reproducibility.

D3 Tuning Hyperparameters for Pre-Trained Language Models

To identify optimal configurations for transformer-based text classification, we fine-tuned several pre-trained language models (PLMs) on the same training and validation splits as the classical machine learning and CNN experiments.

We evaluated four widely used pre-trained transformer models—*BERT-base-cased* (Devlin et al., 2019), *RoBERTa-base* (Liu et al., 2019), *XLNet-base-cased* (Yang et al., 2019), and *ELECTRA-base-discriminator* (Clark et al., 2020)—using the Hugging Face *transformers* library (Wolf et al., 2020). For each model, the associated tokenizer was used to preprocess text, truncating or padding each sample to a fixed *maximum_length* (=512 tokens). In Demonstration Study 1, labels were zero-indexed, and models were trained using a standard cross-entropy objective. In Demonstration Study 2, labels were transformed to multi-hot vectors, and models were configured for multi-label classification using a sigmoid output and binary cross-entropy loss.

Hyperparameter optimization was performed using *Optuna* (Akiba et al., 2019) with 5 trials per model and training set. For each trial, the following parameters were tuned: *learning_rate* (log-uniform between 1e-5 and 5e-5), *batch_size* (8 or 16), *num_train_epochs* (3-5), and *weight_decay*

(0.0-0.01). Each configuration was trained and evaluated using the Hugging Face *Trainer* framework, and the macro-averaged F1 score on the holdout sample was used as the selection criterion. All experiments were seeded (*random_state*=94032) for reproducibility. For each model and training split, the configuration with the highest validation macro-averaged F1 score was selected and saved for downstream evaluation.

----- Insert Table D4 about here -----

Table D4 summarizes the best-performing hyperparameter configurations identified via *Optuna* trials for the PLMs used in the analyses in the main article. For each model, we report the macro-averaged F1 score of the selected configuration on the holdout sample. Comprehensive details including all tested configurations, cross-validation results, search parameters, predictions on the holdout sample, and summary metrics are provided in the online repository to ensure full reproducibility.

Appendix E. Implications of Design Decisions

Design decisions, such as model selection, prompt engineering, temperature settings, training data distribution, and data size, are critical when applying the proposed evaluation framework because they can affect the model’s performance, output consistency, and applicability to specific tasks. These decisions shape how well the model aligns with the intended use case and determine the quality and interpretability of its outputs (Carlson and Burbano, 2026; Cheng et al., 2024). Therefore, transparency in design decisions is required. For reviewers and editors tasked with verifying approaches, understanding these decisions is essential to critically assess the validity of the results, as well as to ensure that the chosen decisions are appropriate for the research objectives.

E1 Model Selection

When applying the proposed evaluation framework, researchers have to select an appropriate LLM from the numerous available models from providers like OpenAI (GPT), Anthropic (Claude), and Mistral (Ministral/Mistral). Their models differ in attributes such as model age, model size in terms of parameters and pre-training data scale, reasoning capabilities, and general evaluation benchmarks⁴. In both demonstration studies, we evaluate the model families available from each provider at the time of analysis (see Appendix C).

----- Insert Table E5 about here -----

⁴ General-purpose evaluation of language models relies on standardized benchmarks such as GPQA, Humanity’s Last Exam, and Massive Multitask Language Understanding (MMLU). These tests probe a wide range of competencies through multiple-choice and short-answer questions that span dozens of academic and professional subjects. Each item is written by domain experts and linked to a single, unambiguous, and easily verifiable answer that cannot be retrieved through a quick web search, ensuring that scores do not reflect a simple information lookup. Complementing these broad assessments are specialized benchmarks that focus on domain-specific abilities—for example, mathematics (e.g., GSM-8K), programming (e.g., HumanEval), or creative writing. They emphasize complex problem-solving within their respective fields. Thought experiments such as the Turing Test add a qualitative dimension by examining how convincingly a model can mimic human-like understanding and conversation. For fair comparison, models are evaluated in their default configurations. Past developments have shown that newer and larger-sized model versions or explicitly reasoning-optimized models tend to achieve the highest scores.

Table E5 displays the performance results by model provider for both demonstration studies. The results confirm the earlier observations from Tables 3 and 4 that larger models outperform their smaller counterparts in default configurations. For example, GPT-4o demonstrated notably higher performance (Macro F1-Scores of (1) .602 and (2) .640) than GPT-4o-mini (Macro F1-Scores of (1) .484 and (2) .576) or the predecessor GPT-3.5 Turbo (Macro F1-Scores of (1) .439 and (2) .564). This pattern holds consistently across Claude and Mistral model families. However, model recency does not guarantee superior performance within similar size categories. In Demonstration Study 2, Claude Sonnet versions showed comparable results across Sonnet 3.5 (Macro F1-score = .649), Sonnet 3.7 (Macro F1-score = .638), and Sonnet 4 (Macro F1-score = .631), while in Demonstration Study 1 earlier versions even outperformed the newest model, with Sonnet 3.5 achieving a Macro F1-score of .644 and Sonnet 3.7 achieving .613, compared to Sonnet 4's score of .588 (Table 3, main article).

Fine-tuning significantly improves model performance across all tested models. Most remarkably, fine-tuned GPT-3.5 models exceeded the performance of GPT-4.1 models in Demonstration Study 1 and nearly matched their performance levels in Demonstration Study 2. Smaller Mistral models (Minstral and Open Mistral Nemo), designed primarily for local implementation with minimal hardware requirements, initially performed poorly with F1-scores below .4 in Demonstration Study 1. However, these models showed substantial performance improvements through fine-tuning, approaching the performance levels of larger fine-tuned Mistral model classes.

The performance of new reasoning models varies across providers. While Claude Opus 4 underperforms compared to Sonnet models in Demonstration Study 1 and matches them in Demonstration Study 2, o3 achieved performance values close to fine-tuned GPT models. This capability simplifies large-scale text analysis implementation within our framework without

requiring additional training data. However, reasoning models are considerably more resource-intensive, requiring up to several hundred times more tokens to generate outputs similar to non-reasoning models.

Taken together, larger-sized or reasoning models already achieve strong results in their default versions. Provider differences are minor, with OpenAI’s GPT models slightly outperforming others particularly when fine-tuned. Notably, when fine-tuned on adequately sized, task-specific datasets, mid- and large-sized models exhibited negligible performance differences regardless of provider or size. Consequently, researchers applying our framework should systematically monitor multiple LLM candidates to identify the best fit for their evaluation tasks rather than assuming that higher parameter count, more recent releases, or provider reputation automatically translates into superior outcomes. Appendix I extends this comparison to the most recent model generation.

In addition to performance considerations, practical factors such as usage costs, energy consumption, and model accessibility are also worth considering. The majority of the models utilized in our demonstration studies are proprietary, accessed via APIs with usage-based fees, typically lower for smaller-sized models. Many LLM candidates are also available as open-weight, downloadable, and runnable on local computing infrastructure. However, while smaller models such as Ministral 8B or Open Mistral Nemo are compatible with standard computational resources, mid-sized open-weight models like Mistral Small already require significantly enhanced computational capabilities, especially when fine-tuning them.⁵ Due to such resource constraints, our demonstration studies did not include locally run tests of models comparable in size to those

⁵ Running Mistral Small on a single GPU requires at least 44 GB of GPU RAM (see <https://huggingface.co/mistralai/Mistral-Small-Instruct-2409> (Retrieved February 18, 2026) for details). Fine-tuning Mistral Small is optimized for multi-GPU, single-node training setups. For best efficiency, it is recommended to use an A100 or H100 GPU (see <https://github.com/mistralai/mistral-finetune> (Retrieved February 18, 2026) for details).

analyzed via APIs. Still, we expect their performance to align with our observed results from proprietary models. Since their architectures are identical, performance discrepancies would stem from differences in computing infrastructure.

E2 Prompt Engineering

Prompt engineering refers to the strategic modification of prompts to enhance the performance of LLMs on specific tasks (Brown et al., 2020). Studies in several contexts demonstrate that these modifications can yield significant performance improvements for LLM applications (see, for example, the systematic survey by Sahoo et al., 2024). In both demonstration studies, we assessed different prompt engineering techniques using OpenAI’s GPT models: few-shot prompting, automatic chain-of-thought, manual chain-of-thought, and contrastive chain-of-thought. Few-shot prompting involves presenting the model with a limited number of example input-output pairs (Brown et al., 2020). Automatic chain-of-thought prompting aims at generating intermediate reasoning steps by adding a “Let’s think step by step” instruction at the end of the prompt (Zhang et al., 2022). In contrast, manual chain-of-thought prompting integrates a specific logical reasoning chain into the prompt (Wei et al., 2022). Contrastive chain-of-thought prompting adds learning from mistakes by providing both valid and invalid reasoning explanations alongside the original prompt (Chia et al., 2023). The prompt texts for each technique are documented in Tables E6 and E7.

----- Insert Tables E6, E7, & E8 about here -----

Table E8 displays the performance results by prompt engineering technique for both demonstration studies. For default models, few-shot prompting showed slight performance improvements compared to the “default” prompt. However, it is important to understand that few-shot prompting introduces the risk of selection bias stemming from the choice of examples. Chain-of-thought prompting yielded mixed results. While GPT-4.1 Mini and GPT-4.1 performance

decreased with automatic and manual variants but improved with contrastive chain-of-thought prompting in Study 1, all chain-of-thought variations slightly improved performance in Study 2. Notably, GPT-4.1 Nano performance dropped strongly with chain-of-thought prompting in Study 1 (baseline Macro F1-Score with the default prompt = .440).

Interestingly, prompt engineering had only minimal performance implications for fine-tuned models, even though these models had been fine-tuned with a different prompt. Fine-tuned models thus appear largely robust to prompt variation once prompt-specific training has taken place. However, the fine-tuned GPT-4.1 Nano model showed a different pattern, with performance declining when prompts deviated from training data. In Study 1, the ‘default’ prompt achieved a Macro F1-score of .710, while alternative prompt variations: few-shot (.650), automatic chain-of-thought (.593), manual chain-of-thought (.554), and contrastive chain-of-thought (.649) scored lower. Similar declines, though less pronounced, appeared in Study 2.

The results highlight that prompt engineering can impact performance across different models and contexts, even for fine-tuned systems. However, for larger models we observe that when prompts are carefully crafted and validated with expert input—particularly by guiding through data structures and classification patterns—the potential for performance gains through prompt engineering appears limited.

E3 Temperature Value

The temperature is a hyperparameter used in language models to control how random or predictable the generated outputs are (Ackley et al., 1985). When models produce outputs, they first compute raw scores (logits) for each possible next token. These logits are converted into probabilities using the softmax function, which normalizes them so that they sum to 1. The temperature parameter influences the shape of this probability distribution by redistributing the probabilities across tokens. A higher temperature (> 1) flattens the distribution, increasing uncertainty and randomness.

Conversely, a lower temperature (< 1) sharpens the distribution, emphasizing tokens with higher probability and reducing randomness.⁶ Temperature values typically range from 0 to 2, though some providers use different scales (e.g., Claude models cap at 1). Setting temperature = 0 removes randomness entirely, causing the model to always choose the token with the highest probability (greedy sampling). Conversely, higher temperature settings can introduce randomness, enabling more creative or unexpected outputs (Peeperkorn et al., 2024). However, increased randomness comes at the expense of output reliability across repeated runs, which is a critical factor in research-oriented evaluations.

We analyze the implications of this setting in our demonstration studies by gradually increasing temperature values. Figure E3 displays F1-scores of the first run and Consistency Rate (CR) across three independent runs for temperature values of 0.0, 0.5, 1.0, and 1.5 for both demonstration studies. The metrics indicate that higher temperature settings do not improve model performance and instead result in a slight decline in performance alongside a reduction in reliability as measured by the Consistency Rate for some models. Most changes are minor, with OpenAI’s GPT models in particular showing stable performance and reliability. For example, in Demonstration Study 1, our framework applied to a GPT-4.1 (default) model shows a decline in macro averaged F1-score from .657 at temperature = 0.0 to .638 at 0.5, .636 at 1.0, and .650 at 1.5. The Consistency Rate changes from .970 at temperature = 0.0 to .98 at 0.5, .979 at 1.0, and .968 at 1.5. In contrast, a more pronounced decrease in Consistency Rate is evident for the Claude models and Mistral Large (default). For instance, the Consistency Rate of Mistral Large falls from .973 at

⁶ *Top_p* is a further parameter used to adjust the output distribution. The *top_p* parameter selects a subset of tokens whose probability exceeds a threshold *p*. A lower *top_p* value restricts generation to the most likely tokens and thus reduces randomness, while a higher value allows more tokens to be considered. For instance, a *top_p* of 0.2 means only the tokens that together account for the top 20% of the probability mass are considered. In practice, since both temperature and *top_p* influence the output by adjusting the effective token probability distribution, tuning them simultaneously can lead to redundant effects.

temperature = 0.0 to .828 at temperature = 1.5 in Demonstration Study 1, and from .911 to .611 in Demonstration Study 2.

----- Insert Figure E3 about here -----

Practically, this decrease in consistency means that at higher temperature settings, models like Mistral Large and Claude are less likely to generate the same output for identical input across runs, which undermines reproducibility. When operationalizing text-based variables this variability can introduce noise, making it difficult to reliably measure constructs, replicate results, or compare findings across studies. In addition, increased randomness raises the effort required for output cleaning, as deviations from the defined format become more frequent. This problem is particularly pronounced at the maximum temperature setting (2.0), where our tests revealed that model outputs often deviated so strongly from expected formats that label identification became unfeasible, ultimately rendering these outputs unusable for analysis and leading to their exclusion from further evaluation. Based on these findings, we recommend that researchers adopt a conservative default temperature (≤ 0.5) for tasks requiring strict output structure or having high reproducibility demands, unless a higher value is conceptually justified.

E4 Training Data Distribution and Data Size

As demonstrated in previous analyses across both demonstration studies, LLMs fine-tuned with additional training data can lead to substantial improvements in performance. Likewise, earlier ML and PLM approaches have performed well in both studies. However, when the distribution of the training data reflects that of the real-world holdout sample, some labels become underrepresented. Models that rely heavily on these training data distributions tend to perform worse on underrepresented labels compared to those labels that have sufficient representation in the training set.

To address this imbalance, two primary approaches can be considered (e.g., Johnson and Khoshgoftaar, 2019). The first approach focuses on the algorithmic level, where model weights are adjusted to assign greater importance to underrepresented classes, thereby imposing heavier penalties for classification errors related to these classes in the training or fine-tuning process. Essentially, this method instructs the model to pay greater attention to rare labels by amplifying the consequences of mistakes associated with them. The second is at the data level, adjusting the training data directly. This involves modifying the distribution of the training set through strategic oversampling of underrepresented labels, thereby enhancing model exposure to these less frequent examples. This method is particularly relevant and effective when algorithm-level adjustments are infeasible, as is often the case with fine-tuning proprietary LLMs.

Building on these considerations, it is important to recognize that, in practice, particularly data-level approaches depend on the availability of sufficiently large datasets. In previous analyses, we deliberately created extensive training and validation datasets to avoid performance differences arising solely from data limitations. However, creating large training and validation datasets is often time-consuming, complex, or constrained by data or the availability of human coders, depending on the evaluation task. To better understand how data constraints interact with distributional characteristics, we systematically reduced the size of the training datasets for both balanced and representative class distributions. Specifically, we generated subsamples with $n = 1,000$, $n = 500$, $n = 250$, and $n = 100$, maintaining either a balanced or representative class distribution.

----- Insert Figure E4 about here -----

Figure E4 presents the macro-averaged F1-scores as training data size increases, for both balanced and representative class distributions across both demonstration studies.⁷ In Demonstration Study 1 with the representative (real-world) class distribution, clear differences between models emerge. The fine-tuned GPT-4.1 Nano and Mini models outperform their default counterparts even with smaller sample sizes, showing steady gains up to $n = 500$, a slight decline at $n = 1,000$, and reaching peak performance at $n = 2,000$. The larger GPT-4.1 model also starts above its default version with small sample sizes, dips below default at $n = 500$, but then shows a steady increase, plateauing from $n = 1,000$ onwards. In contrast, both fine-tuned Mistral models initially underperform relative to their default versions, only surpassing default performance at the largest sample size ($n = 2,000$). When using a balanced class distribution, performance increases more smoothly for all models as sample size grows, and maximum F1-scores are higher than with the representative distribution. Still, the Mistral Small model does not surpass its default until $n = 1,000$, and Mistral Large until $n = 2,000$. In Demonstration Study 2, balanced distributions again yield steady growth and higher peak F1-scores compared to representative distributions, and no fine-tuned model underperforms its default version at any sample size. The representative distribution in Study 2 shows less pronounced overall gains, echoing the overall moderate improvements seen in Study 1.

Taken together, larger-sized or reasoning models already achieve strong results in their default versions. Provider differences are minor, with OpenAI’s GPT models slightly outperforming others particularly when fine-tuned. Notably, when fine-tuned on adequately sized, task-specific datasets, mid- and large-sized models exhibited negligible performance differences regardless of provider or size. Consequently, researchers applying our framework should

⁷ Additional analyses for variations in training data size and distribution for classical ML models, CNN, and earlier PLMs are presented in Figure E5.

systematically monitor multiple LLM candidates to identify the best fit for their evaluation tasks rather than assuming that higher parameter count, more recent releases, or provider reputation automatically translates into superior outcomes.

In addition to performance considerations, practical factors such as usage costs, energy consumption and model accessibility are also worth considering. The majority of the models utilized in our demonstration studies are proprietary, accessed via APIs with usage-based fees, typically lower for smaller-sized models. The tested Mistral models are also available in open-weight versions, and can be run on local computing infrastructure. However, while smaller models such as Ministral 8B or Open Mistral Nemo can operate effectively on standard computational resources, mid-sized open-weight models like Mistral Small already require significantly enhanced computational power, especially when fine-tuning them. Due to such resource constraints, our demonstration studies did not include local testing of models comparable in size to those analyzed via APIs. Still, we expect their performance to align with our observed results from proprietary models, particularly given models such as Mistral’s offerings are available in both proprietary and open-weight versions. Since their architectures are identical, any performance discrepancies would likely stem from variations in computing infrastructure.

Appendix F. Teacher-Student Distillation Setup

Both demonstration studies show that the highest-performing applications of our framework rely on proprietary, frontier LLMs that are pre-trained on extremely large corpora. While frontier models are expected to perform best in this setting, in many cases they are accessible only via provider APIs, which limits control over model versioning and may create uncertainty regarding long-run reproducibility. Moreover, deploying open-weight models at comparable capability levels often requires substantial hardware resources or cloud infrastructure, making large-scale inference costly or operationally complex. At the same time, smaller models and earlier text-classification approaches typically underperform unless substantial expert-labeled training data is available—data that is often expensive or infeasible to obtain when expert time is scarce.

One opportunity to address these concerns is to use an LLM-based application of our framework—once it achieves high expert-aligned performance—as a labeling instrument to expand training data for more lightweight approaches (e.g., Durvasula et al., 2024). The objective of this extension is not to surpass the teacher LLM’s performance, but to produce a cost-efficient “clone”—a student model that approximates the teacher model’s expert-mimicking evaluations while being locally deployable and stable over time. Conceptually, this approach transfers the teacher model’s ability to operationalize expert cues (as encoded in our framework) into models that are cheaper to run and easier to reproduce. This teacher-student distillation setup yields three practical benefits. First, it improves control and replicability because student models can be run locally, independent of API provider changes. Second, it reduces marginal cost for large-scale measurement once a student model is trained. Third, it can retain much of the evaluation quality to the extent that student models learn the teacher’s decision boundary from sufficiently large pseudo-labeled datasets.

We implement this approach in Demonstration Study 1. Specifically, we use one of the best-performing framework specification as a teacher to generate labels for progressively larger sets of additional release notes. We then train and tune a set of student models, including classical ML baselines and smaller transformer-based models, on these pseudo-labeled datasets and evaluate them on the same expert-labeled holdout sample used throughout the study.

----- Insert Figure F6 about here -----

Figure F6 plots macro-averaged F1-Score as a function of scaled training size (2k expert labels; 10k and 20k teacher-generated labels). Two patterns emerge. First, teacher labeling improves student performance broadly. Relative to the 2k baseline, the classical ML student models move into the mid-.60 Macro F1-Scores at 10k (Naïve Bayes = .651; SVM = .664; XGBoost = .624), and the PLM student models improve more strongly (BERT = .710; XLNet = .736). Second, additional scale from 10k to 20k yields more model-specific returns. The PLM students continue to close the gap to the teacher model benchmark (dashed line at Macro F1 = .744). In fact, BERT rises further to a Macro F1-Score = .743 (essentially matching the benchmark), while XLNet remains high at .733.

Two additional results are noteworthy. GPT-4.1 Nano benefits from additional teacher model-labeled data and slightly exceeds the teacher benchmark at 20k (Macro F1 = .751), indicating that distillation can transfer much of the teacher model’s decision boundary into a smaller, cheaper-to-run model when sufficient pseudo-labeled data are available. The open-weight Ministral 8B model also improves with scale (Macro F1-Score = .623 at 20k), but its gains are comparatively muted in this setup, indicating that not all compact open-weight models may absorb the teacher signal equally well in this setting—potentially due to architectural constraints or optimization limits defined by pre-training (Mirzadeh et al., 2019).

Overall, the results indicate that teacher-generated labels can materially reduce the dependence on scarce human expert labeling while enabling locally deployable models that approach teacher-level measurement quality. However, with any distillation approach, this procedure inherits the teacher model’s strengths and weaknesses, meaning that systematic teacher model errors can be propagated into the student models. For this reason, we recommend maintaining a fixed expert-labeled benchmark for periodic auditing of disagreement cases when scaling the approach.

Appendix G. Summary of Fine-Tuning Decisions

----- Insert Table G9 about here -----

Appendix H. Measurement Performance and Implications for Downstream Estimates

H1 Simulation Design

With the simulation we aim to quantify how the measurement error of each evaluated configuration can impact downstream estimates in empirical analyses. For case (i), we adopt a *plasmode* design⁸ (Franklin et al., 2014; Vaughan et al., 2009) in which a simulation builds on real data and only one component is constructed, so that its true properties are known. In our setting, the constructed component is the outcome, which we generate from the expert labels with a defined effect of innovation-related updates. The classifications of each configuration and their errors are real. Estimating the outcome model with these classifications in place of the expert labels then shows how each configuration’s errors distort the estimate.

For each of the $n = 1,000$ release notes in the Demonstration Study 1 holdout we observe the expert label and every configuration’s ($n = 676$) actual classification (see Appendix E). The 676 configurations comprise 155 default LLM configurations (models, prompt designs, and temperature settings), 211 fine-tuned LLM configurations (additionally varying training data size and composition), 270 classical ML configurations (algorithms and training data), and 40 PLM configurations. The binary indicator marks innovation-related updates (dimensions 1–2 of the coding scheme, Appendix B). For case (i), six configurations produce no variation in the resulting binary measure and thus do not permit estimation of a regression coefficient, leaving 670 estimable configurations.

We construct a synthetic outcome

⁸ Plasmode simulations originate in biostatistics and pharmacoepidemiology, where methods are evaluated on real datasets in which one component is replaced by a construct with known properties. The design fits our purpose as it preserves each configuration’s empirical error structure, which a fully synthetic simulation would have to assume, while providing the known benchmark that real applications lack.

$$y_i = \beta x_i^* + \varepsilon_i \quad (1)$$

from expert labels x_i^* with $\beta = .50$ and ε_i drawn independently for each release note from a normal distribution with mean zero and variance chosen such that innovation-related updates explain $R^2 = .15$ of the outcome variance. We then estimate Equation (1) with each configuration's classification in place of x_i^* . Estimates are averaged over 1,000 simulation draws with noise realization shared across configurations within each draw, so that differences between configurations reflect only their error structures. Because the true coefficient is known by construction, deviations from it can be directly interpreted as bias.

Case (ii) requires no synthetic component. We split the holdout sample at a fixed date (April 26, 2021) into an earlier (56% of release notes) and a later period (44%) and compute for each configuration the change in the measured innovation share between the two periods. The expert labels provide the true change of $-.06$ percentage points. Sampling uncertainty in the measurement distortion is assessed using 2,000 paired bootstrap resamples, stratified by period. Within each resample, we recompute both the expert-labeled and configuration-based changes and calculate their difference. We further decompose each measured change into the expert-labeled true change, the attenuation induced by baseline false-positive and false-negative rates, and the additional contributions from drift in false-positive and false-negative rates between periods.

Two assumptions warrant attention. First, the synthetic outcome serves only to establish a known benchmark against which bias can be measured. The simulation therefore demonstrates how measurement error propagates into estimates, not the magnitude of bias in any particular identification strategy. Second, the split date carries no substantive interpretation. It provides two subsamples across which the stability of error rates can be assessed, as would any partition along the cells of a design.

H2 Innovation-Related Updates as Regressor (Case i)

Table H10 reports one representative configuration per model family: default and fine-tuned GPT-4.1 configurations, the best-performing classical ML configuration, and BERT as the PLM benchmark reported in Table 2 of the main article. Column β reports the estimated coefficient over the 1,000 simulation draws. The distance from the set $\beta = .50$ reflects the bias each configuration's errors produce. The remaining columns report the 95% simulation interval and the error-type analyses.

----- Insert Table H10 about here -----

For each configuration, we isolate the contribution of one error type by correcting the other and re-estimating. Column $\beta^{FN\ only}$ reports estimates in which only the false negatives remain and all false positives are corrected. Column $\beta^{FP\ only}$ reports estimates in which only the false positives remain and all false negatives are corrected. The relative contribution of both error types varies across configurations. For the default GPT-4.1 configuration, false positives and false negatives produce similar attenuation, with somewhat greater attenuation from false negatives. For the fine-tuned GPT-4.1, classical ML, and BERT configurations, false positives account for a larger share of the bias. In all cases, both error types attenuate the estimate toward zero.

H3 Innovation-Related Updates as Outcome (Case ii)

Table H11 reports the same four configurations for case (ii). For each configuration, we report the change in the measured innovation share between periods and its distortion relative to the expert-labeled change of $-.06$ percentage points. The 95% bootstrap confidence intervals are obtained from 2,000 paired resamples and refer to this measurement distortion. The configurations differ substantially in their point estimates of distortion, but none of the four confidence intervals exclude zero.

----- Insert Table H11 about here -----

To locate the source of the measurement-induced changes, the remaining columns decompose each estimate into the contribution of baseline measurement error and drift in false-positive and false-negative rates between the periods. Because the true change is close to zero, the contribution of baseline error is negligible for all four configurations. Differences between the measured and expert-labeled changes arise almost entirely from changes in error rates across periods. For the default GPT-4.1 configuration, the distortion stems primarily from false-negative drift, whereas for the fine-tuned GPT-4.1 configuration both false-positive and false-negative drift contribute. The classical ML configuration produces a positive distortion driven mainly by false-negative drift, while BERT exhibits comparatively stable error rates and consequently almost no distortion. Notably, the fine-tuned GPT-4.1 configuration produces the largest distortion among the four configurations despite its higher aggregate performance. Whereas aggregate performance provides a useful indication of suitability in case (i), the suitability of a configuration for a comparative design depends more strongly on the stability of its error rates across the compared periods. Drift in error rates can arise when classification errors depend on text characteristics whose distribution shifts between periods, such as the length or composition of release notes. Because period-specific error rates are estimable on the holdout sample, such distortions can be detected before analyses on the full corpus.

Appendix I. Temporal Relevance Robustness Check

Because model capabilities and versions evolve rapidly, a relevant question is whether our framework’s performance depends on a specific model snapshot or whether it generalizes across model generations. We already implicitly tested for this concern in the analysis of the model selection design decision (Online Appendix E1). However, over the course of developing the framework and conducting the two demonstration studies, several LLM providers released new model versions (e.g., OpenAI’s GPT-5 series). To assess temporal robustness, we re-ran the baseline specification of both demonstration studies on the same validation data for the most recent generation of LLMs.

----- Insert Table I12 about here -----

Table I12 reports performance (macro-averaged F1-Scores) and the CR for each demonstration study. Column (1) shows the results for the multi-class version of the update classification; Column (2) shows the results for the review classification. Across both demonstrations, the results fall in the same performance range as the LLM baselines reported in Tables 3 and 4 of the main article, indicating that our framework’s effectiveness is not specific to a single model snapshot but generalizes to newer model generations.

In Demonstration Study 1, the frontier models reach Macro F1-Scores between .467 and .642 (Weighted F1-Scores between .516 and .822) with consistently high run-to-run agreement ($CR \geq .817$). GPT-5.2 and Claude Opus 4.6 (reasoning) achieve the strongest overall performance (Macro F1 = .642 and .636, respectively). These values are broadly comparable to the strongest non-fine-tuned LLM baselines in Table 3 of the main article, though they remain below the fine-tuned GPT-4.1 variants.

In Demonstration Study 2, the newest models achieve Macro F1-Scores between .605 and .676 (Weighted F1-Scores between .801 and .843), with high consistency overall ($CR \geq .829$). GPT-5.2

again performs at the top end (Macro F1-Score = .676; Weighted F1-Score = .841), closely matching the best non-fine-tuned models in Table 4 of the main article but not exceeding the strongest fine-tuned models.

Notably, CR is lower for GPT-5 Mini and GPT-5 Nano, aligning with the baseline constraint that these models were only available at temperature = 1.0, whereas all other models can use temperature = 0.0.

Appendix J. Summary of Conditions for Applying the Framework

----- Insert Table J13 about here -----

Appendix K. Handling Multi-Lingual Data

Our demonstration studies restrict attention to English-language text. This restriction serves baseline comparability because the dictionary and classical supervised ML methods against which we benchmark are developed for and validated on English text, so that only an English corpus provides the appropriate setting for comparing approaches. The framework itself is not language-specific. The scope conditions of the framework concern the relationship between text and construct and the availability of reliable expert judgment (see Section 5 in the main article for a detailed discussion). None depends on the language of the corpus.

What varies across languages is model performance. Multilingual evaluations document that frontier LLMs perform at near-English levels for high-resource languages while performance degrades for low-resource languages (Ahuja et al., 2023; Bandarkar et al., 2023). Researchers applying the framework to non-English corpora should therefore treat language capability as an empirical question to be settled within the framework’s own validation logic. As such, the holdout sample used for performance assessment should represent the languages of the target corpus, so that measurement quality is demonstrated in the language of application. Where fine-tuning is used, the same applies to the training data. For corpora spanning multiple languages, reporting performance separately by language guards against aggregate metrics masking weak performance in individual languages. Expert coders articulating and validating cues should likewise be proficient in the target languages, since the cues that carry a construct may be language- and culture-specific and not necessarily direct translations of their English counterparts.

References

- Ackley, D.H., Hinton, G.E., Sejnowski, T.J., 1985. A Learning Algorithm for Boltzmann Machines*. *Cognitive Science* 9, 147–169. https://doi.org/10.1207/s15516709cog0901_7
- Ahuja, K., Diddee, H., Hada, R., Ochieng, M., Ramesh, K., Jain, P., Nambi, A., Ganu, T., Segal, S., Ahmed, M., Bali, K., Sitaram, S., 2023. MEGA: Multilingual Evaluation of Generative AI, in: *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*. 4232–4267. <https://doi.org/10.18653/v1/2023.emnlp-main.258>
- Akiba, T., Sano, S., Yanase, T., Ohta, T., Koyama, M., 2019. Optuna: A Next-generation Hyperparameter Optimization Framework, in: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 2623–2631. <https://doi.org/10.1145/3292500.3330701>
- Bandarkar, L., Liang, D., Muller, B., Artetxe, M., Shukla, S.N., Husa, D., Goyal, N., Krishnan, A., Zettlemoyer, L., Khabsa, M., 2024. The Belebele Benchmark: a Parallel Reading Comprehension Dataset in 122 Language Variants, in: *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*. 749–775. <https://doi.org/10.18653/v1/2024.acl-long.44>
- Brown, T.B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., [...], Amodei, D., 2020. Language Models are Few-Shot Learners. <https://doi.org/10.48550/ARXIV.2005.14165>
- Carlson, N.A., Burbano, V., 2026. The use of LLMs to annotate data in management research: Foundational guidelines and warnings. *Strategic Management Journal* 47, 699–725. <https://doi.org/10.1002/smj.70023>
- Chen, T., Guestrin, C., 2016. XGBoost: A Scalable Tree Boosting System, in: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 785–794. <https://doi.org/10.1145/2939672.2939785>
- Cheng, X., Mayya, R., Sedoc, J., 2024. From Human Annotation to LLMs: SILICON Annotation Workflow for Management Research. <https://doi.org/10.48550/ARXIV.2412.14461>
- Chia, Y.K., Chen, G., Tuan, L.A., Poria, S., Bing, L., 2023. Contrastive Chain-of-Thought Prompting. <https://doi.org/10.48550/ARXIV.2311.09277>
- Clark, K., Luong, M.-T., Le, Q.V., Manning, C.D., 2020. ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators. <https://doi.org/10.48550/ARXIV.2003.10555>
- Comino, S., Manenti, F.M., Mariuzzo, F., 2019. Updates management in mobile applications: iTunes versus Google Play. *Journal of Economics & Management Strategy* 28, 392–419. <https://doi.org/10.1111/jems.12288>
- de Kok, T., 2025. ChatGPT for Textual Analysis? How to Use Generative LLMs in Accounting Research. *Management Science* 71, 7888–7906. <https://doi.org/10.1287/mnsc.2023.03253>
- Devlin, J., Chang, M.-W., Lee, K., Toutanova, K., 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, in: *Proceedings of the NAACL-HLT 2019. Association for Computational Linguistics*, 4171–4186. <https://doi.org/10.18653/v1/N19-1423>
- Durvasula, M.M., Eyuboglu, S., Ritzwoller, D.M., 2024. Counting Clinical Trials: New Evidence on Pharmaceutical Sector Productivity. <https://doi.org/10.48550/ARXIV.2405.08030>
- Einhorn, H.J., 1974. Expert judgment: Some necessary conditions and an example. *Journal of Applied Psychology* 59, 562–571. <https://doi.org/10.1037/h0037164>

- Foerderer, J., Kude, T., Mithas, S., Heinzl, A., 2018. Does Platform Owner's Entry Crowd Out Innovation? Evidence from Google Photos. *Information Systems Research* 29, 444–460. <https://doi.org/10.1287/isre.2018.0787>
- Franklin, J.M., Schneeweiss, S., Polinski, J.M., Rassen, J.A., 2014. Plasmode simulation for the evaluation of pharmacoepidemiologic methods in complex healthcare databases. *Computational Statistics & Data Analysis* 72, 219–226. <https://doi.org/10.1016/j.csda.2013.10.018>
- Gioia, D.A., Corley, K.G., Hamilton, A.L., 2013. Seeking Qualitative Rigor in Inductive Research: Notes on the Gioia Methodology. *Organizational Research Methods* 16, 15–31. <https://doi.org/10.1177/1094428112452151>
- Hannigan, T.R., Haans, R.F.J., Vakili, K., Tchalian, H., Glaser, V.L., Wang, M.S., Kaplan, S., Jennings, P.D., 2019. Topic Modeling in Management Research: Rendering New Theory from Textual Data. *Academy of Management Annals* 13, 586–632. <https://doi.org/10.5465/annals.2017.0099>
- Harris, Z.S., 1954. Distributional Structure. *Word* 10, 146–162. <https://doi.org/10.1080/00437956.1954.11659520>
- Johnson, J.M., Khoshgoftaar, T.M., 2019. Survey on deep learning with class imbalance. *Journal of Big Data* 6, 27. <https://doi.org/10.1186/s40537-019-0192-5>
- Just, J., Ströhle, T., Füller, J., Hutter, K., 2024. AI-based novelty detection in crowdsourced idea spaces. *Innovation: Organization & Management* 26, 359–386. <https://doi.org/10.1080/14479338.2023.2215740>
- Kim, Y., 2014. Convolutional Neural Networks for Sentence Classification, in: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 1746–1751. <https://doi.org/10.3115/v1/D14-1181>
- Kircher, T., Foerderer, J., 2024. Ban Targeted Advertising? An Empirical Investigation of the Consequences for App Development. *Management Science* 70, 1070–1092. <https://doi.org/10.1287/mnsc.2023.4726>
- Liu, P., Yuan, W., Fu, J., Jiang, Z., Hayashi, H., Neubig, G., 2023. Pre-train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing. *ACM Computer Surveys* 55, 1–35. <https://doi.org/10.1145/3560815>
- Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., Stoyanov, V., 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. <https://doi.org/10.48550/ARXIV.1907.11692>
- Manning, C.D., Raghavan, P., Schütze, H., 2008. *Introduction to Information Retrieval*, 1st ed. Cambridge University Press. <https://doi.org/10.1017/CBO9780511809071>
- Mikolov, T., Chen, K., Corrado, G., Dean, J., 2013. Efficient Estimation of Word Representations in Vector Space. <https://doi.org/10.48550/ARXIV.1301.3781>
- Miric, M., Jia, N., Huang, K.G., 2023. Using supervised machine learning for large-scale classification in management research: The case for identifying artificial intelligence patents. *Strategic Management Journal* 44, 491–519. <https://doi.org/10.1002/smj.3441>
- Mirzadeh, S.I., Farajtabar, M., Li, A., Levine, N., Matsukawa, A., Ghasemzadeh, H., 2019. Improved Knowledge Distillation via Teacher Assistant, in: *AAAI Conference on Artificial Intelligence* 34, 5191–5198. <https://doi.org/10.1609/aaai.v34i04.5963>
- Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Müller, A., Nothman, J., Louppe, G., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas,

- J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, É., 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 2825–2830.
- Peeperkorn, M., Kouwenhoven, T., Brown, D., Jordanous, A., 2024. Is Temperature the Creativity Parameter of Large Language Models? <https://doi.org/10.48550/ARXIV.2405.00492>
- Pennington, J., Socher, R., Manning, C., 2014. GloVe: Global Vectors for Word Representation, in: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 1532–1543. <https://doi.org/10.3115/v1/D14-1162>
- Petersen, A.M., 2022. Evolution of biomedical innovation quantified via billions of distinct article-level MeSH keyword combinations. *Advances in Complex Systems* 25, 2150016. <https://doi.org/10.1142/S0219525921500168>
- Sahoo, P., Singh, A.K., Saha, S., Jain, V., Mondal, S., Chadha, A., 2024. A Systematic Survey of Prompt Engineering in Large Language Models: Techniques and Applications. <https://doi.org/10.48550/ARXIV.2402.07927>
- Strauss, A.L., Corbin, J.M., 1998. *Basics of qualitative research: techniques and procedures for developing grounded theory*, 2nd ed. SAGE, Thousand Oaks, CA.
- Tietz, M., Fan, T.J., Nouri, D., Bossan, B., skorch Developers, 2017. skorch: A scikit-learn compatible neural network library that wraps PyTorch.
- Turney, P.D., Pantel, P., 2010. From Frequency to Meaning: Vector Space Models of Semantics. *Journal of Artificial Intelligence Research* 37, 141–188. <https://doi.org/10.1613/jair.2934>
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I., 2017. Attention Is All You Need, in: *Advances in Neural Information Processing Systems*. Curran Associates, Inc.
- Vaughan, L.K., Divers, J., Padilla, M.A., Redden, D.T., Tiwari, H.K., Pomp, D., Allison, D.B., 2009. The use of plasmodes as a supplement to simulations: A simple example evaluating individual admixture estimation methodologies. *Computational Statistics & Data Analysis* 53, 1755–1766. <https://doi.org/10.1016/j.csda.2008.02.032>
- Verhoeven, D., Bakker, J., Veugelers, R., 2016. Measuring technological novelty with patent-based indicators. *Research Policy* 45, 707–723. <https://doi.org/10.1016/j.respol.2015.11.010>
- Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E., Le, Q.V., Zhou, D., 2022. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. *Advances in Neural Information Processing Systems* 35, 24824–24837.
- Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Le Scao, T., Gugger, S., Drame, M., Lhoest, Q., Rush, A., 2020. Transformers: State-of-the-Art Natural Language Processing, in: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Association for Computational Linguistics, 38–45. <https://doi.org/10.18653/v1/2020.emnlp-demos.6>
- Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R., Le, Q.V., 2019. XLNet: Generalized Autoregressive Pretraining for Language Understanding, in: *Proceedings of the 33rd International Conference on Neural Information Processing Systems*. Curran Associates Inc., 5753–5763. <https://doi.org/10.5555/3454287.3454804>
- Zhang, Z., Zhang, A., Li, M., Smola, A., 2022. Automatic Chain of Thought Prompting in Large Language Models. <https://doi.org/10.48550/ARXIV.2210.03493>

Figure B1 Data structure for software application updates (Study 1)

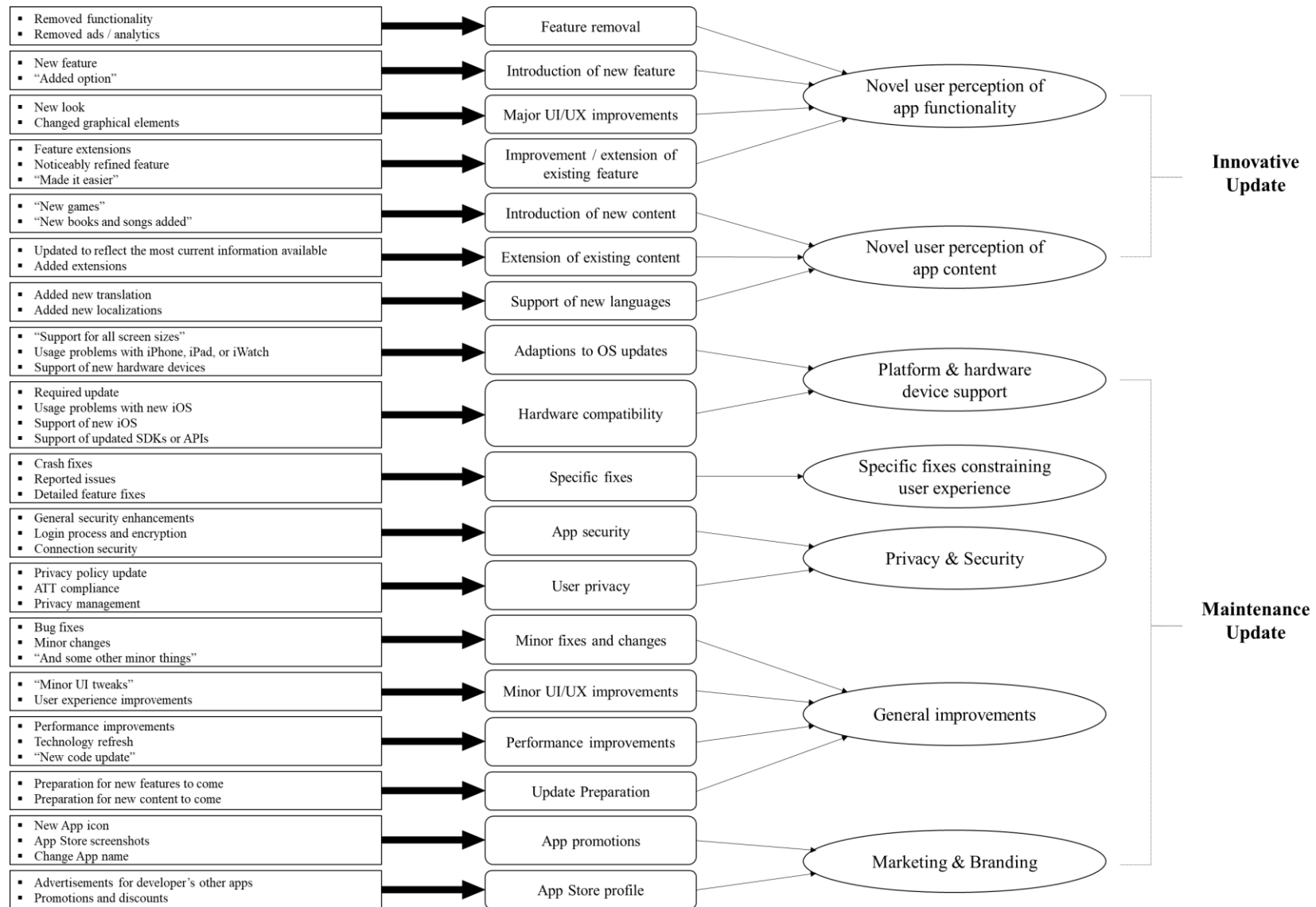


Figure B2 Data structure for product reviews (Study 2)

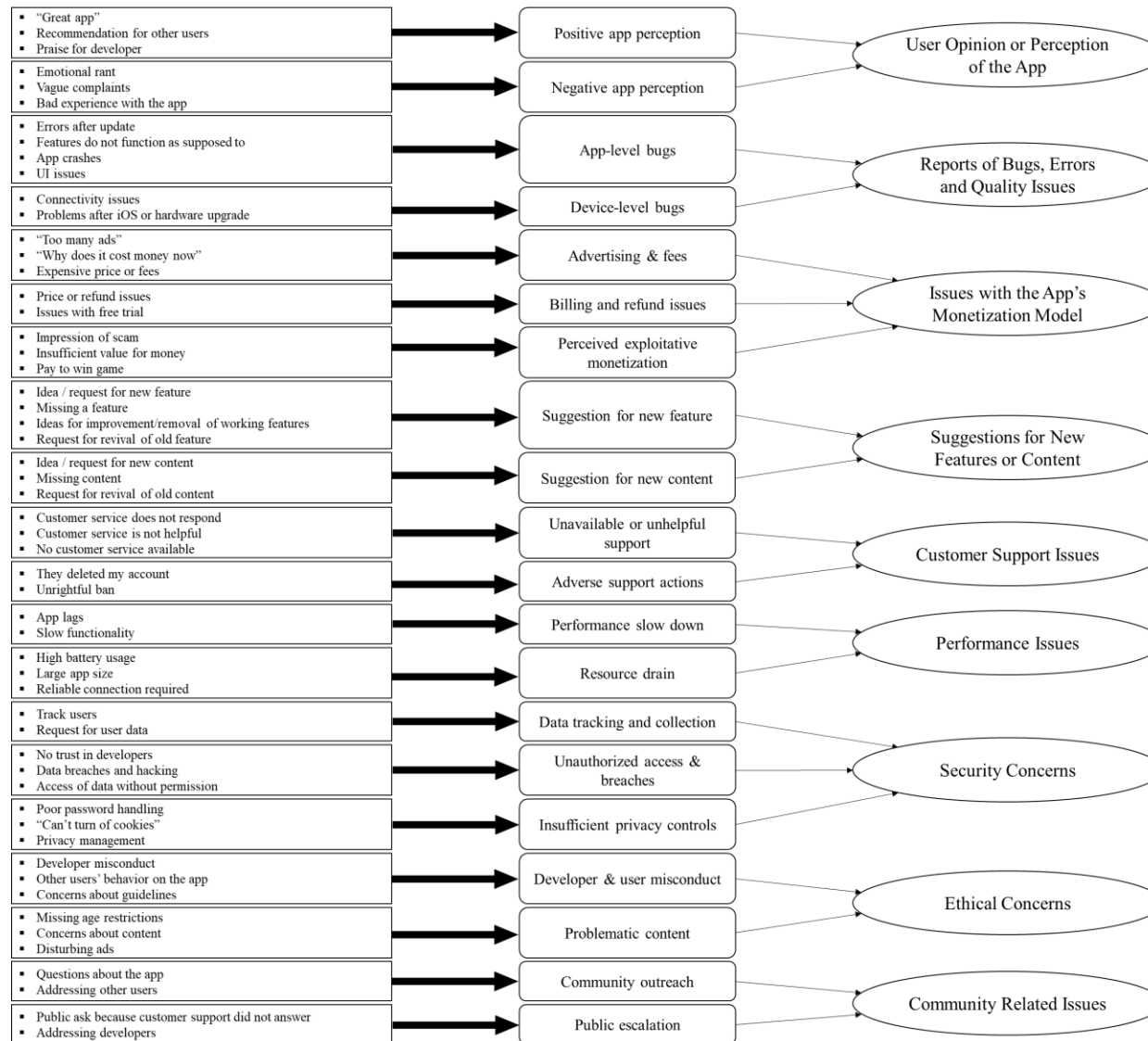
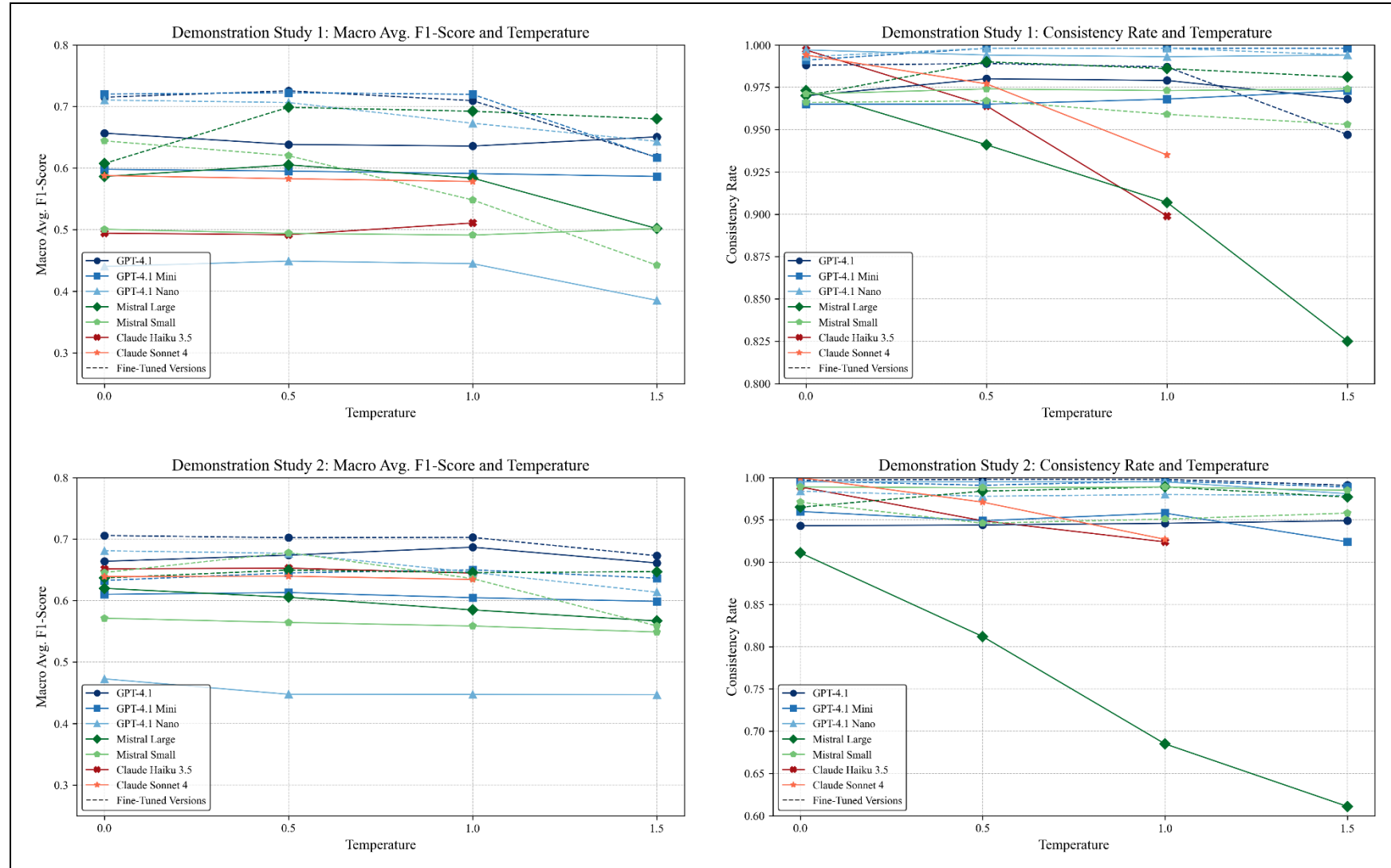
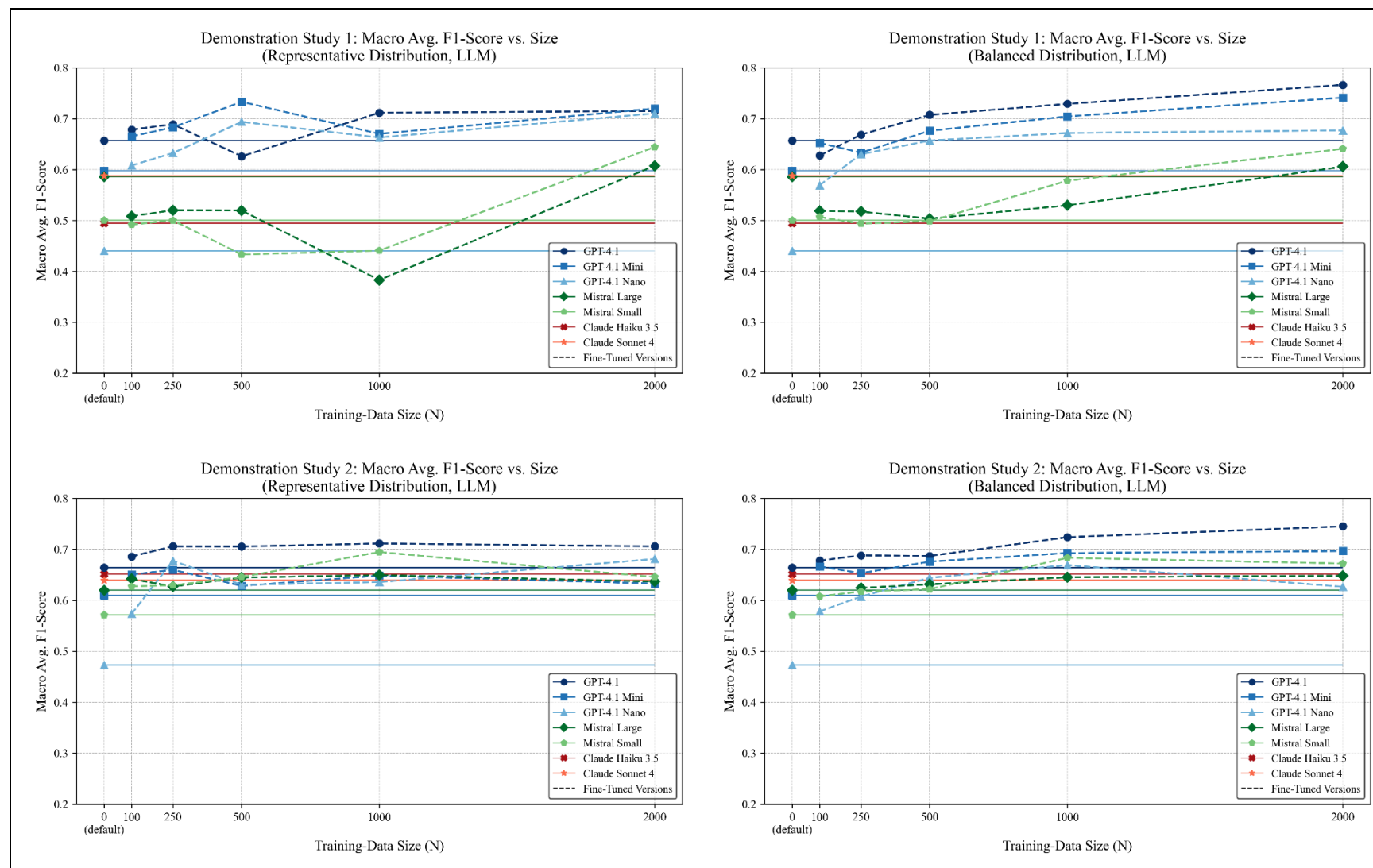


Figure E3 Model performance for classification with different temperatures



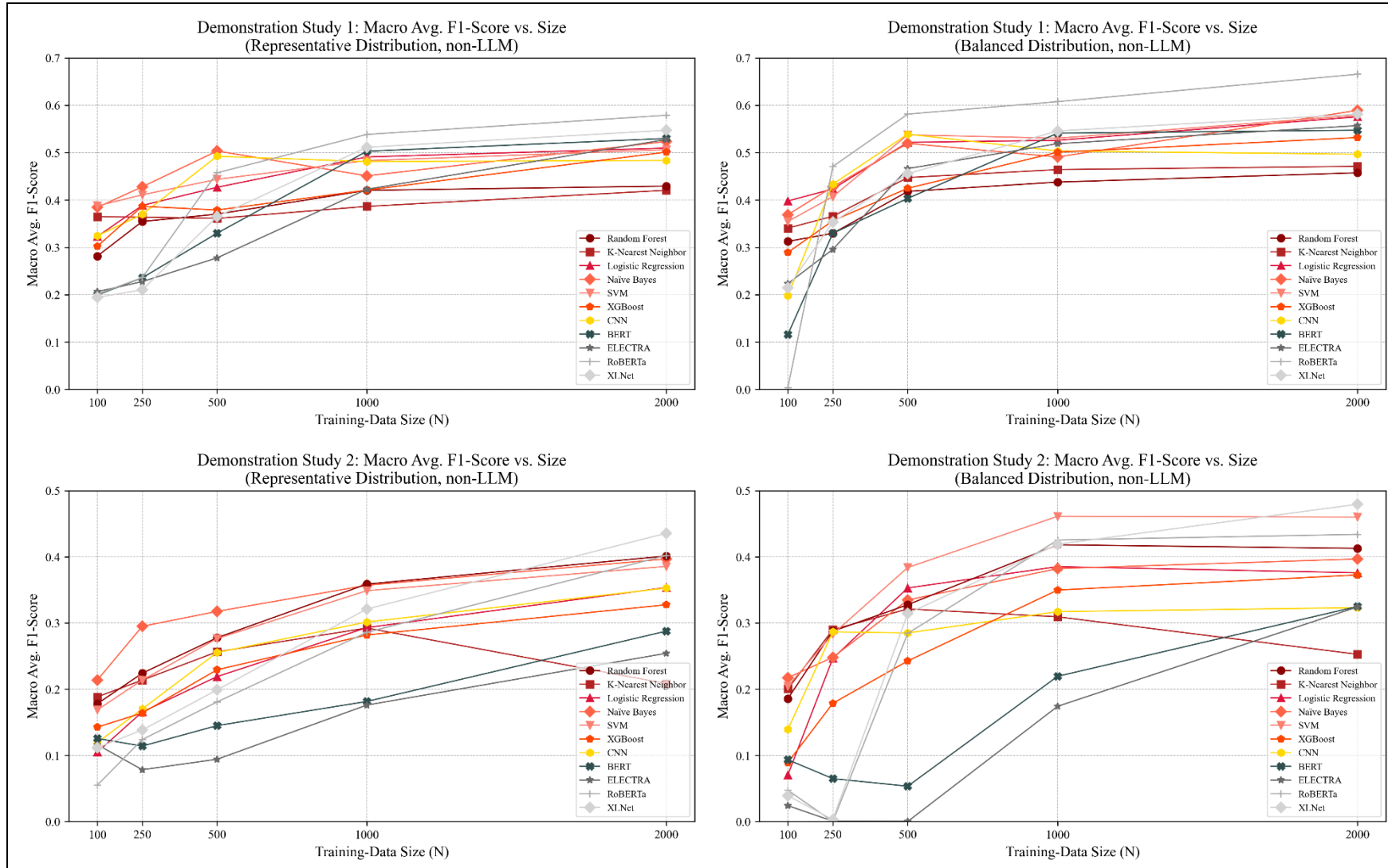
Note. LLMs were fine-tuned with a dataset $n = 2,000$, with categories distributed similarly to the validation data. F1-score is reported for the first run. Consistency Rate (CR) is based on three independent runs. Y-axes are scaled to fit best and worst model values. The maximum temperature for Claude models is limited to 1.0 per provider documentation.

Figure E4 Model performance for classification with different training data distribution and size



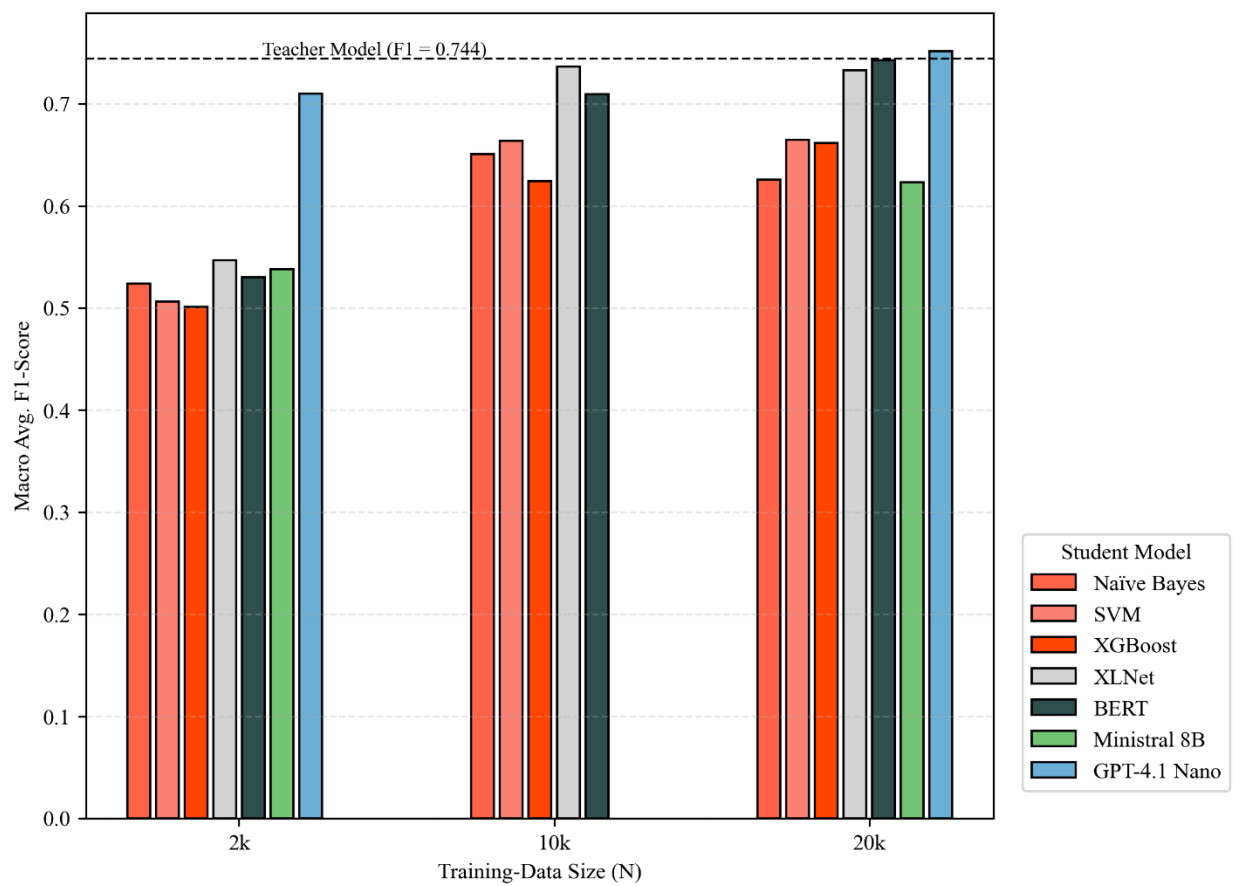
Note. F1-score is reported for the first run. Y-axes are scaled to fit best and worst model values.

Figure E5 Additional Analyses on the Role of Training Data Distribution and Data Size



Note. Embeddings for the ML classifiers were created using TF-IDF, and embeddings for the CNN were generated using glove-42b-300d. Y-axes are scaled to fit best and worst model values.

Figure F6 Model performance for teacher-student distillation setup



Note. Fine-Tuned GPT-4o Mini model is used as the teacher model. Sample sizes of $n = 10,000$ and $n = 20,000$ labels are labeled with the teacher model.

Table B1 Interviews with app developers

Interviewee	Duration
Professional Mobile App Developer (iOS & Android), Computer Scientist (M.Sc.)	31:13 min
Professional Mobile App Developer (iOS & Android), Computer Scientist (B.Sc.)	81:26 min
Professional iOS Developer, Data Scientist (M.A.)	Written survey
Professional Mobile & Web Developer, Software Engineer (Bachelor's Degree)	24:14 min
Professional Mobile App Developer (iOS & Android), Computer Scientist (B.Sc.)	18:17 min
Professional Unity Game Developer and Designer (for Mobile Games), Software Engineer (Bachelor's Degree)	26:32 min
Professional Unity Game Developer (for Mobile Games)	37:49 min

Table D2 Hyperparameter configurations and performance evaluation for ML classifiers for Demonstration Studies 1 & 2

		(1)		(2)		
		Best Performing Hyperparameter Configuration	Best CV Macro F1	HO Macro F1	Best Performing Hyperparameter Configuration	Best CV Macro F1 HO Macro F1
TF-IDF	K-Nearest-Neighbor	{"leaf_size": 38, "n_neighbors": 8, "weights": "distance", "max_df": 0.9240475867068707, "max_features": 8335, "min_df": 1, "ngram_range": [1, 3]}	.486	.420	{"leaf_size": 37, "n_neighbors": 18, "weights": "uniform", "max_df": 0.9882818784859139, "max_features": 7518, "min_df": 1, "ngram_range": [1, 1]}	.198 .208
	Logistic Regression	{"C": 6.828621268586129, "penalty": "l2", "solver": "lbfgs", "max_df": 0.996760773680571, "max_features": 9801, "min_df": 3, "ngram_range": [1, 1]}	.543	.510	{"C": 8.839465761662865, "penalty": "l2", "solver": "saga", "max_df": 0.9958605089165147, "max_features": 4093, "min_df": 4, "ngram_range": [1, 1]}	.302 .354
	Naïve Bayes	{"alpha": 0.11917071876314952, "fit_prior": false, "max_df": 0.8326423978632586, "max_features": 9392, "min_df": 3, "ngram_range": [1, 2]}	.543	.524	{"alpha": 0.11917071876314952, "fit_prior": false, "max_df": 0.8326423978632586, "max_features": 9392, "min_df": 3, "ngram_range": [1, 2]}	.380 .397
	Random Forest	{"bootstrap": false, "criterion": "entropy", "max_depth": 96, "max_features": 0.6779687087336601, "min_samples_leaf": 3, "min_samples_split": 11, "n_estimators": 448, "max_df": 0.8260307570294546, "max_features": 2630, "min_df": 1, "ngram_range": [1, 2]}	.477	.429	{"bootstrap": false, "criterion": "gini", "max_depth": 95, "max_features": 0.2077430655332096, "min_samples_leaf": 1, "min_samples_split": 8, "n_estimators": 198, "max_df": 0.6627743230167918, "max_features": 5848, "min_df": 1, "ngram_range": [1, 3]}	.344 .401
	SVM	{"C": 8.73445728344814, "degree": 2, "gamma": "auto", "kernel": "linear", "max_df": 0.685093473488098, "max_features": 5666, "min_df": 1, "ngram_range": [1, 2]}	.590	.506	{"C": 6.918621268586129, "degree": 2, "gamma": "scale", "kernel": "linear", "max_df": 0.6041717425276485, "max_features": 5270, "min_df": 4, "ngram_range": [1, 1]}	.393 .386
	XGBoost	{"colsample_bytree": 0.6965180821364413, "learning_rate": 0.07583861356997272, "max_depth": 3, "n_estimators": 445, "subsample": 0.6734468864139178, "max_df": 0.6783234961893887, "max_features": 8100, "min_df": 3, "ngram_range": [1, 3]}	.512	.501	{"colsample_bytree": 0.767077736372738, "learning_rate": 0.19276041730313465, "max_depth": 3, "n_estimators": 164, "subsample": 0.7816961974601263, "max_df": 0.6913515729152793, "max_features": 3109, "min_df": 4, "ngram_range": [1, 2]}	.294 .328
Glove	K-Nearest-Neighbor	{"vectorizer": glove-840B-300d, "leaf_size": 24, "n_neighbors": 5, "weights": "distance"}	.426	.406	{"vectorizer": glove-42B-300d, "leaf_size": 33, "n_neighbors": 4, "weights": "distance"}	.282 .337
	Logistic Regression	{"vectorizer": glove-42B-300d, "C": 6.999135822979566, "penalty": "l2", "solver": "lbfgs"}	.572	.513	{"vectorizer": glove-840B-300d, "C": 9.76042118194638, "penalty": "l2", "solver": "lbfgs"}	.340 .335
	Random Forest	{"vectorizer": glove-42B-300d, "bootstrap": false, "criterion": "gini", "max_depth": 95, "max_features": 0.2077430655332096, "min_samples_leaf": 1, "min_samples_split": 8, "n_estimators": 198}	.423	.424	{"vectorizer": glove-42B-300d, "bootstrap": false, "criterion": "entropy", "max_depth": 80, "max_features": 0.9425728949509299, "min_samples_leaf": 8, "min_samples_split": 17, "n_estimators": 326}	.263 .231
	SVM	{"vectorizer": glove-42B-300d, "C": 5.121773443923852, "degree": 4, "gamma": "auto", "kernel": "linear"}	.556	.500	{"vectorizer": glove-840B-300d, "C": 9.291710982638396, "degree": 5, "gamma": "auto", "kernel": "linear"}	.352 .358
	XGBoost	{"vectorizer": glove-840B-300d, "colsample_bytree": 0.557957280470468, "learning_rate": 0.23112312784449587, "max_depth": 4, "n_estimators": 455, "subsample": 0.6505946735495828}	.537	.503	{"vectorizer": glove-42B-300d, "colsample_bytree": 0.594405164158504, "learning_rate": 0.2845324775722167, "max_depth": 6, "n_estimators": 115, "subsample": 0.6025027592743828}	.300 .283
N			667	1,000		711 1,047

Note. ML-based classifiers were trained with a dataset $n = 2,000$ with a representative label distribution. Observation count (N) of macro averaged F1-score of cross-validated samples can vary depending on the data splits. On average, they are about one-third of the training data size.

Table D3 Hyperparameter configurations and performance evaluation for convolutional neural network for Demonstration Studies 1 & 2

(1)			(2)			
	Best Performing Hyperparameter Configuration	Best CV Macro F1	HO Macro F1	Best Performing Hyperparameter Configuration	Best CV Macro F1	HO Macro F1
Convolutional Neural Network	{“batch_size”: 60, “max_epochs”: 12, “dense_units”: 72, “dropout”: 0.46777990469323594, “kernel_size”: 6, “n_filters”: 84, “trainable”: true, “num_words”: 16763, “seq_len”: 214}	.464	.484	{“batch_size”: 58, “max_epochs”: 10, “dense_units”: 72, “dropout”: 0.40989527282572635, “kernel_size”: 5, “n_filters”: 228, “trainable”: true, “num_words”: 26815, “seq_len”: 195}	.280	.353
N		667	1,000		711	1,047

Note. CNN was trained with a dataset $n = 2,000$ with a representative label distribution. Observation count (N) of macro averaged F1-score of cross-validated samples can vary depending on the data splits. On average, they are one-third of the training data size.

Table D4 Hyperparameter configurations and performance evaluation for pre-trained language models for Demonstration Studies 1 & 2

(1)			(2)	
	Best Performing Hyperparameter Configuration	HO Macro F1	Best Performing Hyperparameter Configuration	HO Macro F1
BERT	{"learning_rate": 2.3139649033257978e-05, "batch_size": 8, "num_train_epochs": 3, "weight_decay": 0.0072173948625224625}	.530	{"learning_rate": 1.3205102630174912e-05, "batch_size": 8, "num_train_epochs": 5, "weight_decay": 0.0069800190183135335}	.288
ELECTRA	{"learning_rate": 1.2868586517068187e-05, "batch_size": 8, "num_train_epochs": 5, "weight_decay": 0.004205974315203549}	.528	{"learning_rate": 2.3139649033257978e-05, "batch_size": 8, "num_train_epochs": 3, "weight_decay": 0.0072173948625224625}	.254
RoBERTa	{"learning_rate": 1.2868586517068187e-05, "batch_size": 8, "num_train_epochs": 5, "weight_decay": 0.004205974315203549}	.579	{"learning_rate": 1.2868586517068187e-05, "batch_size": 8, "num_train_epochs": 5, "weight_decay": 0.004205974315203549}	.402
XLNet	{"learning_rate": 1.2868586517068187e-05, "batch_size": 8, "num_train_epochs": 5, "weight_decay": 0.004205974315203549}	.547	{"learning_rate": 1.3205102630174912e-05, "batch_size": 8, "num_train_epochs": 5, "weight_decay": 0.0069800190183135335}	.436
N		1,000		1,047

Note. PLMs were fine-tuned with a dataset $n = 2,000$ with a representative label distribution.

Table E5 Model performance for classification with different model provider, size, age, and reasoning capability

		(1)			(2)		
		Macro F1	Weighted F1	CR	Macro F1	Weighted F1	CR
Claude Opus 3	Anthropic	.569	.740	.991	.596	.779	.982
Claude Opus 4 (reasoning)		.511	.787	.852	.650	.823	.996
Claude Sonnet 3.5		.644	.800	.994	.649	.814	.991
Claude Sonnet 3.7		.613	.771	.996	.638	.815	.987
Claude Haiku 3		.481	.563	1.000	.533	.732	1.000
Ministral 3B	Mistral	.399	.508	.970	.483	.703	.945
Ministral 8B		.287	.378	.978	.402	.674	.983
Open Mistral Nemo		.277	.245	.880	.491	.695	.862
Mistral Medium		.594	.690	.964	.622	.806	.941
Ministral 8B (fine-tuned)		.538	.756	.989	.571	.793	.994
Open Mistral Nemo (fine-tuned)		.485	.714	.945	.556	.788	.947
o3 (reasoning)	OpenAI	.684	.809	.932	.693	.845	.935
o4 Mini (reasoning)		.580	.734	.998	.693	.855	.978
GPT-3.5 Turbo		.439	.511	.916	.564	.780	.961
GPT-4o Mini		.484	.571	.976	.576	.797	.934
GPT-4o		.602	.768	.956	.640	.813	.948
GPT-3.5 Turbo (fine-tuned)		.751	.884	.997	.677	.854	1.000
GPT-4o Mini (fine-tuned)		.744	.881	.993	.662	.858	.993
GPT-4o (fine-tuned)		.741	.889	.990	.678	.861	.993
N		1,000	1,000	3,000	1,047	1,047	3,000

Note. Transformer-based models were fine-tuned with a dataset $n = 2,000$, with categories distributed similarly to the validation data. Claude models, Ministral 3B, Mistral Medium, o3, and o4 Mini are not open for supervised fine-tuning. LLM temperature = 0.0, LLM seed = 94032.

Table E6 Variations in prompt engineering for Demonstration Study 1

Prompt Engineering Technique	Prompt Text
Few-Shot	<p>ONLY provide the category number (1-7) in response. Determine the category for the following app update text. If multiple categories seem applicable, always choose the lowest category number (1<2<3<4<5<6<7):</p> <ol style="list-style-type: none"> (1) Novel features - Introducing or enhancing significant functionalities that modify user experience (2) Content extensions – Adding or expanding curated or user-facing content (3) Platform and device support – Enabling compatibility with new OS versions, development kits, device types, or hardware features (4) Specific fixes of bugs - Addressing distinct known issues (5) Privacy & Security - Strengthening user data protection, permissions, or security protocols (6) General Improvements – Tweaks addressing undisclosed bug fixes, performance improvements, minor changes, or UI/UX polish (7) Marketing & Branding – Changes purely about promotional or visual branding assets <p>EXAMPLES: Updates: ‘new style’, ‘new features’, ‘new interface’ Response: 1</p> <p>Updates: ‘optimized performance’, ‘new levels’, ‘new stage added’ Response: 2</p> <p>Updates: ‘updated for iOS 9’, ‘resolves iOS 9 issues’, ‘adapt to iOS 12 system’ Response: 3</p> <p>Updates: ‘fix for push-messaging’, ‘fixed game freeze bug’, ‘fixed a crash that could occur when starting the game’ Response: 4</p> <p>Updates: ‘security updates’, ‘terms of use updated’, ‘added prompt to ask permission for App Tracking Transparency’ Response: 5</p> <p>Updates: ‘bug fixes’, ‘stability improvements’, ‘improved feeds’ Response: 6</p> <p>Updates: ‘new app icon introduced’, ‘screenshots for App Store added’, ‘Spring has sprung and it's time to get floral! Collect as many blossoms as you can; the more you collect, the bigger the reward!’ Response: 7</p>
Automatic Chain-of-Thought	<p>ONLY provide the category number (1-7) in response. Determine the category for the following app update text. If multiple categories seem applicable, always choose the lowest category number (1<2<3<4<5<6<7):</p> <ol style="list-style-type: none"> (1) Novel features - Introducing or enhancing significant functionalities that modify user experience (2) Content extensions – Adding or expanding curated or user-facing content (3) Platform and device support – Enabling compatibility with new OS versions, development kits, device types, or hardware features (4) Specific fixes of bugs - Addressing distinct known issues (5) Privacy & Security – Strengthening user data protection, permissions, or security protocols (6) General Improvements – Tweaks addressing undisclosed bug fixes, performance improvements, minor changes, or UI/UX polish (7) Marketing & Branding – Changes purely about promotional or visual branding assets <p>Let’s think step by step.</p>
Manual Chain-of-Thought	<p>ONLY provide the category number (1-7) in response. Determine the category for the following app update text. If multiple categories seem applicable, always choose the lowest category number (1<2<3<4<5<6<7):</p> <ol style="list-style-type: none"> (1) Novel features - Introducing or enhancing significant functionalities that modify user experience (2) Content extensions – Adding or expanding curated or user-facing content (3) Platform and device support – Enabling compatibility with new OS versions, development kits, device types, or hardware features (4) Specific fixes of bugs – Addressing distinct known issues (5) Privacy & Security - Strengthening user data protection, permissions, or security protocols (6) General Improvements – Tweaks addressing undisclosed bug fixes, performance improvements, minor changes, or UI/UX polish

(7) Marketing & Branding – Changes purely about promotional or visual branding assets

Let's think step by step.

ANALYZE STEP BY STEP:

1. Read the app update text carefully
2. Identify the main purpose of the update
3. Check the different categories starting from 1 to 7 and determine which category applies best
4. If multiple categories apply, choose the category with the lowest number
5. Provide the number of the category as response

EXAMPLE:

Update text: 'game runs faster and more levels have been added'

1. Reading of update text 'game runs faster and more levels have been added'
2. Main purpose of 'game runs faster' is performance improvement and purpose of 'more levels have been added' is extensions of app content
3. Performance improvement aligns with category 6, extensions of app content aligns with category 2
4. Lower category number 2 is chosen
5. 2 is provided as response

ONLY provide the category number (1-7) in response. Determine the category for the following app update text. If multiple categories seem applicable, always choose the lowest category number (1<2<3<4<5<6<7):

- (1) Novel features - Introducing or enhancing significant functionalities that modify user experience
- (2) Content extensions – Adding or expanding curated or user-facing content
- (3) Platform and device support – Enabling compatibility with new OS versions, development kits, device types, or hardware features
- (4) Specific fixes of bugs – Addressing distinct known issues
- (5) Privacy & Security - Strengthening user data protection, permissions, or security protocols
- (6) General Improvements – Tweaks addressing undisclosed bug fixes, performance improvements, minor changes, or UI/UX polish
- (7) Marketing & Branding – Changes purely about promotional or visual branding assets

Let's think step by step.

ANALYZE STEP BY STEP:

1. Read the app update text carefully
2. Identify the main purpose of the update
3. Check the different categories starting from 1 to 7 and determine which category applies best
4. If multiple categories apply, choose the category with the lowest number
5. Provide the number of the category as response

Contrastive
Chain-of-
Thought

CORRECT EXAMPLE:

Update text: 'game runs faster and more levels have been added'

1. Reading of update text 'game runs faster and more levels have been added'
2. Main purpose of 'game runs faster' is performance improvement and purpose of 'more levels have been added' is extensions of app content
3. Performance improvement aligns with category 6, extensions of app content aligns with category 2
4. Lower category number 2 is chosen
5. 2 is provided as response"

WRONG EXAMPLE:

Update text: 'game runs faster and more levels have been added'

1. Reading of update text 'game runs faster and more levels have been added'
 2. Main purpose of 'game runs faster' is extensions/additions of app content and purpose of 'more levels have been added' is privacy & security
 3. Extensions/additions of app content aligns with category 6, privacy and security aligns with category 2
 4. Lower category number 2 is chosen
 5. 1 is provided as response
-

Table E7 Variations in prompt engineering for Demonstration Study 2

Prompt Engineering Technique	Prompt Text
Few-Shot	<p>ONLY provide a number (0-8) in response. Categorize the following app review text by assigning the most fitting category/categories out of the following nine categories. If the text contains elements from multiple categories, provide the categories separated by ; .</p> <ul style="list-style-type: none"> (0) User Opinion without specific reports, issues, suggestions - i.e., review only about good/bad perception of the app but nothing else (1) Reports of bugs, errors, or bad quality issues - i.e., something does not work in the app or is of bad quality (2) Issues of the app's monetization model - i.e., complaints or issues of how the app monetizes content (3) Suggestions for new features or content or revival of removed features (4) Customer support issues - i.e., problems or complaints regarding customer support (5) Performance issues - i.e., the app needs too much space, is too slow or similar (6) Security concerns - i.e., user is concerned about their data or privacy (7) Ethical concerns - i.e., user is concerned about practices in the app, fairness, discrimination (8) Community related issues - user asks openly for help (no feedback at customer support) <p>EXAMPLES:</p> <p>Review texts: 'I love this game', 'clear, friendly, effective', 'horrible app; if I could give these creators 0 stars, I would'</p> <p>Response: 0</p> <p>Review texts: 'I'm getting error messages when trying to use the app', 'makes my phone incredibly hot, then the game crashes', 'I am unable to access the app'</p> <p>Response: 1</p> <p>Review texts: 'the items should be at a lower price', 'the apps are unbelievably disturbing', 'the app hides the ability to cancel the subscription'</p> <p>Response: 2</p> <p>Review texts: 'add home screen widgets', 'please support on iPad', 'create stories'</p> <p>Response: 3</p> <p>Review texts: 'there is no solution even though I reported the issue already', 'nobody replies from support', 'I cannot contact the support'</p> <p>Response: 4</p> <p>Review texts: 'the app drains 50% battery overnight', 'updating the app takes too long', 'the game lagged a bit'</p> <p>Response: 5</p> <p>Review texts: 'do not trust anything about this app, it's a scam', 'they are selling our privacy', 'app reads and saves photos'</p> <p>Response: 6</p> <p>Review texts: 'the community tends to be very toxic towards each other', 'the app is debased by its selective censorship', 'this app clearly promotes sex trafficking'</p> <p>Response: 7</p> <p>Review texts: 'how can I change the language', 'hey guys, I have no idea what the color paints are for', 'please unfreeze my account and support me'</p> <p>Response: 8</p>
Automatic Chain-of-Thought	<p>Only provide a number (0-8) as response. Categorize the following app review text based on the most applicable out of the following 9 categories. If multiple categories apply, provide the categories separated by ; .</p> <ul style="list-style-type: none"> (0) User Opinion without specific reports, issues, suggestions - i.e., review only about good/bad perception of the app but nothing else (1) Reports of bugs, errors, or bad quality issues - i.e., something does not work in the app or is of bad quality (2) Issues of the app's monetization model - i.e., complaints or issues of how the app monetizes content (3) Suggestions for new features or content or revival of removed features (4) Customer support issues - i.e., problems or complaints regarding customer support (5) Performance issues - i.e., the app needs too much space, is too slow or similar (6) Security concerns - i.e., user is concerned about their data or privacy (7) Ethical concerns - i.e., user is concerned about practices in the app, fairness, discrimination (8) Community related issues - user asks openly for help (no feedback at customer support) <p>Let's think step by step.</p>
Manual Chain-of-Thought	<p>Only provide a number (0-8) as response. Categorize the following app review text based on the most applicable out of the following 9 categories. If multiple categories apply, provide the categories separated by ; .</p> <ul style="list-style-type: none"> (0) User Opinion without specific reports, issues, suggestions - i.e., review only about good/bad perception of the app but nothing else (1) Reports of bugs, errors, or bad quality issues - i.e., something does not work in the app or is of bad quality (2) Issues of the app's monetization model - i.e., complaints or issues of how the app monetizes content (3) Suggestions for new features or content or revival of removed features (4) Customer support issues - i.e., problems or complaints regarding customer support

	<p>(5) Performance issues - i.e., the app needs too much space, is too slow or similar</p> <p>(6) Security concerns - i.e., user is concerned about their data or privacy</p> <p>(7) Ethical concerns - i.e., user is concerned about practices in the app, fairness, discrimination</p> <p>(8) Community related issues - user asks openly for help (no feedback at customer support)</p> <p>ANALYZE STEP BY STEP:</p> <ol style="list-style-type: none"> 1. Read the app review text carefully 2. Identify the main purpose of the review 3. Check the different categories starting from 0 to 8 and determine which category/categories applies/apply best 4. Provide the number(s) of the category/categories as response <p>EXAMPLE:</p> <p>Review text: 'Trying to get an answer about anonymity. Can a winner stay anonymous? They have my name and address. Emailed for an answer- no response. Reading of update text 'game runs faster and more levels have been added'</p> <ol style="list-style-type: none"> 1. Reading of update text 'Trying to get an answer about anonymity. Can a winner stay anonymous? They have my name and address. Emailed for an answer- no response.' 2. Main purpose of 'trying to get an answer' and 'Emailed for an answer - no response' is customer support issue and purpose of 'anonymity' and 'They have my name and address' is security concerns 3. Customer support issue aligns with category 4, security concerns aligns with category 6 4. 4;6 are provided as response
Contrastive Chain-of-Thought	<p>Only provide a number (0-8) as response. Categorize the following app review text based on the most applicable out of the following 9 categories. If multiple categories apply, provide the categories separated by ; .</p> <ol style="list-style-type: none"> (0) User Opinion without specific reports, issues, suggestions - i.e., review only about good/bad perception of the app but nothing else (1) Reports of bugs, errors, or bad quality issues - i.e., something does not work in the app or is of bad quality (2) Issues of the app's monetization model - i.e., complaints or issues of how the app monetizes content (3) Suggestions for new features or content or revival of removed features (4) Customer support issues - i.e., problems or complaints regarding customer support (5) Performance issues - i.e., the app needs too much space, is too slow or similar (6) Security concerns - i.e., user is concerned about their data or privacy (7) Ethical concerns - i.e., user is concerned about practices in the app, fairness, discrimination (8) Community related issues - user asks openly for help (no feedback at customer support) <p>ANALYZE STEP BY STEP:</p> <ol style="list-style-type: none"> 1. Read the app review text carefully 2. Identify the main purpose of the review 3. Check the different categories starting from 0 to 8 and determine which category/categories applies/apply best 4. Provide the number(s) of the category/categories as response <p>CORRECT EXAMPLE:</p> <p>Review text: 'Trying to get an answer about anonymity. Can a winner stay anonymous? They have my name and address. Emailed for an answer- no response. Reading of update text 'game runs faster and more levels have been added'</p> <ol style="list-style-type: none"> 1. Reading of update text 'Trying to get an answer about anonymity. Can a winner stay anonymous? They have my name and address. Emailed for an answer- no response.' 2. Main purpose of 'trying to get an answer' and 'Emailed for an answer - no response' is customer support issue and purpose of 'anonymity' and 'They have my name and address' is security concerns 3. Customer support issue aligns with category 4, security concerns aligns with category 6 4. 4;6 are provided as response <p>WRONG EXAMPLE:</p> <p>Review text: 'Trying to get an answer about anonymity. Can a winner stay anonymous? They have my name and address. Emailed for an answer- no response. Reading of update text 'game runs faster and more levels have been added'</p> <ol style="list-style-type: none"> 1. Reading of update text 'Trying to get an answer about anonymity. Can a winner stay anonymous? They have my name and address. Emailed for an answer- no response.' 2. Main purpose of 'trying to get an answer' and 'Emailed for an answer - no response' is security concerns and purpose of 'anonymity' and 'They have my name and address' is ethical concerns 3. Security concerns aligns with category 4, ethical concerns aligns with category 6 4. 1;7 are provided as response

Table E8 Model performance for classification with different prompt designs

		(1)			(2)		
		Macro F1	Weighted F1	CR	Macro F1	Weighted F1	CR
GPT-4.1 Nano	Few-Shot	.445	.535	.997	.586	.784	.986
GPT-4.1 Mini		.621	.783	.954	.610	.813	.927
GPT-4.1		.658	.822	.975	.662	.831	.952
GPT-4.1 Nano (fine-tuned)		.650	.856	.995	.660	.842	.987
GPT-4.1 Mini (fine-tuned)		.724	.866	.999	.663	.856	.988
GPT-4.1 (fine-tuned)		.739	.881	.987	.695	.866	.985
GPT-4.1 Nano	Chain-of-Thought (Automatic)	.428	.485	.996	.482	.753	.991
GPT-4.1 Mini		.596	.751	.962	.621	.806	.946
GPT-4.1		.647	.821	.977	.672	.837	.961
GPT-4.1 Nano (fine-tuned)		.593	.858	.993	.653	.840	.984
GPT-4.1 Mini (fine-tuned)		.721	.869	.999	.645	.843	.992
GPT-4.1 (fine-tuned)		.710	.874	.991	.702	.860	.991
GPT-4.1 Nano	Chain-of-Thought (Manual)	.411	.415	.997	.470	.722	.980
GPT-4.1 Mini		.577	.723	.957	.621	.798	.958
GPT-4.1		.631	.808	.976	.674	.831	.926
GPT-4.1 Nano (fine-tuned)		.554	.841	.994	.662	.834	.982
GPT-4.1 Mini (fine-tuned)		.719	.872	.995	.647	.850	.993
GPT-4.1 (fine-tuned)		.722	.876	.983	.691	.863	.992
GPT-4.1 Nano	Chain-of-Thought (Contrastive)	.341	.355	.992	.468	.725	.978
GPT-4.1 Mini		.603	.737	.947	.616	.802	.956
GPT-4.1		.671	.823	.985	.684	.831	.940
GPT-4.1 Nano (fine-tuned)		.649	.840	.993	.676	.841	.982
GPT-4.1 Mini (fine-tuned)		.729	.873	.996	.645	.854	.994
GPT-4.1 (fine-tuned)		.748	.882	.987	.710	.867	.990
N		1,000	1,000	3,000	1,047	1,047	3,000

Note. Transformer-based models were fine-tuned with a dataset $n = 2,000$, with categories distributed similarly to the validation data. LLM temperature = 0.0, LLM seed = 94032. See Appendix E2 for prompts.

Table G9 When to fine-tune an LLM for framework application

Task or deployment characteristic	Ex ante question	Recommendation	Exemplary Demonstration Study Result
Classification with common, balanced classes	Does the default configuration already meet the required performance on the holdout sample?	Default suffices; fine-tuning yields marginal gains that are unlikely to alter substantive conclusions	Study 1, Table 2 (GPT-4.1 default F1 = .901 vs. .926 fine-tuned)
Limited expert-validated training data	Is the available training volume below the model-specific threshold at which fine-tuning pays off?	Do not fine-tune; below this volume, fine-tuned models can underperform their own default configurations	Appendix E4, Figure E4 (GPT-4.1 below its default at n = 500, Mistral models until n = 2,000)
Default configuration misses or under-recalls construct-relevant instances	Does the default configuration miss rare classes or under-recall relevant instances on the holdout sample?	Fine-tune; raises recall and per-class performance, verified per class on the holdout sample	Study 1, Table 2 (GPT-4.1 recall .879 to .961); Study 2, Table 3 (e.g., GPT-4.1 class 4 F1 .727 to .807)
Local, scalable, or API-independent deployment	Must inference run locally or at low recurring cost?	Fine-tune a student model on teacher-generated labels	Online Appendix F (teacher–student distillation)

Table H10 Downstream estimates with innovation-related updates as regressor (case i)

Configuration	F1	Estimated β	95% simulation interval	$\beta^{\text{FN only}}$	$\beta^{\text{FP only}}$
GPT-4.1 default	.901	.416	[.341, .483]	.457	.468
GPT-4.1 fine-tuned	.926	.430	[.359, .502]	.484	.449
Classical ML (best)	.864	.373	[.298, .443]	.470	.415
PLM (BERT)	.887	.395	[.317, .467]	.474	.429

Note. Estimated coefficients of innovation-related updates on the synthetic outcome ($\beta = .50$). The simulation interval spans the 2.5th to 97.5th percentile of the coefficient distribution across 1,000 simulated outcome realizations. F1 refers to the binary innovation indicator on the holdout sample. $\beta^{\text{FN only}}$ retains a configuration's false negatives and corrects its false positives; $\beta^{\text{FP only}}$ retains its false positives and corrects its false negatives.

Table H11 Measured pre-post change with innovation-related updates as outcome (case ii)

Configuration	Measured Δ (pp)	Distortion (pp)	95% paired bootstrap CI	Baseline error (pp)	FPR drift (pp)	FNR drift (pp)
GPT-4.1 default	-1.58	-1.52	[-5.20, 2.18]	.01	-.05	-1.48
GPT-4.1 fine-tuned	-2.72	-2.66	[-5.94, .49]	.01	-1.65	-1.02
Classical ML (best)	1.13	1.20	[-3.37, 5.47]	.02	.07	1.11
PLM (BERT)	-.14	-.08	[-4.27, 4.06]	.01	-.13	.04

Note. Change in the measured share of innovation-related updates between the earlier and later holdout period, in percentage points (pp). Distortion is the measured change minus the expert-labeled change of $-.06$ pp. The confidence interval spans the 2.5th to 97.5th percentile of the distortion across 2,000 paired bootstrap resamples stratified by period. Baseline error, false-positive rate (FPR) drift, and false-negative rate (FNR) drift decompose the distortion into attenuation from period-invariant error rates and the contributions of changes in each error rate between periods.

Table I12 Model performance for classification with most recent model generation

		(1)			(2)		
		Macro F1	Weighted F1	CR	Macro F1	Weighted F1	CR
Claude Opus 4.6 (reasoning)	Anthropic	.636	.809	.993	.605	.813	.966
Claude Sonnet 4.5		.612	.751	.991	.636	.811	.980
Claude Haiku 4.5		.576	.742	.997	.605	.801	.984
Mistral Large 3	M	.512	.667	.967	.609	.802	.920
GPT-5.2	OpenAI	.642	.822	.970	.676	.841	.979
GPT-5 Mini		.520	.635	.817	.667	.843	.901
GPT-5 Nano		.467	.516	.879	.675	.838	.829
N		1,000	1,000	3,000	1,047	1,047	3,000

Note. Recent frontier models are not open for supervised fine-tuning. LLM temperature = 0.0, except for GPT-5 Mini and GPT-5 Nano which only allowed temperature = 1.0. LLM seed = 94032.

Table J13 Scope conditions for the LLM-based framework

Task characteristic	Ex ante question	Recommended approach	Example settings
<i>A. Is the construct manifest in the text?</i>			
Textual cues carry the construct	Would a domain expert reach a judgment reading only the focal text?	The presented framework is well suited.	Release notes, product reviews, idea submissions, changelogs, grant abstracts
Construct requires verification beyond the focal text	Does the judgment require comparison against prior art or an external corpus?	Combine with corpus-comparison methods (e.g., Verhoeven et al., 2016)	Patent-style prior-art assessment
Construct is defined relationally across documents	Is the construct a property of the focal text relative to the full corpus rather than of the focal text alone?	Combine with embedding- or similarity-based methods (e.g., Just et al., 2024)	Novelty operationalized as distance to all other contributions
Construct requires non-linguistic evidence	Do experts rely on visual, auditory, or physical artifacts to judge the construct?	Currently unsuitable; multimodal extensions are a question for future research	Visual design novelty, aesthetic quality
Text producers strategically misrepresent	Do text producers have incentives and the ability to game descriptions in ways indistinguishable within the text?	Unsuitable where misrepresentation leaves no textual trace (Comino et al., 2019); monitor drift when the measure becomes a target	Promotional release notes; competitive funding pitches
<i>B. Can reliable expert ground truth be established?</i>			
Experts converge in their judgments	Do experts identify similar cues?	Unsuitable if reliability is low; no benchmark exists for validation (Einhorn, 1974)	Genuinely contested constructs without expert consensus
Domain experts are accessible	Can a small group of experts validate the data structure?	Required input to the framework	All applications
Target text corpus is temporally, culturally, or linguistically distant from the majority of the model’s training data	Can an expert-validated holdout sample be established for the target period and domain?	Suitable with holdout validation; validity does not transfer from contemporary text corpora	Historical documents; non-Western text corpora
<i>C. Is expert-level judgment required at all?</i>			
Construct is latent, language	Do dictionaries and classical ML fail to reach acceptable	Comparative advantage of the framework	Unstandardized descriptive text

heterogeneous, labeled data scarce	performance at feasible labeling cost?		
Construct resides in surface features, labels are abundant	Is terminology standardized, or is a large labeled dataset already available?	Suitable but dominated on cost by dictionaries or classical ML	Well-defined terminology, established labeled benchmarks
