

The Overlooked Spirit of Open Source: LLM-Based Analysis of Non-Explicit Mentoring in OSS Communities

Xing Cui 

Institute of Software, Chinese Academy of Sciences, Beijing, China
University of Chinese Academy of Sciences, Beijing, China

Jingzheng Wu 

Institute of Software, Chinese Academy of Sciences, Beijing, China

Tianyue Luo 

Institute of Software, Chinese Academy of Sciences, Beijing, China

Xiang Ling 

Institute of Software, Chinese Academy of Sciences, Beijing, China

Abstract

Background. The sustainable evolution of Open Source Software (OSS) depends not only on code production but also on continuous knowledge transfer and collaborative support within communities. However, existing contribution metrics primarily center on code commits and merges, overlooking the value of non-coding activities embedded in technical discussions. Among these activities, Non-explicit Mentoring (NEM) refers to guidance-oriented interactions in Pull Request (PR) and Issue discussions, where contributors convey technical knowledge, facilitate problem diagnosis, clarify design rationale, provide normative guidance, and share feedback within natural conversational contexts. Although prior studies acknowledge the existence of such behaviors, systematic quantitative investigations of their scale, structural patterns, and impact remain limited due to reliance on qualitative methods, insufficient automated detection techniques, and the scarcity of high-quality annotated datasets.

Aims. To address these challenges, this paper proposes MentoScope, a large language model (LLM)-based automated framework for fine-grained identification and classification of NEM in GitHub collaboration contexts.

Method. Built upon Llama-3.1-8B, MentoScope undergoes a three-stage training pipeline. First, continual pre-training (CPT) with LoRA is applied on 92,778 PRs and 118,569 Issues for OSS domain adaptation. Second, supervised fine-tuning (SFT) on 15,214 high-quality annotated samples enables recognition of 8 predefined NEM categories. Third, Odds Ratio Preference Optimization (ORPO) with 3,280 preference pairs aligns model outputs with human judgment to enhance classification robustness.

Results. MentoScope achieves F1-scores of 94.20% and 80.13% on NEM identification and classification tasks respectively, significantly outperforming all baselines, with ablation studies confirming the necessity of each training stage. Large-scale analysis of 591,154 comments from 500 OSS projects reveals that NEM is present in over 70% of collaborative interactions, with distributional patterns varying systematically across project scales and discussion contexts. Furthermore, survival analysis on 1,245 newcomer contributors shows that NEM recipients achieve a 12.6% improvement in PR merge rates and a 35% extension in average active tenure compared to the control group.

Conclusion. These findings demonstrate that NEM constitutes a prevalent and impactful collaborative behavior in open source communities, with significant implications for newcomer integration and community sustainability.

2012 ACM Subject Classification Software and its engineering → Empirical software validation

Keywords and phrases open source software, non-explicit mentoring, large language models, contributor retention



© Xing Cui, Jingzheng Wu, Tianyue Luo, and Xiang Ling;
licensed under Creative Commons License CC-BY 4.0

20th International Symposium on Empirical Software Engineering and Measurement (ESEM 2026).

Editors: Robert Feldt, Maria Paasivaara, Daniel Mendez, Stefan Wagner, and Marvin Muñoz Barón;

Article No. 43; pp. 43:1–43:20



Leibniz International Proceedings in Informatics

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Digital Object Identifier 10.4230/LIPIcs.ESEM.2026.43

Category Technical Track Paper

Funding This work is supported by the National Key R&D Program of China No. 2024YFB3108000, the Beijing Nova Program No. 202604841307, and the YuanTu Large Research Infrastructure.

1 Introduction

OSS constitutes a cornerstone of modern digital infrastructure, whose sustainable evolution depends not only on code production but also on continuous knowledge transfer and collaborative support among community members [8, 47]. However, mainstream code hosting platforms such as GitHub and their associated analytics tools predominantly adopt code-centric contribution metrics, prioritizing quantifiable artifacts such as code merges and commit frequency while systematically overlooking non-coding activities that are critical to community health and developer growth [44]. This structural bias risks contributor burnout, impedes newcomer integration, and ultimately threatens the long-term resilience and sustainability of OSS communities [40].

While non-coding contributions encompass activities such as fundraising, community management, and outreach, these largely occur outside code hosting platforms and are difficult to track at scale. To ensure observability and reproducibility, this study focuses on GitHub interactions, centering on a valuable but overlooked contribution form, namely NEM. We define NEM as guidance-oriented behavior embedded in PR and Issue discussions through which contributors facilitate the technical growth and community integration of others. Unlike explicit mentoring through formal programs or onboarding documentation, NEM manifests within routine code review and technical discourse as fine-grained implicit knowledge transfer, comprising architectural rationale articulation, heuristic debugging guidance, and norm transmission through constructive feedback. Although present in publicly accessible discussion text, NEM remains undetectable by existing rule-based or coarse-grained approaches due to its semantic overlap with routine technical clarification, leaving it unrecognized within prevailing contribution assessment frameworks.

Researchers increasingly recognize that NEM knowledge transfer in OSS technical discussions plays an important role in community functioning and contributor development. Young et al. [44] broaden the understanding of OSS contributions through the “All Contributors” model, while Meluso et al. [29] and Robin et al. [22] show that support-oriented activities beyond software artifacts remain essential to community sustainability. At the role level, Trinkenreich et al. [42] and Javier et al. [2] highlight the indispensable role of non-coding contributions in project development. More directly, Feng et al. [14] find that implicit mentoring appears in 27.41% of PRs, indicating growing attention to guidance-oriented behaviors in OSS technical interactions. The CHAOSS project [3, 4] further incorporates mentorship and knowledge transfer into community health metrics. Collectively, these studies underscore the importance of NEM in open source communities.

Despite the recognized significance of NEM, systematic investigation remains constrained by several critical challenges. Methodologically, reliance on surveys and interviews [14, 22, 42] introduces subjectivity and limits scalability within GitHub’s large-scale asynchronous environment. Technically, existing rule-based or conventional machine learning approaches [14, 44] lack the semantic precision to decode nuanced mentoring behaviors in complex technical discourse. Furthermore, prior research often operates at a coarse-grained level by focusing on implicit roles [2, 42] instead of specific behavioral types. Ultimately, the scarcity of high-quality,

fine-grained annotated benchmarks continues to hinder robust empirical analysis. Recent advances in LLM offer a promising solution for automating mentoring identification in large-scale GitHub discussions due to their robust semantic reasoning [7]. However, general-purpose LLMs often lack the deep understanding of OSS technical conventions and community norms required for direct application.

To address these challenges, we propose MentoScope, an LLM-based automated analysis framework that integrates a multi-stage training strategy for fine-grained identification and classification of NEM in GitHub PR and Issue comments. First, we perform CPT [50] on Llama-3.1-8B [31] using large-scale OSS discussion data to capture domain-specific linguistic patterns. Second, we introduce a classification scheme of NEM comprising 8 fine-grained categories and construct a manually annotated dataset to support SFT [12] of the model. Third, we apply ORPO [20] to align model outputs with expert judgments. Through this design, MentoScope not only detects the presence of NEM but also identifies its specific types, enabling technical support for studying knowledge transfer and contribution behaviors in OSS communities. MentoScope is evaluated on a labeled dataset of 15,214 PR and Issue comments, achieving F1-scores of 94.20% for NEM identification and 80.13% for fine-grained classification, significantly outperforming all baselines. Applied to 591,154 comments from 500 Github OSS projects, large-scale analysis reveals that NEM appears in over 70% of community interactions, with distributional variation across project scales and interaction contexts. Survival analysis and non-parametric tests on 1,245 newcomers further show that NEM recipients achieve a 12.6% improvement in PR merge rates and a 35% extension in average active periods relative to the control group, providing strong evidence that NEM facilitates knowledge transfer and drives sustainable open source community development.

Our contributions are as follows:

- (1) We introduce a fine-grained classification scheme of 8 NEM behavioral categories in GitHub collaborations, establishing a standardized foundation for quantifying and analyzing mentoring dynamics in OSS communities.
- (2) We propose MentoScope, a three-stage LLM training framework achieving 94.20% F1-score on binary NEM identification and 80.13% on fine-grained classification, substantially outperforming all baselines, with ablation studies validating each training stage.
- (3) Large-scale analysis of 591,154 comments from 500 OSS projects reveals that NEM appears in over 70% of interactions, with prevalence varying by project scale and discussion context. Survival analysis further shows that newcomers receiving NEM exhibit a 12.6% increase in PR merge rates and a 35% longer active participation period, highlighting NEM's role in facilitating knowledge transfer and community sustainability.

2 Background

2.1 Contribution recognition in open source communities

Open source community sustainability depends on sustained contributor engagement, prompting communities to establish recognition mechanisms for identifying and rewarding contributions. GitHub, the largest code hosting platform, centers its recognition system on code contributions, defining contributors as users who commit to a repository's default branch via a GitHub-associated email, and ranking the top 100 contributors by commit frequency on its Insights page. While effective for quantifying explicit code contributions, these mechanisms fail to capture non-code contributions [29]. CHAOSS, launched by the Linux Foundation in 2017, addresses this gap through a comprehensive community health framework spanning 10 metric themes including contributor definition and contribution diversity [4], but its reliance on qualitative surveys and self-reported data limits scalability and applicability to large unstructured datasets.

2.2 Mentorship in open source communities

Knowledge transfer and newcomer integration are vital to OSS sustainability, with mentoring playing a central role [13, 15]. Mentoring in OSS communities takes two forms: explicit and NEM. Explicit mentoring involves formal mentor-apprentice relationships with clear roles, goals, and timelines, typically organized by communities or external organizations [16]. Representative programs include the Linux Foundation Mentorship Program and Google Summer of Code (GSoC) [23, 41], where participants receive dedicated one-on-one guidance. CHAOSS defines mentorship metrics to assess such programs' impact on participation, retention, diversity, and career development. NEM occurs informally within routine PR and Issue discussions, without a designated mentor or formal channel [14]. Unlike explicit mentoring, NEM lacks formal commitments and arises naturally alongside technical responses, often without the mentee's awareness. Although NEM is present in publicly accessible discussions, existing recognition mechanisms fail to capture it effectively [4, 44], as its identification requires fine-grained semantic reasoning beyond the capacity of rule-based or coarse-grained approaches. Consequently, valuable mentoring behaviors go unrecognized, potentially reducing mentor motivation to sustain such support.

2.3 LLMs for software engineering tasks

LLMs have become transformative in software engineering, excelling in code completion, repair, and review [46], and demonstrating strong potential for open source community analysis through their contextual understanding and complex reasoning capabilities [37]. However, identifying NEM in PR and Issue discussions requires deep reasoning about both technical accuracy and mentoring intent, and general-purpose models lack the OSS-specific terminology and community norms necessary for reliable direct application. Domain adaptation addresses this through four complementary strategies. CPT [50] internalizes domain language patterns from OSS-specific corpora. SFT [12] calibrates models for specific tasks using labeled data. Preference optimization methods such as PPO [38], DPO [34] and ORPO [20] improve the reliability of subjective tasks by comparing pairwise. Parameter-Efficient Fine-Tuning (PEFT) [11] reduces computational overhead by updating only a small parameter subset, enabling large-scale experimentation in resource-constrained settings.

3 Approach

3.1 Overview

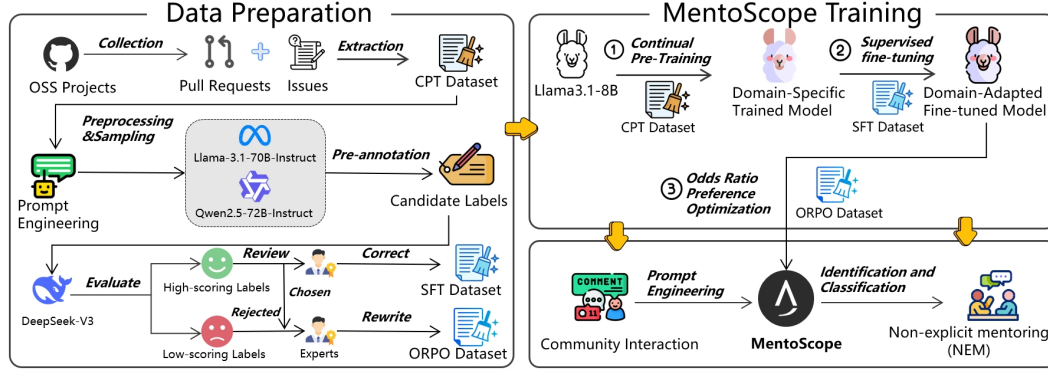
This section presents MentoScope, an LLM-based framework for automated identification and fine-grained classification of NEM in GitHub PR and Issue discussions. As illustrated in Figure 1, MentoScope comprises four sequential stages: data preparation, CPT, SFT, and ORPO. Each stage is described in detail in the following subsections.

3.2 Data preparation

3.2.1 Classification scheme for NEM

To identify NEM in GitHub communities, this study constructs a classification framework through a two-phase approach combining theoretical deduction and empirical induction.

In the theoretical deduction phase, we draw on existing literature on OSS mentoring, including Feng et al.'s definition of implicit mentoring [14], task-oriented mentoring strategies [41], and the CHAOSS measurement framework on mentoring mechanisms and



■ **Figure 1** Overview of the MentoScope framework, including data preparation, multi-stage training (CPT, SFT, and ORPO), and downstream NEM identification and analysis.

contributor diversity. Based on this foundation, NEM is decomposed into three dimensions: technical skill development, project knowledge transfer, and community culture assimilation, yielding 10 initial categories.

In the empirical induction phase, 1,200 PR and Issue comments with mentoring characteristics are randomly sampled from 10 high-activity OSS projects. Four researchers with software engineering backgrounds, including the authors, independently classify the data and iteratively refine the categories through constant comparison and negotiation. Specifically, “providing solutions”, “debugging help”, and “problem clarification” are merged into “Problem Solving Guidance” as all three serve the same function of guiding autonomous issue resolution; “environment configuration guidance” is expanded into “Workflow & Tooling Mastery” to reflect its broader coverage of project-specific toolchains and collaborative workflows. The resulting framework comprises 8 mutually exclusive categories, as detailed in Table 1.

3.2.2 Data collection and preprocessing

To support NEM analysis, we collect OSS project data from GitHub using a data collection strategy guided by three principles: representativeness, activity, and diversity. Repositories are filtered via the GitHub REST API and GitHub Archive metadata according to the following criteria: (1) more than 1,000 stars; (2) over 500 total PRs and Issues; (3) at least 50 commits within the past year, excluding stagnant projects. Major programming languages including C/C++, Java, and Python are covered to ensure linguistic diversity. A total of 500 OSS projects are selected. All PR and Issue discussions are crawled, capturing URLs, creation dates, authors, titles, descriptions, status changes, and commenter details. Automated comments are filtered using regular expressions and a bot list (e.g., github-actions[bot], dependabot), and short non-substantive replies and redundant emojis are removed. The resulting dataset comprises 591,154 high-quality comment records spanning 92,778 PRs, 118,569 Issues, and 101,835 users.

3.2.3 Dataset construction

To support the three-stage training pipeline of MentoScope, three distinct datasets are constructed.

■ **Table 1** Fine-grained classification scheme for NEM.

Category	Description	Examples
A. Code Quality & Best Practices	Guidance on code implementation targeting maintainability, performance, and security principles.	Recommending design patterns; explaining security risks; teaching unit testing.
B. Problem Solving Guidance	Strategic guidance helping contributors resolve issues independently without providing direct solutions.	Suggesting debugging strategies; explaining log anomalies; guiding race condition analysis.
C. Technical Concept Explanation	Explaining foundational or project-specific concepts and terminology to bridge knowledge gaps.	Explaining asynchronous I/O; clarifying cryptographic algorithm principles.
D. Project Rationale & History	Explaining project-specific design philosophy, architectural evolution, and past decision contexts.	Explaining library exclusion decisions; clarifying historical coupling reasons; linking to past architectural discussions.
E. Workflow & Tooling Mastery	Guiding contributors on project-specific workflows, tooling, environment configuration, and CI/CD pipelines.	Assisting with environment setup; instructing on rebase over merge for clean commit history.
F. Community Norms & Process Onboarding	Guiding newcomers through contribution guidelines, communication etiquette, and governance rules to facilitate community integration.	Reminding commit message formatting; correcting inappropriate communication; guiding proper attribution.
G. Collaborative Mediation	Facilitating resolution of technical disagreements through compromise solutions or decision-making frameworks.	Providing trade-off analysis; guiding toward technical consensus; resolving discussion deadlocks.
H. Strategic Motivation & Recognition	Strengthening community belonging through positive feedback, contribution affirmation, or career development advice.	Affirming a newcomer's first PR value; encouraging broader maintenance responsibilities.

► **CPT dataset construction.** The CPT dataset is designed to inject OSS specific language patterns and interaction dynamics through large-scale unsupervised pretraining. For each of the 92,778 PRs and 118,569 Issues, a context reconstruction strategy is applied in which the title and description are concatenated to establish topical context, followed by chronological linearization of associated comments. This structure preserves the narrative and temporal order of technical discussions, enabling the model to learn long-range contextual dependencies. The resulting CPT dataset comprises 211,347 samples.

► **SFT dataset construction.** The SFT dataset equips the model with the ability to identify and classify NEM behaviors through a three-stage annotation pipeline consisting of LLM pre-annotation, automatic filtering, and expert review. Zero-shot prompting is applied to Qwen2.5-72B-Instruct [33] and Llama-3.1-70B-Instruct [30] to perform semantic analysis on each comment. DeepSeek-V3 [24] then evaluates annotation quality across three dimensions: recognition accuracy (weight 0.4), classification accuracy (weight 0.4), and analytical completeness (weight 0.2). A weighted quality score is computed as $\text{Score} = 0.4 \times \text{Recognition Accuracy} + 0.4 \times \text{Classification Accuracy} + 0.2 \times \text{Analytical Completeness}$, which is used to select candidates for expert review. Three technical experts review the annotations, achieving a Fleiss' Kappa of 0.86, and disagreements are resolved through majority voting. The final SFT dataset contains 15,214 high-quality annotated samples.

► **ORPO dataset construction.** A total of 3,280 samples are randomly drawn from the remaining comment pool with no overlap with the SFT dataset. Preference pairs are constructed by reusing the annotation and scoring mechanisms from the SFT stage, where the highest-scoring annotation serves as the chosen response and a lower-scoring annotation exhibiting recognition bias or analytical omissions serves as the rejected response. Samples with insufficient quality distinctions are manually reconstructed to ensure each pair clearly contrasts correct reasoning with typical error patterns. For instance, a problem clarification comment mislabeled as “Workflow & Tooling Mastery” is relabeled as “Technical Concept Explanation”, a more plausible confusable category, to maximize training discriminability.

3.3 CPT: OSS domain adaptation

CPT [50] adapts the general-purpose LLM to the OSS domain by training on large-scale unlabeled GitHub PR and Issue discussions, including titles, descriptions, and comments. This stage is essential because GitHub interactions contain abundant domain-specific terminology, code snippets, implicit context, and distinctive technical communication styles that cannot be adequately captured through fine-tuning alone. Llama-3.1-8B is selected as the base model based on its open-source availability, moderate parameter scale, and strong fine-tuning capability. In preliminary experiments on 100 benchmark samples, Llama-3.1-8B is compared against models of comparable scale, including Qwen2.5-7B [6], and Gemma-2-9B [18], demonstrating superior stability and overall performance.

To ensure computational efficiency during CPT while preserving parameter integrity, Low-Rank Adaptation (LoRA) [21] is employed. LoRA injects trainable low-rank matrices into specific layers of the pre-trained model, representing weight updates as a low-rank decomposition $\Delta W = BA$, where $B \in \mathbb{R}^{d \times r}$, $A \in \mathbb{R}^{r \times k}$, and $r \ll \min(d, k)$, rather than modifying the original weight matrix $W_0 \in \mathbb{R}^{d \times k}$ directly. During CPT, the base model (Llama-3.1-8B) parameters remain frozen and only the LoRA adapters are optimized.

The objective of continual pre-training is to enhance the model’s capability to predict the next token conditioned on preceding context, which corresponds to a standard language modeling task. During training, we employ the cross-entropy loss to quantify the discrepancy between the predicted token distribution and the ground-truth next-token distribution.

For a sample in the CPT dataset corresponding to a complete discussion sequence of a PR or Issue, let its length be T_i and denote the token sequence as $\mathbf{w}_i = (w_{i,1}, w_{i,2}, \dots, w_{i,T_i})$. The overall continual pre-training loss is defined as:

$$L_{\text{CPT}} = -\frac{1}{|D_{\text{CPT}}|} \sum_{i=1}^{|D_{\text{CPT}}|} \sum_{t=1}^{T_i} \log P(w_{i,t} \mid w_{i,<t}; \theta, \Phi) \quad (1)$$

where $|D_{\text{CPT}}|$ denotes the total number of CPT samples and T_i the length of the i -th sample. The token $w_{i,t}$ is the target token at position t , and $w_{i,<t}$ represents all preceding tokens serving as context. The term $P(w_{i,t} \mid w_{i,<t}; \theta, \Phi)$ is the model’s predicted probability of generating $w_{i,t}$ given its context, parameterized by the frozen base model parameters θ and the trainable LoRA parameters Φ comprising matrices A and B . Gradients are computed via backpropagation and only Φ is updated to minimize L_{CPT} , while θ remains unchanged to preserve the linguistic competence of the pre-trained model.

Through this process, the Llama-3.1-8B model acquires OSS-specific technical terminology, abbreviations, code references, and implicit communication patterns from developer discussions, achieving domain adaptation within its latent representation space. After training, the learned LoRA weights are merged back into the base model to obtain a domain-adapted backbone, defined as $W_{\text{merged}} = W_0 + \alpha_{\text{LoRA}} \cdot BA$, where α_{LoRA} denotes the scaling factor controlling the contribution of the LoRA adaptation weights.

3.4 SFT: NEM identification and classification

SFT [12] further optimizes the domain-adapted base model to accurately identify and classify NEM behaviors in PR and Issue comments. To this end, we design prompts that guide the model to transform unstructured comments into structured outputs aligned with our research requirements. Following established prompt design principles of providing necessary context, defining clear task instructions, and specifying output format [25, 27], we construct the following prompt:

System Context

You are an expert software engineer and senior OSS maintainer specializing in analyzing developer interactions to identify Non-explicit Mentoring (NEM). NEM refers to spontaneous guidance embedded in technical discussions, through which experienced contributors support others' technical growth or community integration.

Task Description

Analyze the given GitHub comment and determine whether it contains NEM. If present, classify the comment into one or more NEM categories based on the classification scheme. Multiple categories may apply.

Input Data

Interaction Type: {{Interaction Type}}
 Title: {{Title}}
 Discussion Summary: {{Summary}}
 Target Comment Content: {{Comment}}

Mentoring classification scheme

Classify the comment into one or more of the following categories:
 {{Scheme Definition}}

Output Format

Provide the result in strict JSON format:

```
{
  "contains_nem": true / false,
  "categories": [
    {
      "category_name": "Category name from classification scheme",
      "evidence": "Brief justification based on comment content."
    }
  ]
}
```

In the prompt template, {{Interaction Type}} specifies whether the comment originates from a PR or an Issue. {{Title}} represents the title of the PR or Issue, and {{Summary}} provides a concise overview of its main description; full descriptions are summarized in advance to avoid exceeding context window limits and to reduce noise from lengthy code snippets or environmental details. {{Comment}} denotes the specific text under analysis, serving as the direct input for NEM identification and classification.

Each annotated sample in the SFT dataset is converted into a (Prompt, Response) pair for supervised learning. The training objective is to minimize the discrepancy between model predictions and human annotations. As the SFT dataset is smaller than the CPT dataset and the task requires precise instruction-following and structured output, full-parameter fine-tuning is adopted. For each sample (x, y) , where x is the input prompt and y is the human-annotated target response, the loss function is defined as:

$$\mathcal{L}_{\text{SFT}} = -\frac{1}{|\mathcal{D}_{\text{SFT}}|} \sum_{(x,y) \in \mathcal{D}_{\text{SFT}}} \sum_{t=1}^{|y|} \log P(y_t | x, y_{<t}; \theta) \quad (2)$$

where $|\mathcal{D}_{\text{SFT}}|$ is the total number of SFT samples, $|y|$ the length of the target sequence, y_t the t -th output token, and $y_{<t}$ the preceding $t - 1$ tokens. The term $P(y_t | x, y_{<t}; \theta)$ denotes the model’s predicted probability of generating y_t given the input prompt x and the preceding tokens $y_{<t}$, with θ representing the full set of model parameters initialized from the CPT-adapted backbone with merged LoRA weights. The model minimizes $\mathcal{L}_{\text{SFT}}$ via backpropagation to update θ , learning to perform NEM identification and classification based on prompt instructions and contextual input, thereby generating structured outputs that follow the prescribed format.

3.5 ORPO: Preference alignment for error mitigation

Although SFT maximizes the probability of generating correct responses, it does not differentiate between outputs of varying quality, leaving the model prone to plausible but erroneous analyses in ambiguous contexts. To address this, we incorporate ORPO [20], a preference alignment method that integrates supervised fine-tuning and preference alignment into a single optimization objective, eliminating the need for a reference model while increasing the generation probability of chosen responses and suppressing that of rejected ones.

The ORPO training dataset consists of triplets (x, y_c, y_r) , where x is the input prompt following the SFT format, y_c is a human-verified high-quality NEM analysis, and y_r is a rejected response containing recognition biases or logical errors. ORPO reinforces preferred outputs by optimizing the log odds ratio between y_c and y_r . For an input x , the odds of generating y are defined as $\text{odds}_\theta(y | x) = P_\theta(y | x) / (1 - P_\theta(y | x))$, where $P_\theta(y | x)$ denotes the probability of generating the complete sequence y . The corresponding odds ratio is $\text{OR}_\theta(y_c, y_r | x) = \text{odds}_\theta(y_c | x) / \text{odds}_\theta(y_r | x)$, measuring the model’s relative preference for the chosen response over the rejected response.

After introducing the concept of the odds ratio, the loss function of ORPO is defined as:

$$\mathcal{L}_{\text{ORPO}} = -\frac{1}{|\mathcal{D}_{\text{ORPO}}|} \sum_{(x, y_c, y_r) \in \mathcal{D}_{\text{ORPO}}} (\log P_\theta(y_c | x) + \lambda \log \sigma(\log \text{OR}_\theta(y_c, y_r | x))) \quad (3)$$

Here, θ denotes the trainable model parameters initialized from the SFT-adapted model. $|\mathcal{D}_{\text{ORPO}}|$ is the total number of ORPO samples. The first term in the loss function is a negative log-likelihood term that encourages the model to increase the generation probability of the chosen response y_c . The second term is a preference term that maximizes the log odds ratio of y_c relative to the rejected response y_r , where σ denotes the sigmoid function. The hyperparameter λ controls the relative weight of the two terms, with larger values making the preference term more dominant. By minimizing $\mathcal{L}_{\text{ORPO}}$ in Equation (6), the model simultaneously increases the generation probability of the chosen response y_c and enlarges its probability margin over the rejected response y_r . This joint optimization enables the model to produce accurate NEM analyses while suppressing known error patterns, thereby improving the robustness and overall accuracy of MentoScope.

The resulting model after the ORPO stage constitutes the final MentoScope model, which integrates the general capabilities of the base model, the domain knowledge from CPT, the task execution ability from SFT, and the preference alignment with human experts through ORPO, enabling accurate and reliable identification and classification of NEM in open source community interactions.

4 Experimental setup

4.1 Research questions

This section presents the evaluation of MentoScope on GitHub data, addressing the following research questions:

- (1) RQ1: How effectively does MentoScope identify and classify NEM?
- (2) RQ2: How do different components of MentoScope contribute to overall performance?
- (3) RQ3: How is NEM distributed across different contextual settings?
- (4) RQ4: How does NEM influence newcomer retention and performance?

4.2 Datasets

► **CPT dataset.** This dataset adapts general-purpose LLMs to the OSS domain, constructed from 500 OSS projects collected in Section 3.2.3, encompassing 92,778 PRs and 118,569 Issues. Through contextual reconstruction and linear integration of titles, descriptions, and associated comments, a pre-training dataset of 211,347 samples is produced.

► **SFT dataset.** To support accurate NEM identification and classification, 15,214 high-quality annotated samples are produced from 591,154 comments across 500 OSS projects via a three-stage pipeline of LLM pre-annotation, automated filtering, and expert verification. The dataset is split into training (10,650), validation (1,521), and test (3,043) datasets at a 7:1:2 ratio. The test dataset also serves as the unified benchmark for evaluating the overall MentoScope framework, ensuring consistent comparison across all experimental settings.

► **ORPO dataset.** To further enhance discriminative capability through preference alignment, 3,280 samples are randomly drawn from the remaining comment pool with no overlap with the SFT dataset, and preference pairs comprising chosen and rejected responses are constructed. Each pair is manually verified and refined to ensure a clear and justified quality gap between positive and negative samples. As preference optimization typically adopts a low-epoch training strategy to mitigate catastrophic forgetting, all samples are used for training.

4.3 Baselines

Prior work primarily relies on questionnaire-based qualitative analysis. Feng et al. [14] apply traditional machine learning models including random forest to coarsely distinguish implicit guidance from routine PR interactions, without differentiating specific guidance types. Cobos et al. [7] integrate LLMs with prompt engineering to classify Issue comments by sentiment polarity, focusing on ethical dimensions such as constructive and abusive language. Building on these approaches, we evaluate MentoScope against three categories of baselines.

► **ML-based methods.** Naive Bayes [28], K-Nearest Neighbors [9], and XGBoost [5] serve as lexical baselines, all using TF-IDF [39] features without contextual semantic encoding, enabling assessment of whether lexical-statistical signals alone suffice for NEM identification.

► **PTM-based methods.** BERT [10], RoBERTa [26], and T5 [35] are employed to examine the contribution of contextual semantic representation. BERT and RoBERTa are fine-tuned with a linear classification head, while T5 is cast as a sequence-to-label generation task. All models are fine-tuned on the SFT dataset and evaluated on the same test set.

► **LLM-based methods.** Open source LLMs of comparable parameter scale are included as baselines, namely Qwen2.5-7B [6], Llama-3.1-8B [19], Mistral-7B-v0.3 [32], and Gemma-2-9B [17], each evaluated under zero-shot and few-shot settings. The zero-shot setting directly applies the prompt in Section 3.4, while the few-shot setting incorporates labeled examples to enable in-context learning of NEM definitions.

4.4 Metrics

NEM identification is characterized as a binary classification task that determines whether a PR or Issue comment contains any form of NEM. We evaluate this task using Accuracy, Precision, Recall, and F1-score, defined as $Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$, $Precision = \frac{TP}{TP+FP}$, $Recall = \frac{TP}{TP+FN}$, and $F1-Score = \frac{2 \cdot Precision \cdot Recall}{Precision+Recall}$. Here, TP and TN denote correctly predicted positive and negative samples, whereas FP and FN denote false positives and false negatives.

NEM classification is characterized as a multi-label classification task, where a single comment may involve one or more NEM categories. We consider eight categories, listed in Table 1, and evaluate classification performance using Jaccard Accuracy together with micro-averaged Precision, Recall, and F1-score. These metrics are computed only on comments for which NEM is correctly identified, thereby measuring categorization performance conditioned on successful detection.

For each sample i , the Jaccard Score is defined as $Jaccard-Score_i = \frac{|Y_{true,i} \cap Y_{pred,i}|}{|Y_{true,i} \cup Y_{pred,i}|}$, where $Y_{true,i}$ and $Y_{pred,i}$ denote the ground-truth and predicted label sets. Jaccard Accuracy is defined as the mean Jaccard Score over all N samples, that is, $Jaccard-Accuracy = \frac{1}{N} \sum_{i=1}^N Jaccard-Score_i$.

To assess overall performance in the multi-label setting, we further report micro-averaged Precision, Recall, and F1-score, which aggregate predictions across all $C = 8$ categories and are suitable for imbalanced data. Specifically, $Micro-Precision = \frac{\sum_{c=1}^C TP_c}{\sum_{c=1}^C (TP_c + FP_c)}$, $Micro-Recall = \frac{\sum_{c=1}^C TP_c}{\sum_{c=1}^C (TP_c + FN_c)}$, and $Micro-F1-Score = \frac{2 \cdot (Micro-Precision \cdot Micro-Recall)}{Micro-Precision + Micro-Recall}$, where TP_c , FP_c , and FN_c denote the numbers of true positives, false positives, and false negatives for category c , respectively.

4.5 Implementation details

The MentoScope training pipeline is implemented on the ms-swift framework [48], an official fine-tuning and deployment framework provided by the ModelScope community, and proceeds through the CPT, SFT, and ORPO stages. In the CPT stage, LoRA-based parameter-efficient fine-tuning is applied with rank 64, scaling factor 128, and all linear layers as target modules. The maximum sequence length is set to 8,192, the effective batch size is set to 32 through 4-step gradient accumulation, the learning rate is set to 1×10^{-5} , and the model is trained for 1 epoch. Both SFT and ORPO adopt full-parameter fine-tuning with a maximum sequence length of 4,096, a learning rate of 1×10^{-4} , and the same effective batch size. SFT is trained for 3 epochs to fit the annotated label distribution, whereas ORPO is trained for 1 epoch to reinforce preference for high-quality responses. All experiments are conducted on 8 NVIDIA H100 (80GB) GPUs. During inference, greedy decoding is applied with $temperature = 0$ and $do_sample = False$ to ensure reproducible results.

5 Evaluation and discussion

5.1 Effectiveness of NEM identification and classification

Table 2 presents the performance comparison of MentoScope and baseline methods on NEM identification and classification. MentoScope achieves F1-scores of 94.20% and 80.13% on identification and classification respectively, outperforming the best general LLM baseline (Llama-3.1-8B 5-shot) by 6.38% and 3.60%. ML-based methods fall substantially behind.

■ **Table 2** Performance comparison between MentoScope and baselines on NEM identification and classification tasks.

Model	NEM Identification (%)				NEM Classification (%)			
	ACC	P	R	F1	ACC	P	R	F1
<i>ML-based methods</i>								
KNN	62.34	87.85	62.16	72.81	48.40	50.63	47.66	49.10
Naive Bayes	68.72	85.63	71.81	78.11	46.56	52.63	48.23	50.33
XGBoost	71.63	81.76	76.98	79.30	50.97	61.65	50.46	55.50
<i>PTM-based methods</i>								
BERT	77.38	81.53	88.94	85.08	66.16	73.58	67.25	70.27
RoBERTa	78.64	81.67	90.29	85.76	68.99	78.41	68.74	73.26
T5	79.03	81.89	90.54	86.00	69.37	79.18	68.51	73.46
<i>LLM-based methods (0-shot)</i>								
Mistral-7B-v0.3	64.15	87.24	64.30	74.03	51.88	54.26	52.44	53.33
Qwen2.5-7B	65.32	86.56	65.16	74.35	53.09	53.62	57.92	55.69
Llama-3.1-8B	65.81	88.18	65.52	75.18	58.96	62.23	58.30	60.20
Gemma-2-9B	65.50	88.61	64.79	74.85	57.45	60.55	56.73	58.58
<i>LLM-based methods (5-shot)</i>								
Mistral-7B-v0.3	77.90	87.58	84.89	86.21	68.24	73.35	71.43	72.38
Qwen2.5-7B	82.17	90.56	83.52	86.90	70.48	76.25	73.20	74.70
Llama-3.1-8B	83.98	91.09	84.78	87.82	72.35	78.71	74.46	76.53
Gemma-2-9B	83.72	89.98	84.47	87.14	71.84	78.57	73.94	76.18
MentoScope	88.30	95.74	92.70	94.20	75.73	82.28	78.09	80.13

KNN achieves high precision (87.85%) but extremely low recall (62.16%) on identification, while XGBoost, the strongest ML baseline, reaches only 55.50% classification F1-score, highlighting the inability of shallow TF-IDF features to capture the implicit semantics of NEM behaviors. PTM-based models such as BERT perform adequately on identification ($F1 > 85\%$) but deteriorate on fine-grained classification requiring deep semantic understanding. T5, the best PTM baseline, trails MentoScope by 6.67% in classification F1-score, indicating that standard pretrained models lack the OSS domain specificity needed to distinguish fine-grained NEM categories.

General LLMs struggle in zero-shot settings, typically scoring below 75% and 61% in identification and classification F1-scores, respectively. This reflects their difficulty in grasping the precise academic boundaries of NEM. Providing 5-shot examples significantly enhances LLM performance with an average F1-score increase of 12%, proving that in-context examples help align model comprehension with mentoring behaviors. However, a distinct gap persists compared to the fully optimized MentoScope.

Ultimately, MentoScope achieves exceptionally high precision (95.74%) while maintaining high recall (92.70%). In practical NEM identification, this high precision crucially minimizes false positives, preventing standard bug-fix discussions from being misjudged as mentoring. This accuracy is vital for subsequently quantifying developer contributions and establishing fair community recognition mechanisms.

Answer to RQ1: MentoScope consistently outperforms all baselines, achieving F1-scores of 94.20% for NEM identification and 80.13% for NEM classification, which confirms the effectiveness of the multi-stage training pipeline for identifying and categorizing NEM in open source community interactions.

■ **Table 3** Performance on NEM identification and NEM classification.

Model	NEM Identification (%)				NEM Classification (%)			
	ACC	P	R	F1	ACC	P	R	F1
w/o CPT	86.70	93.50	90.61	92.03	72.11	80.31	74.32	77.20
w/o SFT	78.30	89.71	81.08	85.18	65.03	70.91	68.13	69.49
w/o ORPO	83.60	91.56	87.96	89.72	70.13	76.69	72.41	74.49
w/o Title	86.50	93.42	90.96	92.17	73.39	80.23	74.92	78.02
w/o Summary	80.90	91.29	85.59	88.35	71.16	77.39	75.47	76.42
MentoScope	88.30	95.74	92.70	94.20	75.73	82.28	78.09	80.13

5.2 Impact of component design on performance

To evaluate the contribution of MentoScope’s core components and input representations, we design several ablation variants for comparative analysis. The experimental factors include the multi-stage training strategy (CPT, SFT, ORPO) and prompt-level contextual enhancements (Title, Summary). The variants under evaluation are as follows:

- **w/o CPT**: Removes continual pretraining to assess the effect of domain adaptation.
- **w/o SFT**: Omits supervised fine-tuning to evaluate the necessity of task-specific instruction alignment.
- **w/o ORPO**: Disables preference optimization to measure the effect of contrastive preference learning on resolving ambiguous decision boundaries.
- **w/o Title** and **w/o Summary**: Individually removes the PR/Issue title and textual summary from the prompt to examine the contribution of global contextual information to local comment interpretation.

Table 3 shows that the three training stages contribute unevenly to overall performance. Removing SFT causes the most substantial degradation, with identification and classification F1-scores dropping by 9.02% and 10.64% to 85.18% and 69.49% respectively, confirming that SFT serves as the primary mechanism for acquiring task-specific discriminative capability. Removing ORPO reduces identification and classification F1-scores by 4.48% and 5.64% to 89.72% and 74.49%, indicating that preference alignment improves robustness on semantically ambiguous inputs. Excluding CPT results in comparatively moderate reductions of 2.17% and 2.93%, demonstrating that domain-adaptive pretraining provides valuable contextual priors that benefit subsequent training stages.

The ablation results on prompt components highlight the importance of contextual input design. Removing the PR/Issue summary causes the most substantial degradation, with F1-scores dropping by 5.85% on identification and 3.71% on classification, as raw descriptions introduce noise and increase inference overhead while diluting task-relevant semantic cues. A structured summary thus serves as an effective information compression mechanism that supplies high-density signals for NEM detection. Removing the title has a more limited effect, with reductions of 2.03% and 2.11% on identification and classification respectively, indicating that titles provide a concise topic indicator whose marginal contribution diminishes when a high-quality summary is present.

Answer to RQ2: Ablation results indicate that each component of MentoScope contributes positively to overall performance. Through a multi-stage training process and an optimized prompt design, the framework integrates domain adaptation, task-specific learning, and preference optimization, leading to strong performance in both NEM identification and NEM classification tasks.

■ **Table 4** Distribution of NEM categories.

NEM Category	Proportion (%)
Code Quality & Best Practices	14.28
Problem Solving Guidance	23.45
Technical Concept Explanation	3.12
Project Rationale & History	7.92
Workflow & Tooling Mastery	5.67
Community Norms & Process Onboarding	4.15
Collaborative Mediation	1.84
Strategic Motivation & Recognition	12.36
Regular Discussion (Non-NEM)	27.21

■ **Table 5** Distribution of NEM categories across project sizes (%).

NEM Category	Small	Medium	Large
Code Quality & Best Practices	11.24	14.56	17.04
Problem Solving Guidance	21.15	23.12	26.08
Technical Concept Explanation	2.54	3.10	3.72
Project Rationale & History	5.12	8.24	10.40
Workflow & Tooling Mastery	6.84	5.52	4.65
Community Norms & Process Onboarding	5.42	4.02	3.01
Collaborative Mediation	1.10	1.95	2.47
Strategic Motivation & Recognition	16.51	12.18	8.39
Regular Discussion (Non-NEM)	30.08	27.31	24.24

5.3 Distribution patterns of NEM across contexts

After establishing the effectiveness of MentoScope, we apply it to 591,154 comments from 500 GitHub projects collected in Section 3.2.2 to examine NEM distribution in real-world settings. For clearer comparative analysis, regular discussion (Non-NEM) is included as a reference category.

Table 4 presents the distribution of eight NEM behavior types in open source community interactions. NEM accounts for 72.79% of GitHub collaborations, substantially exceeding routine transactional discussions (27.21%). “Problem Solving Guidance” (23.45%) and “Code Quality & Best Practices” (14.28%) together comprise nearly 40% of NEM instances, indicating that contributors predominantly guide debugging and convey engineering practices rather than providing direct solutions, facilitating the transition from novice to experienced contributors. “Strategic Motivation & Recognition” (12.36%) underscores the role of positive reinforcement in sustaining community cohesion. “Project Rationale & History” (7.92%) reflects the long-tail nature of OSS discussions, where higher-level mentoring on architectural evolution remains rare. “Collaborative Mediation”, though the smallest category (1.84%), is critical for maintaining decision-making consistency. The balanced distribution of “Workflow & Tooling Mastery” and “Community Norms & Process Onboarding” reflects sustained community efforts to onboard new members.

To investigate NEM distribution across OSS projects of varying scales, 500 OSS projects are classified into three categories via multidimensional weighted clustering based on core contributor count, total lines of code (LOC), and PR count. Small-scale projects have fewer than 5 core maintainers and typically comprise emerging tools or libraries with narrowly scoped functionality. Medium-scale projects have 5 to 20 core maintainers, with established collaboration workflows and a stable user base. Large-scale projects have more than 20 core maintainers and represent ecosystem-level projects with high sociotechnical complexity.

Table 5 presents the distribution of NEM categories across projects of varying scales. “Problem Solving Guidance”, “Code Quality & Best Practices”, and “Project Rationale & History” increase monotonically with project scale, reflecting greater architectural complexity

■ **Table 6** Distribution of NEM categories across PR and Issue comments.

NEM Category	PR (%)	Issue (%)
Code Quality & Best Practices	24.15	4.41
Problem Solving Guidance	15.30	31.60
Technical Concept Explanation	2.05	4.19
Project Rationale & History	7.12	8.72
Workflow & Tooling Mastery	6.24	5.10
Community Norms & Process Onboarding	4.88	3.42
Collaborative Mediation	1.35	2.33
Strategic Motivation & Recognition	16.24	8.48
Regular Discussion (Non-NEM)	22.67	31.75

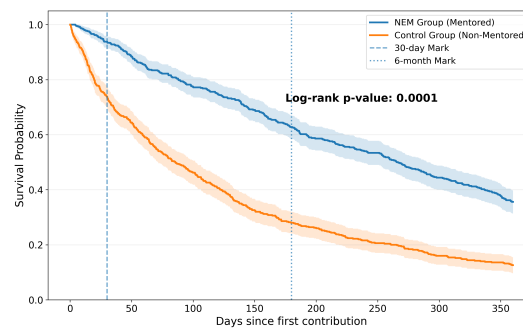
and cognitive load in larger projects, where “Regular Discussion(Non-NEM)” reaches its lowest proportion (24.24%), indicating higher mentoring density. Small-scale projects exhibit affective and normative guidance patterns, with “Strategic Motivation & Recognition” and “Community Norms & Process Onboarding” occurring at substantially higher rates, revealing retention strategies aimed at alleviating contributor scarcity and lowering participation barriers. This structural shift from institutionalized technical norms in large-scale projects to interpersonal motivational mechanisms in small-scale ones demonstrates that NEM adaptively adjusts its knowledge transfer and member cultivation functions according to community scale and organizational structure.

Table 6 presents the distribution of NEM categories across PR and Issue discussions. In PR discussions, “Code Quality & Best Practices” predominates (24.15%), consistent with code review as a quality control mechanism, while “Strategic Motivation & Recognition” occurs at a substantially higher rate (16.24%), reflecting maintainers’ tendency to reinforce contributors’ sense of achievement upon code submission. In Issue discussions, “Problem Solving Guidance” is most prevalent (31.60%), with elevated frequencies of “Technical Concept Explanation” and “Collaborative Mediation” indicating that conceptual clarification and resolution of technical disagreements are primarily addressed at this stage. These patterns demonstrate the functional complementarity of NEM across collaboration contexts, with Issue discussions supporting problem exploration and PR discussions emphasizing norm reinforcement and contribution recognition.

Answer to RQ3: Empirical analysis reveals that NEM accounts for over 72% of community interactions, with distribution varying by project scale and collaboration context. Large-scale projects emphasize technical norm transmission while small-scale projects exhibit stronger affective and normative guidance; Issue discussions prioritize conceptual clarification while PR discussions focus on quality reinforcement, demonstrating that NEM constitutes a context-adaptive form of implicit knowledge transfer in OSS communities.

5.4 RQ4: Impact of NEM on newcomer retention and performance

Based on the observed prevalence and contextual distribution of NEM, we further assess its impact on newcomer retention and performance. We identify 1,245 newcomer contributors from the 500 projects, defined as developers submitting their first PR within a project, and divide them into two groups: (1) the NEM group, consisting of newcomers who receive at least one instance of NEM during their initial PR or Issue interactions, and (2) the control group, comprising those who engage only in routine transactional discussions. We track their activity over a 12-month period and compare outcomes in terms of community retention and technical performance using survival analysis and non-parametric statistical tests.



■ **Figure 2** Kaplan–Meier survival curves: newcomer retention.

To assess the longitudinal effect of NEM on newcomer retention, survival analysis is employed, defining contributor disengagement as the absence of activity for more than 6 months following the first PR submission. As shown in Figure 2, Kaplan-Meier survival curves reveal significant retention differences between the two groups. Within the first 30 days, the NEM group maintains a survival rate above 85%, while approximately 30% of control group members cease all interactions within one month. A log-rank test confirms a statistically significant difference between the two curves ($p < 0.001$). At the sixth month, newcomers receiving NEM exhibit an average active tenure approximately 35% longer than the control group, with this advantage remaining robust at the twelfth month.

PR merge rate is used as the primary metric for newcomers’ short-term technical performance. The Wilcoxon rank-sum test [36] shows that the NEM group achieves a significantly higher average merge rate than the control group (74.82% vs. 62.16%, $p < 0.001$), with a meaningful effect size. This result suggests that NEM provides substantive knowledge support rather than simple social interaction. Guidance such as “Problem-Solving Guidance” and “Code Quality & Best Practices” helps newcomers quickly align with project conventions, reducing PR rejections caused by technical misunderstanding or limited norm awareness.

Answer to RQ4: Empirical results show that NEM has a significant positive effect on newcomer contributors. Newcomers who receive NEM achieve an approximately 12.6% higher PR merge rate, along with significantly stronger long-term retention and longer community activity cycles than the control group. These findings confirm the value of NEM as an effective knowledge transfer mechanism.

6 Threats to Validity

Internal validity. The definition of NEM involves inherent sociological subjectivity, and perceptions of mentoring may vary across cultural contexts. Annotation subjectivity and inconsistency may affect model training outcomes, which we mitigate through a three-stage pipeline combining LLM pre-annotation, automated filtering, and expert review.

External validity. This study is conducted on GitHub OSS projects, and the findings are therefore primarily applicable to GitHub collaboration contexts. Generalizability to other platforms or domains with distinct development constraints warrants further investigation. Additionally, although the survival analysis in RQ4 controls for multiple observable covariates, newcomers’ sustained activity may still be subject to project-external factors.

7 Related work

7.1 Analyzing mentoring behaviors in OSS

Mentorship has long been recognized as a critical factor in open source community sustainability [15]. Early studies on formal mentorship examined role models, guidance strategies, and collaboration dynamics [16], finding that while formal mentorship improves contributor retention, its effectiveness is moderated by project characteristics and social support conditions [13]. Recent attention has shifted toward the more pervasive NEM behaviors. Steinmacher et al. [42] reveal through qualitative analysis the hidden efforts and challenges faced by mentors across technical, social, and motivational dimensions. Feng et al. [14] define NEM as informal mentoring embedded in PR comments and demonstrate its prevalence and positive community effects. However, existing studies rely predominantly on manual analysis or traditional machine learning models, with clear limitations in automated fine-grained NEM classification; impact assessments largely employ questionnaires or interviews, lacking large-scale empirical grounding. This study proposes MentoScope to address these gaps through automated NEM identification and large-scale distributional analysis.

7.2 LLM-based analysis of OSS developer activities

The advancement of LLMs enables deep semantic extraction from large volumes of developer interactions, overcoming the analytical limitations of traditional text mining. Previous work has applied LLMs to understand the intention of developers [1], assess ideological themes [45], construct behavior taxonomies [43], and analyze gendered collaboration patterns [49]. Distinct from these studies on general developer behaviors, this work focuses on NEM as an underexplored yet practically significant behavior type. MentoScope integrates CPT, SFT, and ORPO to adapt a general-purpose LLM to the software engineering domain, enabling precise identification of fine-grained mentoring behaviors in PR comments and supporting large-scale analysis of implicit knowledge transfer among developers.

8 Conclusion

This paper investigates NEM in open source communities as a fine-grained form of guidance-oriented knowledge transfer embedded in PR and Issue discussions. We propose MentoScope, a three-stage LLM framework integrating CPT, SFT, and ORPO, which achieves state-of-the-art performance on automated NEM identification and classification. Large-scale empirical analysis further reveals the prevalence of NEM, its distributional differences across project contexts, and its positive association with newcomer retention and contribution outcomes.

References

- 1 Toufique Ahmed and Premkumar Devanbu. Few-shot training llms for project-specific code-summarization. In *Proceedings of the 37th IEEE/ACM international conference on automated software engineering*, pages 1–5, 2022. doi:10.1145/3551349.3559555.
- 2 Javier Luis Cánovas Izquierdo and Jordi Cabot. On the analysis of non-coding roles in open source development: an empirical study of npm package projects. *Empirical Software Engineering*, 27(1):18, 2022.
- 3 CHAOSS Working Group. Metric: Mentorship. CHAOSS. [Online]. Available: <https://chaoss.community/kb/metric-mentorship/>, n.d. Accessed: May 16, 2026.

- 4 CHAOSS Working Group. Metric: Types of contributions. CHAOSS. [Online]. Available: <https://chaoss.community/kb/metric-types-of-contributions/>, n.d. Accessed: May 16, 2026.
- 5 Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016. doi:10.1145/2939672.2939785.
- 6 Yunfei Chu, Jin Xu, Qian Yang, Haojie Wei, Xipin Wei, Zhifang Guo, Yichong Leng, Yuanjun Lv, Jinzheng He, Junyang Lin, et al. Qwen2-audio technical report. *arXiv preprint*, 2024. arXiv:2407.10759.
- 7 Sergio Cobos and Javier Luis Cánovas Izquierdo. Ethical classification of non-coding contributions in open-source projects via large language models. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, volume 8(1), pages 643–652, 2025. doi:10.1609/AIES.V8I1.36578.
- 8 Catarina Costa, Jair Figueirêdo, João Felipe Pimentel, Anita Sarma, and Leonardo Murta. Recommending participants for collaborative merge sessions. *IEEE Transactions on Software Engineering*, 47(6):1198–1210, 2019.
- 9 Thomas Cover and Peter Hart. Nearest neighbor pattern classification. *IEEE transactions on information theory*, 13(1):21–27, 1967. doi:10.1109/TIT.1967.1053964.
- 10 Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pages 4171–4186, 2019. doi:10.18653/V1/N19-1423.
- 11 Ning Ding, Yujia Qin, Guang Yang, Fuchao Wei, Zonghan Yang, Yusheng Su, Shengding Hu, Yulin Chen, Chi-Min Chan, Weize Chen, et al. Parameter-efficient fine-tuning of large-scale pre-trained language models. *Nature machine intelligence*, 5(3):220–235, 2023. doi:10.1038/S42256-023-00626-4.
- 12 Guanting Dong, Hongyi Yuan, Keming Lu, Chengpeng Li, Mingfeng Xue, Dayiheng Liu, Wei Wang, Zheng Yuan, Chang Zhou, and Jingren Zhou. How abilities in large language models are affected by supervised fine-tuning data composition. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 177–198, 2024. doi:10.18653/V1/2024.ACL-LONG.12.
- 13 Fabian Fagerholm, Alejandro S Guinea, Jürgen Münch, and Jay Borenstein. The role of mentoring and project characteristics for onboarding in open source software projects. In *Proceedings of the 8th ACM/IEEE international symposium on empirical software engineering and measurement*, pages 1–10, 2014.
- 14 Zixuan Feng, Amreeta Chatterjee, Anita Sarma, and Iftekhhar Ahmed. A case study of implicit mentoring, its prevalence, and impact in apache. In *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 797–809, 2022. doi:10.1145/3540250.3549167.
- 15 Zixuan Feng, Katie Kimura, Bianca Trinkenreich, Anita Sarma, and Igor Steinmacher. Guiding the way: A systematic literature review on mentoring practices in open source software projects. *Information and Software Technology*, 171:107470, 2024. doi:10.1016/J.INFSOF.2024.107470.
- 16 Zixuan Feng, Igor Steinmacher, Marco Gerosa, Tyler Menezes, Alexander Serebrenik, Reed Milewicz, and Anita Sarma. The multifaceted nature of mentoring in oss: strategies, qualities, and ideal outcomes. In *2025 IEEE/ACM 18th International Conference on Cooperative and Human Aspects of Software Engineering (CHASE)*, pages 203–214. IEEE, 2025.
- 17 Gemma Team. Gemma. Kaggle. [Online]. Available: <https://www.kaggle.com/models/google/gemma>, 2024. DOI: 10.34740/KAGGLE/M/3301. Accessed: Mar 16, 2026.
- 18 Google. Gemma 2 model card. Hugging Face. [Online]. Available: <https://huggingface.co/google/gemma-2-9b-it>, 2024. Accessed: May 16, 2026.

- 19 Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint*, 2024. [arXiv:2407.21783](https://arxiv.org/abs/2407.21783).
- 20 Jiwoo Hong, Noah Lee, and James Thorne. Orpo: Monolithic preference optimization without reference model. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 11170–11189, 2024. doi:10.18653/V1/2024.EMNLP-MAIN.626.
- 21 Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yanzhi Li, Shean Wang, Liang Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *Iclr*, 1(2):3, 2022.
- 22 Robin A Lange, Anna Gibson, Milo Z Trujillo, and Brooke Foucault Welles. Invisible labor: The backbone of open source software. *arXiv preprint*, 2025. doi:10.48550/arXiv.2503.13405.
- 23 Linux Foundation. Linux foundation mentorship programs. [Online]. Available: <https://www.linuxfoundation.org/about/mentorship-programs/>, 2025. Accessed: May 16, 2026.
- 24 Aixin Liu, Aoxue Mei, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, et al. Deepseek-v3. 2: Pushing the frontier of open large language models. *arXiv preprint*, 2025. [arXiv:2512.02556](https://arxiv.org/abs/2512.02556).
- 25 Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM computing surveys*, 55(9):1–35, 2023. doi:10.1145/3560815.
- 26 Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint*, 2019. [arXiv:1907.11692](https://arxiv.org/abs/1907.11692).
- 27 Ggaliwango Marvin, Nakayiza Hellen, Daudi Jjingo, and Joyce Nakatumba-Nabende. Prompt engineering in large language models. In *International conference on data intelligence and cognitive informatics*, pages 387–402. Springer, 2023.
- 28 Andrew McCallum. A comparison of event models for naive bayes text classification, 1998.
- 29 John Meluso, Amanda Casari, Katie McLaughlin, and Milo Z Trujillo. Invisible labor in open source software ecosystems. *Proceedings of the ACM on Human-Computer Interaction*, 9(7):1–32, 2025. doi:10.1145/3757417.
- 30 Meta. Llama-3.1-70b-instruct. Hugging Face. [Online]. Available: <https://huggingface.co/meta-llama/Llama-3.1-70B-Instruct>, 2024. Accessed: May 16, 2026.
- 31 Meta. Llama-3.1-8b. Hugging Face. [Online]. Available: <https://huggingface.co/meta-llama/Llama-3.1-8B>, 2024. Accessed: May 16, 2026.
- 32 Mistral AI Team. Mistral-7b-instruct-v0.3. Hugging Face. [Online]. Available: <https://huggingface.co/mistralai/Mistral-7B-Instruct-v0.3>, 2024. Accessed: May 16, 2026.
- 33 Qwen Team. Qwen2.5-72b-instruct. Hugging Face. [Online]. Available: <https://huggingface.co/Qwen/Qwen2.5-72B-Instruct>, 2024. Accessed: May 16, 2026.
- 34 Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems*, 36:53728–53741, 2023.
- 35 Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020. URL: <https://jmlr.org/papers/v21/20-074.html>.
- 36 Bernard Rosner, Robert J Glynn, and Mei-Ling Ting Lee. Incorporation of clustering effects for the wilcoxon rank sum test: a large-sample approach. *Biometrics*, 59(4):1089–1098, 2003.
- 37 Rylan Schaeffer, Brando Miranda, and Sanmi Koyejo. Are emergent abilities of large language models a mirage? *Advances in neural information processing systems*, 36:55565–55581, 2023.
- 38 John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint*, 2017. [arXiv:1707.06347](https://arxiv.org/abs/1707.06347).
- 39 Karen Sparck Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*, 28(1):11–21, 1972.

- 40 Xin Tan, Xiao Long, Yinghao Zhu, Lin Shi, Xiaoli Lian, and Li Zhang. Revolutionizing newcomers' onboarding process in oss communities: The future ai mentor. *Proceedings of the ACM on Software Engineering*, 2(FSE):1091–1113, 2025. doi:10.1145/3715767.
- 41 Xin Tan, Minghui Zhou, and Li Zhang. Understanding mentors' engagement in oss communities via google summer of code. *IEEE Transactions on Software Engineering*, 49(5):3106–3130, 2023. doi:10.1109/TSE.2023.3242415.
- 42 Bianca Trinkenreich, Mariam Guizani, Igor Wiese, Anita Sarma, and Igor Steinmacher. Hidden figures: Roles and pathways of successful oss contributors. *Proceedings of the ACM on human-computer interaction*, 4(CSCW2):1–22, 2020. doi:10.1145/3415251.
- 43 Rosalia Tufano, Antonio Mastropaolo, Federica Pepe, Ozren Dabic, Massimiliano Di Penta, and Gabriele Bavota. Unveiling chatgpt's usage in open source projects: A mining-based study. In *Proceedings of the 21st International Conference on Mining Software Repositories*, pages 571–583, 2024. doi:10.1145/3643991.3644918.
- 44 Jean-Gabriel Young, Amanda Casari, Katie McLaughlin, Milo Z Trujillo, Laurent Hébert-Dufresne, and James P Bagrow. Which contributions count? analysis of attribution in open source. In *2021 IEEE/ACM 18th International Conference on Mining Software Repositories (MSR)*, pages 242–253. IEEE, 2021. doi:10.1109/MSR52588.2021.00036.
- 45 Yang Yue, Yi Wang, and David Redmiles. Ideorate: Towards a semi-automated assessment methodology for oss ideologies. In *Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering*, pages 2478–2479, 2024. doi:10.1145/3691620.3695343.
- 46 Quanjun Zhang, Chunrong Fang, Yang Xie, Yaxin Zhang, Yun Yang, Weisong Sun, Shengcheng Yu, and Zhenyu Chen. A survey on large language models for software engineering. *arXiv preprint*, 2023. doi:10.48550/arXiv.2312.15223.
- 47 Luyin Zhao and Fadi P Deek. User collaboration in open source software development. *Electronic Markets*, 14(2):89–103, 2004. doi:10.1080/10196780410001675040.
- 48 Yuze Zhao, Jintao Huang, Jinghan Hu, Xingjun Wang, Yunlin Mao, Daoze Zhang, Zeyinzi Jiang, Zhikai Wu, Baole Ai, Ang Wang, et al. Swift: a scalable lightweight infrastructure for fine-tuning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39(28), pages 29733–29735, 2025.
- 49 Yuqian Zhuang, Mingya Zhang, Yiyuan Yang, and Liang Wang. Analyzing women's contributions to open-source software projects based on large language models. In *2024 27th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, pages 2363–2368. IEEE, 2024. doi:10.1109/CSCWD61410.2024.10580385.
- 50 Ke Zixuan, Shao Yijia, Lin Haowei, Konishi Tatsuya, Kim Gyuhak, Liu Bing, et al. Continual pre-training of language models. In *Proceedings of The Eleventh International Conference on Learning Representations (ICLR-2023)*, 2023.