НАУЧНО-ТЕХНИЧЕСКИЙ ВЕСТНИК ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ
ноябрь–декабрь 2024 Том 24 № 6 http://ntv.ifmo.ru/
SCIENTIFIC AND TECHNICAL JOURNAL OF INFORMATION TECHNOLOGIES, MECHANICS AND OPTICS
November–December 2024 Vol. 24 No 6 http://ntv.ifmo.ru/en/
ISSN 2226-1494 (print) ISSN 2500-0373 (online)

НАУЧНО-ТЕХНИЧЕСКИЙ ВЕСТНИК
ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ

doi: 10.17586/2226-1494-2024-24-6-1024-1034

Improving question answering in programming domain with pretrained language model finetuning using structured diverse online forum data

Alexey V. Gorbatovski[1], Alexandr D. Razin[2], Artem A. Aliev[3], Sergey V. Kovalchuk[4]✉

[1,2,4] ITMO University, Saint Petersburg, 197101, Russian Federation
[3] St. Petersburg State University (SPbSU), Saint Petersburg, 199034, Russian Federation

[1] gorbatoski@itmo.ru, https://orcid.org/0000-0003-3705-0047
[2] adrazin@itmo.ru, https://orcid.org/0009-0007-8396-2801
[3] artem.aliev@gmail.com, https://orcid.org/0000-0001-7984-4721
[4] kovalchuk@itmo.ru✉, https://orcid.org/0000-0001-8828-4615

Abstract

Today, Community Question Answering (CQA) forums such as Stack Overflow are becoming an irreplaceable tool for software developers, providing fast and efficient solution search and prompt community response. Although modern Pretrained Language Models (PLMs), also trained including on data from such forums, have the potential to automate answering of domain-specific questions, they often show significant limitations in complex domains such as programming due to the heterogeneity of the domain and variety in contexts of the questions being asked. In our study, we propose an approach to solving this problem based on structuring data in a complex domain. The first stage includes decomposing available forum data with the selection of thematic subsets. Next, for individual topics, models are finetuned using Reinforcement Learning with Human Feedback (RLHF) using the voting available in the forum data. Finally, to manage the ensemble of finetuned models, question classification is used with subsequent selection of the appropriate model. Experimental studies were conducted on a subset of Python-related questions from Stack Overflow, using the Llama 7B model as the base language model. Experimental studies were conducted on a subset of Python-related questions from Stack Overflow forum using the Llama 7B model as a base PLM. The results of the studies showed that by classifying questions we can improve the model performance up to +22.5 % on the Rouge metric. Moreover, the inclusion of RLHF resulted in an additional improvement of up to +11.2 %. To validate these results, we performed human evaluation of the generated responses, which confirmed the effectiveness of our approach. This study shows that by structuring community data and processing implicit feedback, we can significantly improve PLM performance in CQA tasks in complex domains characterized by high heterogeneity, such as software development.

Keywords

question answering, natural language processing, natural language generation, pretrained language models, large language models, finetuning, software development

Acknowledgements

The research was supported by the Russian Science Foundation, agreement No. 24-11-00272, https://rscf.ru/project/24-11-00272/.

For citation: Gorbatovski A.V., Razin A.D., Aliev A.A., Kovalchuk S.V. Improving question answering in programming domain with pretrained language model finetuning using structured diverse online forum data. *Scientific and Technical Journal of Information Technologies, Mechanics and Optics*, 2024, vol. 24, no. 6, pp. 1024–1034. doi: 10.17586/2226-1494-2024-24-6-1024-1034

УДК 004.896

Улучшение вопросно-ответных систем в области программирования с дообучением языковых моделей на структурированных разнородных данных онлайн-форумов

Алексей Валерьевич Горбатовский[1], **Александр Дмитриевич Разин**[2], **Артем Александрович Алиев**[3], **Сергей Валерьевич Ковальчук**[4]✉

[1,2,4] Университет ИТМО, Санкт-Петербург, 197101, Российская Федерация

[3] Санкт-Петербургский государственный университет, Санкт-Петербург, 199034, Российская Федерация

[1] gorbatoski@itmo.ru, https://orcid.org/0000-0003-3705-0047
[2] adrazin@itmo.ru, https://orcid.org/0009-0007-8396-2801
[3] artem.aliev@gmail.com, https://orcid.org/0000-0001-7984-4721
[4] kovalchuk@itmo.ru✉, https://orcid.org/0000-0001-8828-4615

Аннотация

Введение. Тематические вопросно-ответные онлайн-форумы предметных сообществ, такие как Stack Overflow, сегодня становятся незаменим инструментом разработчиков программного обеспечения. Форумы обеспечивают быстрый и эффективный поиск решений и оперативный отклик сообщества. Современные большие языковые модели, обучаемые, в том числе, на данных таких форумов, обладают потенциалом для автоматизации ответов на тематические вопросы. Но часто языковые модели демонстрируют существенную ограниченность в сложных областях, таких как программирование из-за разнородности области и контекстов задаваемых вопросов. **Метод.** В работе представлен подход к решению проблемы разнородных данных на основе структурирования данных сложной предметной области. На первом этапе предлагается декомпозиция доступных данных форумов с выделением тематических подмножеств. Далее, для отдельных тематик происходит дообучение моделей, применяя обучение с подкреплением с человеческой обратной связью (Reinforcement Learning with Human Feedback, RLHF) с использованием пользовательских оценок доступных в данных. Для управления ансамблем дообученных моделей используется классификация вопросов с последующим выбором соответствующей модели. **Основные результаты.** Экспериментальные исследования были проведены на подмножестве вопросов, связанных с Python, из Stack Overflow, с использованием модели Llama 7B в качестве базовой языковой модели. Результаты исследований показали, что путем классификации вопросов возможно повысить производительность модели до +22,5 % по метрике Rouge. Кроме того, включение RLHF привело к дополнительному улучшению до +11,2 %. Для валидации этих результатов выполнена экспертная оценка сгенерированных ответов, которая подтвердила эффективность представленного подхода. **Обсуждение.** Исследование показывает, что путем структурирования данных онлайн-форумов и обработки неявной обратной связи возможно значительно улучшить производительность больших языковых моделей в таких сложных областях, характеризующихся высокой неоднородностью, как разработка программного обеспечения.

Ключевые слова

вопросно-ответные системы, обработка естественного языка, генерация естественного языка, предобученные языковые модели, большие языковые модели, дообучение, разработка программного обеспечения

Благодарности

Исследование выполнено за счет гранта Российского научного фонда № 24-11-00272, https://rscf.ru/project/24-11-00272/.

Ссылка для цитирования: Горбатовский А.В., Разин А.Д., Алиев А.А., Ковальчук С.В. Улучшение вопросно-ответных систем в области программирования с дообучением языковых моделей на структурированных разнородных данных онлайн-форумов // Научно-технический вестник информационных технологий, механики и оптики. 2024. Т. 24, № 6. С. 1024–1034 (на англ. яз.). doi: 10.17586/2226-1494-2024-24-6-1024-1034

Introduction

Pretrained Language Models (PLMs) have revolutionized Natural Language Processing, leading to advanced tools such as ChatGPT, Bard, and ClaudeAI. However, challenges remain, especially in complex domains like software development. Issues, such as biases, hallucinations, weak contextual understanding, still cause developer mistrust, leading to potential security risks and suboptimal real-world outcomes [1–3]. Even AI-driven tools like CopilotX and OverflowAI, designed to enhance coding and analysis, inherit these limitations [4–6].

In Community Question Answering (CQA) [7], PLMs are trained on large datasets sourced from forums like Stack Overflow (SO), a crucial resource in programming [8]. However, domain complexity introduces variability in the structure of questions and answers, which negatively affects model performance [9]. Direct application of large Question Answering (QA) datasets often leads to “spurious solutions” [10], exacerbating the problem.

Current evaluation metrics, both general (e.g., Rouge, BertScore) and code-specific (e.g., CodeBLEU, Ruby), struggle to accurately measure performance in such complex domains [11]. Reinforcement Learning with Human Feedback (RLHF) has shown promise in addressing this, but gathering human feedback in the programming domain is time-consuming and costly [12]. Forums like SO offer an alternative by providing implicit human feedback through community votes, although the high diversity of content poses challenges in RLHF adaptation.

In this study, we utilize a large dataset from SO, focusing on Python-related questions. We explore how question classification, combined with implicit feedback from forums, can improve PLM-based CQA systems, aiming to address the complexity of the domain during model fine-tuning. Our research seeks to answer the following questions:

How can we implement a structuring procedure to identify classes and modalities (text, code) in questions and expected answers within a PLM-based CQA solution?

How can a human-level evaluation procedure be built using implicitly collected data from online forums?

How can the model training procedure be improved using the collected implicit feedback, considering an RLHF scenario?

Method

General Approach

Developing PLM-based QA applications typically involves training or finetuning models on datasets of questions and corresponding answers from CQA processes. These datasets may include evaluations, discussions, and comments, especially when multiple alternative answers exist. PLM-generated answers are typically evaluated against reference answers using specific metrics. We propose several enhancements to standard pipelines to improve PLM performance in complex domains with diverse questions and answers (Fig. 1), using SO as our primary example. We believe this approach is applicable to other forums and complex domains such as those on StackExchange.

First (see block 1 in Fig. 1), we enhance *question structuring* to extract the most relevant information, allowing the model to identify different modalities (text, code, links, figures) and key entities (libraries, methods, context entities) within SO data. Question summarization further compresses context for PLM application. Second (see block 2 in Fig. 1), we apply *question classification* to select the most suitable PLM, assuming models perform better when trained on narrower domains. We achieve this by detecting specific question classes using topic modeling and heuristic selection, facilitating model selection where multiple PLMs are trained or finetuned on corresponding dataset subsets. Third (see block 3 in Fig. 1), we extend the training with *finetuning using human feedback* from CQA forum data, such as votes, which can represent implicit feedback in RLHF procedures. However, the diversity in human feedback can limit RLHF application [13], leading to a detailed manual assessment of responses and compare with existing metrics. We focus less on question structuring, as it has been extensively studied (e.g., Gorbatovski et al. [14]; Rvanova et al. [15]), and instead concentrate on question classification and model training.

Following the proposed approach, we developed a method to improve PLM quality with the following steps. First, *question structuring* is performed, which integrates community forum data with decomposing the dataset into subsets. Next, *question classification* is applied to *model selection* and within a set of models *finetuned with human feedback*. This last includes key processes such as user feedback structuring and reward model training, addressing the challenges of optimizing model performance in complex, diverse programming domains. Following subsections explains the mentioned steps in more details.

Question Structuring

Community forums like SO provide a rich source of diverse programming knowledge, with questions, answers, comments, and user interactions forming a comprehensive dataset ideal for training and improving Large Language Models (LLMs). In this study, we focused on Python-related questions from SO, leveraging the structured format of user submissions, such as tags and topic-specific content, to streamline the data for model training.

To effectively handle the heterogeneity of programming questions, we employed a thematic classification approach, grouping similar questions into broader categories that mirror SO tag system. For Python-related queries, we established the following thematic categories.

— *Data Science and Machine Learning (DS and ML)*: Topics related to machine learning, data visualization, and relevant libraries. 87 tags, examples: k-means, computer-vision, deep-learning.

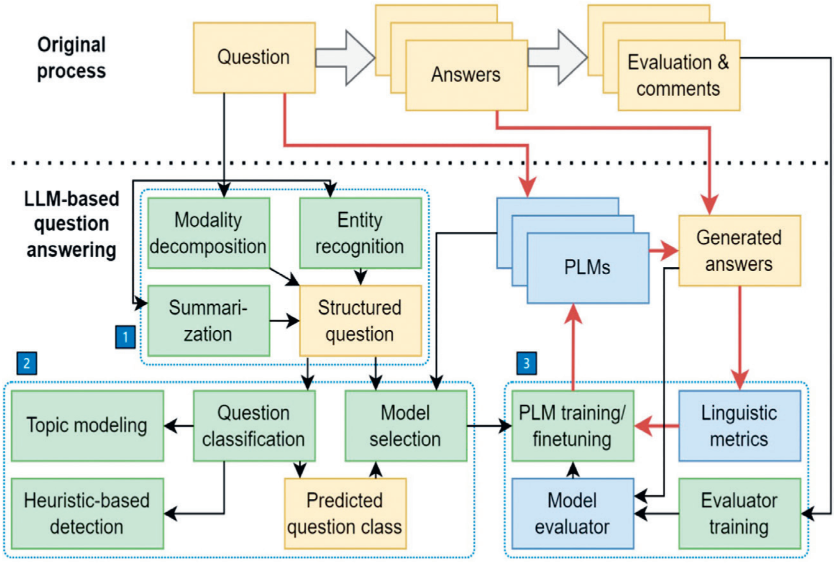

Fig. 1. General approach to CQA

- *Database and SQL*: Questions involving database connections, SQL queries, and related libraries. 28 tags, examples: sql-server, excel, postgresql.
- *GUI and Desktop Applications*: Issues pertaining to graphical user interfaces and associated frameworks. 40 tags, examples: pyqt4, widget, kivy.
- *Networking and APIs*: Questions about network protocols, API development, and troubleshooting. 58 tags, examples: tweepy, selenium-webdriver, google-api.
- *Python Basics and Environment*: Foundational Python concepts, environment setup, and troubleshooting. 152 tags, examples: python-unittest, jupyter, python-2.x.
- *System Administration and DevOps*: Topics related to system-level programming, automation, and deployment in Python. 50 tags, examples: dockerfile, apache-kafka, permissions.
- *Web Development*: Web application development and associated frameworks. 50 tags, examples: django, web, flask.
- *Other*: A catch-all category for questions not fitting into predefined groups. 152 tags, examples: mechanize, text-files, boost-python.

In parallel, we applied modality classification to distinguish between different types of questions based on their structure and content, using a Python-specific regular expression method inspired by Beyer et al. (2020) [9]. This method classified the questions into three main modalities: “API usage”, “Conceptual”, and “Errors”. Such a structuring process allowed us to refine the representation of the dataset, facilitating more accurate downstream processing, particularly during question classification and model selection.

Question Classification

To improve the granularity and precision of our model performance, we employed a comprehensive question classification approach. The strategy is based on two main streams: modality classification and domain classification.

- *Modality Classification* interprets the structure and intent of questions by identifying various modalities, reflecting context and linguistic patterns.
- *Domain Classification* examines the thematic content, categorizing queries into different programming-related topics — from specific languages to broader concepts — to facilitate targeted responses.

To improve classification accuracy, especially for code-related questions, we utilized code summarization techniques which convert code snippets into clear summaries that reflect the question intent. Additionally, model distillation was employed to reduce model size while maintaining performance, allowing for efficient classification across different environments. Together, these methods enhance the reliability and scalability of the classification process, improving downstream response generation.

Model Selection

Selecting the appropriate model for answering diverse programming questions is crucial in our methodology. To address the variety of queries on platforms like SO, we developed a flexible and adaptive pipeline that incorporates domain-specific adapters to augment the base PLM.

After classifying questions by modality and domain, the system dynamically attaches the relevant adapter to the base model. These adapters, which are lightweight and modular, are fine-tuned on specific subsets of data associated with each question class. In our setup, we utilized Low-Rank Adaptation (LoRA) adapters [16] which were fine-tuned based on the distinct domains identified during the classification process.

This adapter-based approach allows the model to adjust dynamically to different types of questions while preserving the core parameters of the base model. By attaching domain-specific adapters, we avoided the need to retrain separate models for each domain, making the pipeline efficient and scalable. The modularity of the system enables the integration of new adapters as needed for emerging domains or question types, conserving computational resources and allowing the model to handle evolving question structures effectively.

Model Finetuning with Human Feedback

RLHF has become a critical approach for enhancing the performance of LLMs by incorporating human feedback

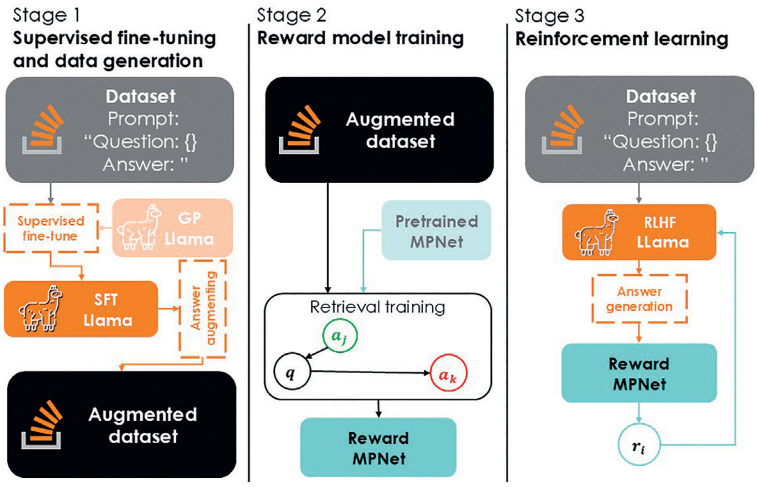

Fig. 2. Stages of implemented RLHF schema: SFT and data augmentation through generation with the SFT model; RM training via retrieval training, where q denotes the question, a_j and a_k represent the preferred and less preferred answers, respectively; reinforcement learning with the trained RM, utilizing a per-sample reward r_i

during the training process. Initially, models undergo Supervised FineTuning (SFT), where they are iteratively refined, often using automated human scoring via a dedicated Reward Model (RM) [17, 18].

While RLHF has been successfully applied in domains like text summarization and web navigation [12, 18], its use in complex domains, such as programming, CQA presents unique challenges. In programming, multiple correct answers may exist, and the structure and clarity of the response are critical. The semantic complexity of queries in this domain requires models to deeply understand the problem before generating meaningful answers. RLHF allows models to better capture these complexities by aligning their outputs with human-evaluated examples.

Fig. 2 illustrates the RLHF framework used in our study progressing from initial policy training through supervised learning to advanced finetuning guided by an RM using Proximal Policy Optimization (PPO) [19]. The SFT phase equipped the model with foundational knowledge, including syntax, semantics, and common challenges, encountered in programming domains [20]. Through RLHF, the model learned to refine its answers producing outputs that more closely align with human expectations and judgment.

To emulate human evaluation, we developed an RM that assesses model outputs using comparative evaluations of answers generated in response to specific questions [18, 21]. The RM employs pairwise comparisons to simulate human preferences through techniques such as the Bradley-Terry model [22] or regression-based methods [23]. This framework allows the model to more accurately capture the nuances of what constitutes a high-quality answer, ensuring that the generated responses align with user expectations in complex, context-driven scenarios.

User Feedback Structuring. SO voting system provides user-generated scores useful for training RM to identify high-quality answers, similar to strategies in Reinforcement Learning from AI Feedback [24]. However, applying regression scores to SO data is challenging due to time-sensitive data, variations in answer quantity, and voting patterns. To address this, we employed a contrastive method focusing on pairwise comparisons, implementing a logarithmically scaled scoring system [25]. Accepted answers receive an increment, while negatively rated responses are normalized to a fixed negative value. The transformed scoring formula is:

$$Score_{\log} = \begin{cases} \lceil \log(Score_{origin} + 1) \rceil & if\, Score_{origin} \geq 0 \\ -1 & otherwise \end{cases},$$

where $Score_{\log}$ is the logarithmically scaled score used further, and $Score_{origin}$ is the original score from SO users.

This transformation allows for a more balanced and robust training dataset by mitigating the bias introduced by extreme voting patterns, facilitating the RM ability to generalize from the implicit human feedback present in the voting data.

Reward Model Training and Data Augmentation. To enhance the quality of model training, the RM was trained using a triplet loss function which anchors a question and pairs it with both positive and negative answers [26]. This approach refines the model embedding space, ensuring that answers evaluated positively by the community are positioned closer to the question embedding than those rated negatively. This distance serves as a quantitative measure of answer quality, aligning the model response generation with human expectations.

Given the complexity of programming queries, data augmentation plays a crucial role in balancing the training dataset and improving the robustness of the RM. SO dataset, while extensive, exhibits an imbalance between positive and negative answers which can bias the model learning. To address this, we applied several augmentation techniques:

SFT Model-Generated Augmentation. Initially, we generated answers using SFT model, marking these generated answers with a –1 $Score_{\log}$. We hypothesized that this would help the RM differentiate between high- and low-quality responses. However, manual evaluation showed these SFT-generated answers were of acceptable quality which failed to provide the expected augmentation benefits for RM and RLHF training. This highlighted the need for alternative techniques.

Paraphrasing Augmentation. Recognizing the limitations of SFT-generated augmentation, we turned to paraphrasing. By using a pretrained paraphrasing model, we generated semantically equivalent but varied responses to existing answers. This method improved RM training by introducing diversity into the dataset without sacrificing answer quality. The resulting paraphrased data contributed to balancing the training set and enhance the RM ability to generalize across different types of responses leading to more accurate performance during RLHF training.

Experimental Setup

Data Selection and Structuring

SO was selected as the primary data source due to its extensive programming-related content and active community engagement, making it ideal for our study. The dataset, spanning from 02.08.2008 to 07.06.2023, was extracted from Stack Exchange archives[1]. We filtered the data to exclude questions with links or figures, resulting in 153,914 Python-tagged questions with positive scores and 256,624 unique answers, averaging 1.97 answers per question. Of these, 32,193 questions and 166,335 answers contained code blocks. The answers also included 166,335 code snippets, along with 2,195 figures and 42,217 links.

For classification and fine-tuning, questions and answers with code blocks were grouped into distinct categories. The largest category, “Python Basics and Environment”, contained 29,984 entries, while other major categories included “Data Science and Machine Learning” (8,889 entries), “Web Development” (14,416), and “System Administration and DevOps” (10,801). This structured dataset allowed for focused, domain-specific learning.

SFT Dataset. For further refinement and the SFT phase in RLHF, we concentrated on the “Python Basics and

[1] Stack Exchange. Available at: https://archive.org/details/stackexchange, free. (accessed: 21.10.2024).

Environment" class, as it was identified as most suitable for effective domain-specific learning. The final dataset for this phase included 28,484 entries for training, 500 for validation, and 1,000 for testing.

Contrastive Dataset. To train the RM beyond the SFT and RLHF stages, a contrastive dataset was necessary. We explored three methods:

- comparing SO answers,
- comparing SO answers with SFT-generated responses,
- comparing SO answers with paraphrased responses.

Each method relied on pairwise comparisons based on log-transformed score values. While SFT-generated responses were initially considered, they were eventually excluded due to their lower empirical performance. Paraphrased comparisons, generated using a pretrained T5 model, demonstrated the highest efficacy, enhancing the RM ability to generalize reward patterns for both positive and negative responses.

The final contrastive dataset comprised 6,166 SO-SO comparisons, 36,366 SO-paraphrased (SO-Par) comparisons, and 55,494 paraphrased-paraphrased (Par-Par) comparisons.

Classification Model setup

Classification model training. Our classification pipeline used the EmbEddings from bidirEctional Encoder rEpresentations (E5) encoder model[1] [27] combined with a linear classifier. The E5 transformer with 109M parameters was configured to handle eight classes corresponding to our dataset domains, with 768 input features. Batch normalization and a dropout rate of 0.3 improved stability and prevented overfitting. We set the maximum input length to 512 and used batch sizes of 256. The model was fine-tuned using the AdamW optimizer with a learning rate of 5.0×10^{-5} and weight decay of 0.25. Focal loss addressed class imbalance.

Code summarization integration. To enhance question classification, we summarized code snippets using the pretrained CodeT5 model [28]. Replacing code snippets with their summaries demonstrated higher effectiveness than adding descriptions alongside them. We explored two methods of query improvement: first, by adding code descriptions alongside code snippets, and second, by entirely substituting code snippets with their descriptions. The latter approach, which replaces code snippets with their summaries, was found to be more effective in enhancing question classification performance.

Distillation training. To improve efficiency, we employed online distillation to create a smaller model. The teacher model had eight classes and 768 input features; the student model used the E5 model with 33.4M parameters[2] and 384 input features. Both used batch normalization and a dropout rate of 0.3. The student model loss function combined its binary cross-entropy loss with the Kullback-Leibler Divergence (KD) loss from the teacher model.

The input data length was set to 512 tokens, with training and validation batches of 256. Eight workers were used to improve data handling efficiency. The AdamW optimizer had the learning rate set at 5.0×10^{-5}, weight decay of 0.001, and a scheduler gamma of 0.95. The distillation process was weighted at 0.5, with a temperature of 2, and the student model parameters were frozen to focus on learning from the teacher.

The teacher model used binary cross-entropy as the loss function, while the student model loss combined binary CE with KD. The student binary CE was weighted by a $(1 - \alpha)$ factor, where represents the distillation weight. Concurrently, the KD loss was scaled by $\alpha \times T$, with T being the temperature parameter. The complete loss function for the student model can be expressed as:

$$L_{student} = (1 - \alpha)L_{CE}(student) + \alpha T^2 L_{KD}(student, teacher),$$

where $L_{CE}(student)$ is the binary cross-entropy loss of the student model, and $L_{KD}(student, teacher)$ is the KD loss between the student and teacher models. This combined loss ensures effective learning from the teacher model while maintaining the student model capabilities.

Reward Model setup

We constructed the RM using MPNet[3] [29], which was further fine-tuned for semantic search. According to the contrastive dataset compiled as detailed previously, we fine-tuned the model using the Triplet loss function [26], with questions serving as anchors against positively or negatively scored answer pairs. The quality or reward of an answer was measured using either cosine similarity or the dot product.

For training MPNet to function as the RM, we employed the AdamW optimizer with a learning rate of 2.0×10^{-5}, and the training spanned across 10 epochs. This was complemented by a cosine learning rate scheduler. The weight decay was set at 0.01, with β_1 and β_2 parameters at 0.9 and 0.99, respectively. Configurations with 11, 10, and 1 of unfrozen layers were tested, and for each configuration, we maintained a batch size of 64, with a maximum sequence length of 512 tokens for question-title pairs and 256 tokens for answers.

PLM Question Answering setup

We based our experiments on the Llama [30] 7B model. Our fine-tuning strategy addressed QA tasks in the "Python Basics and Environment" class.

Supervised Finetuning. SFT was conducted using the SFT dataset and an augmented dataset with paraphrased answers to enhance generalization. We used the LoRA adapter with a rank of 16, alpha of 32, and a dropout rate of 0.1. The model was fine-tuned using the AdamW optimizer with a learning rate of 1.0×10^{-4} and a cosine learning rate scheduler. Only answers with non-negative $Scores_{\log}$ were used. Prompts included the title and question formatted as "Title: {title}\nQuestion: {question}\nAnswer:" with a maximum length of 512 tokens and 256 tokens for the response.

[1] Text Embeddings by Weakly-Supervised Contrastive Pre-training. E5 Base model card. Available at: https://huggingface.co/intfloat/e5-base, free. (accessed: 13.10.2024).

[2] Text Embeddings by Weakly-Supervised Contrastive Pre-training. E5 Small model card. Available at: https://huggingface.co/intfloat/e5-small, free. (accessed: 13.10.2024).

[3] Sentence Transformers. QA MPNet model card. Available at: https://huggingface.co/sentence-transformers/multi-qa-mpnet-base-cos-v1, free (accessed: 13.10.2024).

We also extended our study to other classes, using the same setup to evaluate the benefits of domain-specific adaptation.

RLHF Training. The RLHF phase followed the same setup as SFT, using the fine-tuned MPNet with cosine similarity to assess how well an answer matched a question. LoRA adapters were applied to optimize model weights, with key hyperparameters set to a learning rate of 1.41×10^{-5}, a target KL divergence of 6, a discount factor λ of 1, and a Generalized Advantage Estimation (GAE) λ of 0.95. A batch size of 64, with gradient accumulation steps of 4, provided an effective batch size of 256. Training ran for 10 global epochs, using PPO for each batch.

To prevent overly short "Yes" or "No" responses during RLHF training, a length penalty was applied to rewards based on the sequence length L. Short sequences (below 40 tokens, $L_{lower} = 40$) incurred a penalty proportional to $\alpha_1 = 2$, while long sequences (above 128 tokens, $L_{upper} = 128$) were penalized with $\alpha_2 = 0.1$. This helped balance response length and relevance. The adjusted reward was calculated as $R_{adj} = R - P(L)$, where R — is the initial reward and $P(L)$ — the penalty based on the deviation from target lengths, ensured a balance between response relevance and length.

Results

Question Classification

Classification Results. We evaluated the effectiveness of our thematic classification using precision, recall, and F1-score across Python-related topic classes. As shown in Table 1, the best performance was observed in "DS and ML" (F1-score = 0.95), while lower results were seen in "Other" (F1-score = 0.60) and "Networking and APIs" (F1-score = 0.80), likely due to the broader scope and variability of questions in these categories.

To improve classification accuracy, we applied summarization to code snippets, transforming them into concise representations. This summarization resulted in an overall performance improvement, particularly in categories like "Other" and "Networking and APIs", where F1-score significantly increased. After summarization, the "Networking and APIs" F1-score improved to 0.87 from 0.80, while the "Other" category rose to 0.75 from 0.60, highlighting the positive impact of context compression on complex categories.

Distillation results. Model distillation achieved significantly faster inference times (approximately 44 % reduction) with only a drop in quality (F1-score decreased by 0.24). This highlights the efficiency of the approach for optimizing performance.

Class-Specific QA Results. We measured the impact of class-specific adapters on our language model performance using Rouge and SacreBLEU scores (Table 1). These metrics revealed notable improvements in accuracy and relevance of responses across several domains, particularly in "Database and SQL" and "Python Basics and Environment". The use of domain-specific LoRA adapters and modality classification proved to be highly effective in enhancing the model understanding and response generation in specialized contexts.

Human Feedback Learning

Reward Model Results. We conducted experiments to evaluate the RM using cosine similarity and dot product metrics, along with different margin settings. Table 2 shows that incorporating paraphrased answers significantly improved RM performance, with macro accuracy increasing up to 95.77 % for models using cosine similarity.

Reward Model Dependency Analysis. We examined the correlation between RM output and linguistic metrics such as Rouge, SacreBLEU, BertScore, and chrF++ [31–34]. The results show only marginal correlation between RM performance and traditional metrics, with chrF++ demonstrating the highest correlation (Pearson: 0.20, Spearman: 0.21). In contrast, metrics like SacreBLEU (Pearson: 0.01) and BertScore (Pearson: 0.00) showed almost no correlation. This suggests that RM captures quality aspects not fully covered by these metrics, providing deeper insight into answer evaluation.

Table 1. Classification Results Before and After Summarization and Relative Metrics Gain (LLM with pretrained LoRA adapter to base LLM) per Class

Class	Precision	Recall	F1-score	Summarized			Rouge 1	Rouge 2	Rouge L	Sacre BLEU
				Precision	Recall	F1-score				
DS and ML	0.93	**0.97**	0.95	**0.96**	0.93	**0.95**	0.8928	0.7258	0.9041	0.8868
Database and SQL	0.92	**0.90**	**0.91**	**0.99**	0.77	0.88	**1.0395**	**1.2248**	**1.0861**	**1.0214**
GUI and Desktop Apps	0.90	**0.93**	0.91	**0.96**	0.90	**0.93**	**1.0043**	**1.0756**	**1.0205**	0.9838
Networking and APIs	**0.98**	0.68	0.80	0.77	**0.96**	**0.87**	**1.0090**	**1.0109**	**1.0187**	**1.0040**
Other	**0.95**	0.44	0.60	0.88	**0.62**	**0.75**	0.9961	**1.0173**	**1.0351**	**1.0187**
Python Basics and Env	**0.93**	0.91	0.92	0.90	**0.94**	**0.92**	**1.0280**	**1.0459**	**1.0910**	**1.0593**
SyS and DevOps	0.85	**0.76**	0.80	**0.96**	0.62	**0.79**	**1.0051**	**1.0062**	**1.0181**	**1.0121**
Web Development	0.89	**0.92**	0.90	**0.93**	0.90	**0.92**	**1.0001**	**1.0001**	**1.0001**	**1.0001**
API usage	**0.83**	0.79	0.81	0.82	**0.87**	**0.85**	**1.0016**	**1.0104**	**1.0203**	**1.0003**
Conceptual	0.91	0.88	0.90	**0.93**	**0.92**	**0.93**	0.9954	**1.0006**	**1.0112**	0.9903
Errors	0.93	0.92	0.93	0.89	**0.99**	**0.94**	0.9974	**1.0538**	**1.0341**	**1.0113**

Table 2. Reward Model Accuracies on Test Data with different training pair configurations using SO and Paraphrased (Par) data

Model Configuration	SO-SO	SO-Par	Par-Par	Macro Accuracy
Cosine Similarity (margin=0.55)	0.68	—	—	0.68
Cosine Similarity (margin=0.35)	0.75	—	—	0.75
Cosine Similarity (margin=0.5)	0.91	0.98	0.95	0.96
Dot-product (margin=5)	**0.95**	1.00	0.99	**0.98**

Table 3. Comparison of average metrics for different models. Each entry contains the mean of the corresponding metric across ten generation attempts, with indicating the standard deviation for that metric

Model	Rouge 1	Rouge 2	Rouge L	SacreBLEU	BertScore	chrF++	Length
GP Llama	0.1889 (σ: 0.07)	0.0218 (σ: 0.02)	0.1143 (σ: 0.03)	0.0472 (σ: 0.01)	0.9441 (σ: 0.01)	**0.1978** (σ: 0.05)	128.54 (σ: 50.81)
SFT Llama Data: SO	0.1942 (σ: 0.06)	0.0228 (σ: 0.02)	0.1247 (σ: 0.04)	0.0500 (σ: 0.02)	0.9480 (σ: 0.01)	0.1851 (σ: 0.05)	66.69 (σ: 29.08)
SFT Llama Data: SO-Par	0.1937 (σ: 0.06)	0.0207 (σ: 0.02)	0.1232 (σ: 0.04)	0.0504 (σ: 0.02)	**0.9490** (σ: 0.01)	0.1835 (σ: 0.05)	58.34 (σ: 21.80)
RLHF Llama Data: SO	0.1835 (σ: 0.09)	0.0219 (σ: 0.03)	0.1210 (σ: 0.06)	**0.0513** (σ: 0.03)	0.9482 (σ: 0.01)	0.1614 (σ: 0.07)	52.99 (σ: 34.68)
RLHF Llama Data: SO-Par	**0.1978** (σ: 0.06)	**0.0229** (σ: 0.02)	**0.1271** (σ: 0.04)	0.0500 (σ: 0.02)	0.9470 (σ: 0.01)	0.1878 (σ: 0.05)	57.18 (σ: 11.99)

LLM Results Analysis. We evaluated multiple LLM configurations by generating responses and comparing them using linguistic metrics (Table 3). The RLHF Llama model trained with SO-Par data performed best in Rouge metrics, indicating high content alignment with reference texts. The same model also achieved a balanced performance across all metrics, suggesting that this configuration produces concise yet high-quality responses.

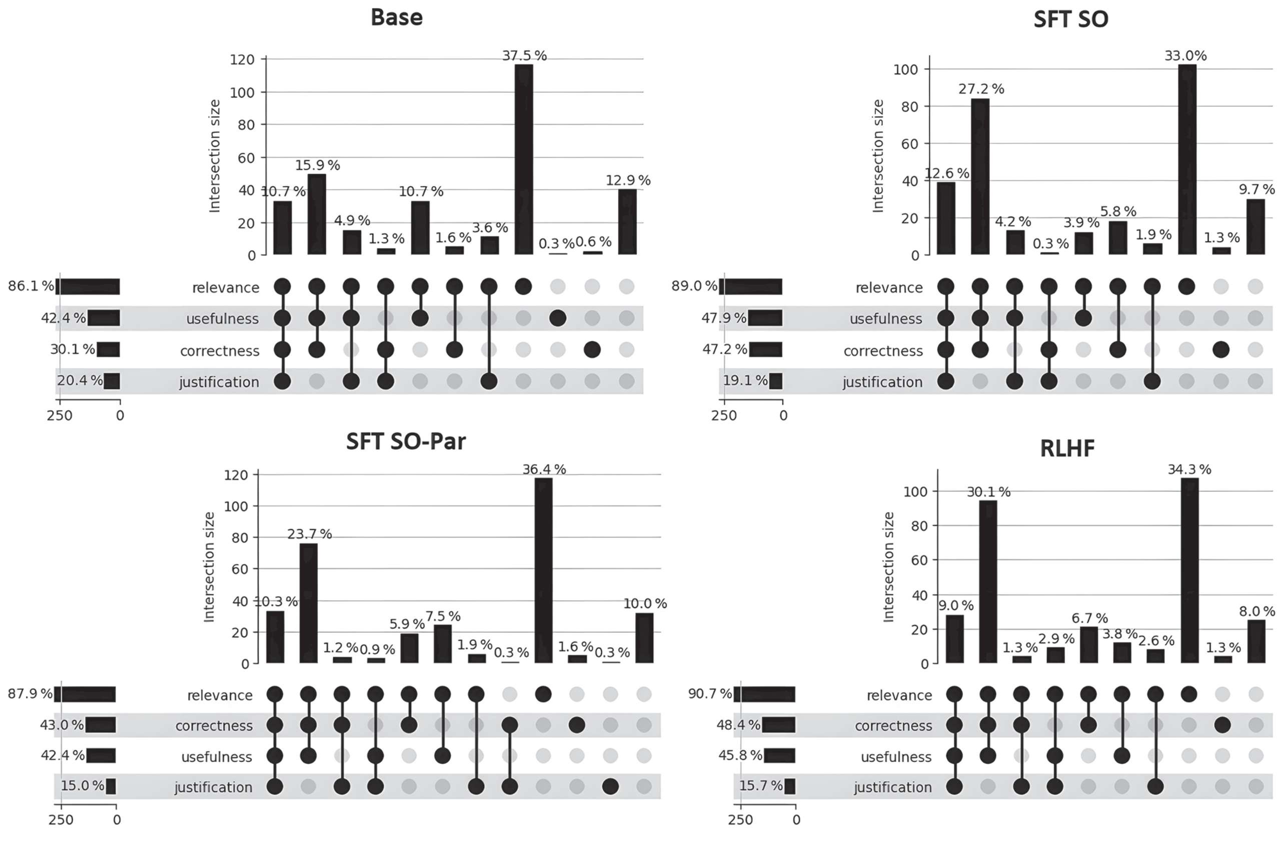

Fig. 3. UpSet Plots for human evaluated parameters for each type of model

Discussion

This study aimed to enhance LLMs for QA by structuring diverse SO data. Using our techniques within a RLHF framework improved QA quality according to metrics like Rouge, SacreBLEU, and BertScore. However, significant advancements require more sophisticated human-level evaluation methods, as our prior studies [11] and experiments with metrics like chrF++ suggest.

To address this, we conducted a manual evaluation to gain deeper insights. We developed a user interface to collect human feedback on answers generated by four configurations of the Llama model: GP Llama, SO SFT Llama, SO-Par SFT Llama, and RLHF Llama, each adjusted using the SO-Par Reward Model (RM). Evaluators reviewed three responses to 100 questions per model assessing them based on: *relevance* (alignment with the question topic); *correctness* (factual accuracy); *usefulness* (practical benefit in solving the problem); *justification* (logical support through explanations or references).

This approach allowed us to compare traditional computational metrics and our RM against human judgment, highlighting each model alignment with human evaluative standards.

Evaluation results. We analyzed coherence using Pearson and Spearman correlations and Cohen's Kappa coefficient [35], comparing user ratings with metric-derived rankings. Rouge 1 and chrF++ were the most consistent linguistic metrics, moderately indicating relevance and correctness but not fully capturing user preferences. Notably, the cosine similarity from the SO-Par RM closely mirrored human judgments across all criteria.

A closer look at the alignment between user ratings and the ratings according to different metrics revealed that the cosine similarity from the SO-Par RM consistently mirrored human judgment across the board, with chrF++ also displaying notable consistency.

UpSet plots (Fig. 3) illustrated comparative performance. The RLHF model achieved combined relevance, correctness, and usefulness scores in 45.8 % of outputs, slightly surpassing the SO-Par SFT Llama 42.4 %. This suggests the RLHF model more closely aligns with human evaluation, effectively balancing relevance, accuracy, and practical utility.

This assessment underscores the strengths of each model and highlights the RLHF model potential for delivering high-quality answers suitable for practical applications requiring depth and precision.

Conclusion

We explored two primary approaches to enhance Pretrained Language Model (PLM) applications in Community Question Answering (CQA), leveraging the diverse Question Answering (QA) data from forums like Stack Overflow. Structuring questions and community evaluations proved effective in improving PLM performance. Implementing classification-based model selection for specific question classes can further refine PLM-based CQA systems.

Unlike existing methods, we proposed using structured community forum data for PLM finetuning in complex domains like software development, avoiding the need for costly manual feedback or significantly larger models.

Our domain decomposition through classification and implicit human feedback harnesses available CQA forum data where users interact and evaluate content, paving the way for broader PLM applications in other complex fields.

However, the study has limitations, notably the weak representational power of common linguistic metrics, evidenced by their low correlation with expert evaluations. We also see substantial potential in hybrid approaches that combine modern Large Language Models with semantic technologies to represent complex contexts, enhancing question classification and structuring for more effective QA systems.

References

1. Koubaa A., Boulila W., Ghouti L., Alzahem A., Latif S. Exploring ChatGPT capabilities and limitations: A survey. *IEEE Access*, 2023, vol. 11, pp. 118698–118721. https://doi.org/10.1109/access.2023.3326474
2. Dave T., Athaluri S.A., Singh S. ChatGPT in medicine: an overview of its applications, advantages, limitations, future prospects, and ethical considerations. *Frontiers in Artificial Intelligence*, 2023, vol. 6. https://doi.org/10.3389/frai.2023.1169595
3. Hadi M.U., Tashi Q.A., Qureshi R., Shah A., Muneer A., Irfan M., Zafar A., Shaikh M.B., Akhtar N., Hassan S.Z., Shoman M., Wu J., Mirjalili S., Shah M. Large language models: a comprehensive survey of its applications, challenges, limitations, and future prospects. *Authorea Preprints*, 2023.
4. Bird C., Ford D., Zimmermann T., Forsgren N., Kalliamvakou E., Lowdermilk T., Gazit I. Taking Flight with Copilot: Early insights and opportunities of AI-powered pair-programming tools. *Queue*, 2022, vol. 20, no. 6, pp. 35–57. https://doi.org/10.1145/3582083
5. Ernst N.A., Bavota G. AI-Driven Development Is Here: Should You Worry? *IEEE Software*, 2022, vol. 39, no. 2, pp. 106–110. https://doi.org/10.1109/ms.2021.3133805
6. Liang J.T., Yang C., Myers B.A. A large-scale survey on the usability of AI programming assistants: successes and challenges. *Proc. of the*

Литература

1. Koubaa A., Boulila W., Ghouti L., Alzahem A., Latif S. Exploring ChatGPT capabilities and limitations: A survey // IEEE Access. 2023. V. 11. P. 118698–118721. https://doi.org/10.1109/access.2023.3326474
2. Dave T., Athaluri S.A., Singh S. ChatGPT in medicine: an overview of its applications, advantages, limitations, future prospects, and ethical considerations // Frontiers in Artificial Intelligence. 2023. V. 6. https://doi.org/10.3389/frai.2023.1169595
3. Hadi M.U., Tashi Q.A., Qureshi R., Shah A., Muneer A., Irfan M., Zafar A., Shaikh M.B., Akhtar N., Hassan S.Z., Shoman M., Wu J., Mirjalili S., Shah M. Large language models: a comprehensive survey of its applications, challenges, limitations, and future prospects // Authorea Preprints. 2023.
4. Bird C., Ford D., Zimmermann T., Forsgren N., Kalliamvakou E., Lowdermilk T., Gazit I. Taking Flight with Copilot: Early insights and opportunities of AI-powered pair-programming tools // Queue. 2022. V. 20. N 6. P. 35–57. https://doi.org/10.1145/3582083
5. Ernst N.A., Bavota G. AI-Driven Development Is Here: Should You Worry? // IEEE Software. 2022. V. 39. N 2. P. 106–110. https://doi.org/10.1109/ms.2021.3133805
6. Liang J.T., Yang C., Myers B.A. A large-scale survey on the usability of AI programming assistants: successes and challenges // Proc. of

46th IEEE/ACM International Conference on Software Engineering (ICSE '24), 2024, pp. 605–617. https://doi.org/10.1145/3597503.3608128

7. Roy P.K., Saumya S., Singh J.P., Banerjee S., Gutub A. Analysis of community question-answering issues via machine learning and deep learning: State-of-the-art review. *CAAI Transactions on Intelligence Technology*, 2023, vol. 8, no. 1, pp. 95–117. https://doi.org/10.1049/cit2.12081
8. Blanco G., Pérez-López R., Fdez-Riverola F., Lourenço A.M.G. Understanding the social evolution of the Java community in Stack Overflow: A 10-year study of developer interactions. *Future Generation Computer Systems*, 2020, vol. 105, pp. 446–454. https://doi.org/10.1016/j.future.2019.12.021
9. Beyer S., Macho C., Di Penta M., Pinzger M. What kind of questions do developers ask on Stack Overflow? A comparison of automated approaches to classify posts into question categories. *Empirical Software Engineering*, 2020, vol. 25, no. 3, pp. 2258–2301. https://doi.org/10.1007/s10664-019-09758-x
10. Yuan S., Qin H., Gu X., Shen B. Clean and learn: Improving robustness to spurious solutions in API question answering. *International Journal of Software Engineering and Knowledge Engineering*, 2022, vol. 32, no. 7, pp. 1101–1123. https://doi.org/10.1142/s0218194022500449
11. Kovalchuk S., Lomshakov V., Aliev A. Human perceiving behavior modeling in evaluation of code generation models. *Proc. of the 2nd Workshop on Natural Language Generation, Evaluation, and Metrics (GEM)*, 2022, pp. 287–294. https://doi.org/10.18653/v1/2022.gem-1.24
12. Nakano R., Hilton J., Balaji S., Wu J., Ouyang L., Kim C., Hesse C., Jain S., Kosaraju V., Saunders W., Jiang X., Cobbe K., Eloundou T., Krueger G., Button K., Knight M., Chess B., Schulman J. WebGPT: Browser-assisted question-answering with human feedback. *arXiv*, 2022, arXiv:2112.09332. 2021. https://doi.org/10.48550/arXiv.2112.09332
13. Casper S., Davies X., Shi C. Open problems and fundamental limitations of reinforcement learning from human feedback. *arXiv*, 2023, arXiv:2307.15217. https://doi.org/10.48550/arXiv.2307.15217
14. Gorbatovski A., Kovalchuk S. Bayesian networks for named entity prediction in programming community question answering. *Lecture Notes in Computer Science*, 2023, vol. 14074, pp. 282–289. https://doi.org/10.1007/978-3-031-36021-3_28
15. Rvanova L., Kovalchuk S. Automatic structuring of topics for natural language generation in community question answering in programming domain. *Lecture Notes in Computer Science*, 2023, vol. 14074, pp. 322–329. https://doi.org/10.1007/978-3-031-36021-3_33
16. Hu E.J., Shen Y., Wallis P., Allen-Zhu Z., Li Y., Wang S., Wang L., Chen W. LoRA: Low-rank adaptation of large language models. *International Conference on Learning Representations (ICLR)*, 2022.
17. Stiennon N., Ouyang L., Wu J., Ziegler D.M., Lowe R., Voss C., Radford A., Amodei D., Christiano P. Learning to summarize from human feedback. *Advances in Neural Information Processing Systems*, 2020, vol. 33, pp. 3008–3021.
18. Ouyang L., Wu J., Jiang X., Almeida D., Wainwright C.L., Mishkin P., Zhang C., Agarwal S., Slama K., Ray A., Schulman J., Hilton J., Kelton F., Miller L., Simens M., Askell A., Welinder P., Christiano P., Leike J., Lowe R. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 2022, vol. 35, pp. 27730–27744.
19. Schulman J., Wolski F., Dhariwal P., Radford A., Klimov O. Proximal policy optimization algorithms. *arXiv*, 2017, arXiv:1707.06347. https://doi.org/10.48550/arXiv.1707.06347
20. Zhou C., Liu P., Xu P., Iyer S., Sun J., Mao Y., Ma X., Efrat A., Yu P., Yu L., Zhang S., Ghosh G., Lewis M., Zettlemoyer L., Levy O. LIMA: Less Is More for Alignment. *arXiv*, 2023, arXiv:2305.11206. https://doi.org/10.48550/arXiv.2305.11206
21. Achiam J., Adler S., Agarwal S. et. al. GPT-4 Technical Report. *arXiv*, 2023, arXiv:2303.08774. https://doi.org/10.48550/arXiv.2303.08774
22. Hunter D.R. MM algorithms for generalized Bradley-Terry models. *Annals of Statistics*, 2003, vol. 32, no. 1, pp. 384–406. https://doi.org/10.1214/aos/1079120141
23. Sorokin N., Abulkhanov D., Piontkovskaya I., Malykh V. Ask me anything in your native language. *Proc. of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2022, pp. 395–406. https://doi.org/10.18653/v1/2022.naacl-main.30

the 46th IEEE/ACM International Conference on Software Engineering (ICSE '24). 2024. P. 605–617. https://doi.org/10.1145/3597503.3608128

7. Roy P.K., Saumya S., Singh J.P., Banerjee S., Gutub A. Analysis of community question-answering issues via machine learning and deep learning: State-of-the-art review // CAAI Transactions on Intelligence Technology. 2023. V. 8. N 1. P. 95–117. https://doi.org/10.1049/cit2.12081
8. Blanco G., Pérez-López R., Fdez-Riverola F., Lourenço A.M.G. Understanding the social evolution of the Java community in Stack Overflow: A 10-year study of developer interactions // Future Generation Computer Systems. 2020. V. 105. P. 446–454. https://doi.org/10.1016/j.future.2019.12.021
9. Beyer S., Macho C., Di Penta M., Pinzger M. What kind of questions do developers ask on Stack Overflow? A comparison of automated approaches to classify posts into question categories // Empirical Software Engineering. 2020. V. 25. N 3. P. 2258–2301. https://doi.org/10.1007/s10664-019-09758-x
10. Yuan S., Qin H., Gu X., Shen B. Clean and learn: Improving robustness to spurious solutions in API question answering // International Journal of Software Engineering and Knowledge Engineering. 2022. V. 32. N 7. P. 1101–1123. https://doi.org/10.1142/s0218194022500449
11. Kovalchuk S., Lomshakov V., Aliev A. Human perceiving behavior modeling in evaluation of code generation models // Proc. of the 2nd Workshop on Natural Language Generation, Evaluation, and Metrics (GEM). 2022. P. 287–294. https://doi.org/10.18653/v1/2022.gem-1.24
12. Nakano R., Hilton J., Balaji S., Wu J., Ouyang L., Kim C., Hesse C., Jain S., Kosaraju V., Saunders W., Jiang X., Cobbe K., Eloundou T., Krueger G., Button K., Knight M., Chess B., Schulman J. WebGPT: Browser-assisted question-answering with human feedback // arXiv. 2022. arXiv:2112.09332. 2021. https://doi.org/10.48550/arXiv.2112.09332
13. Casper S., Davies X., Shi C. Open problems and fundamental limitations of reinforcement learning from human feedback // arXiv. 2023. arXiv:2307.15217. https://doi.org/10.48550/arXiv.2307.15217
14. Gorbatovski A., Kovalchuk S. Bayesian networks for named entity prediction in programming community question answering // Lecture Notes in Computer Science. 2023. V. 14074. P. 282–289. https://doi.org/10.1007/978-3-031-36021-3_28
15. Rvanova L., Kovalchuk S. Automatic structuring of topics for natural language generation in community question answering in programming domain // Lecture Notes in Computer Science. 2023. V. 14074. P. 322–329. https://doi.org/10.1007/978-3-031-36021-3_33
16. Hu E.J., Shen Y., Wallis P., Allen-Zhu Z., Li Y., Wang S., Wang L., Chen W. LoRA: Low-rank adaptation of large language models // International Conference on Learning Representations (ICLR). 2022.
17. Stiennon N., Ouyang L., Wu J., Ziegler D.M., Lowe R., Voss C., Radford A., Amodei D., Christiano P. Learning to summarize from human feedback // Advances in Neural Information Processing Systems. 2020. V. 33. P. 3008–3021.
18. Ouyang L., Wu J., Jiang X., Almeida D., Wainwright C.L., Mishkin P., Zhang C., Agarwal S., Slama K., Ray A., Schulman J., Hilton J., Kelton F., Miller L., Simens M., Askell A., Welinder P., Christiano P., Leike J., Lowe R. Training language models to follow instructions with human feedback // Advances in Neural Information Processing Systems. 2022. V. 35. P. 27730–27744.
19. Schulman J., Wolski F., Dhariwal P., Radford A., Klimov O. Proximal policy optimization algorithms // arXiv. 2017. arXiv:1707.06347. https://doi.org/10.48550/arXiv.1707.06347
20. Zhou C., Liu P., Xu P., Iyer S., Sun J., Mao Y., Ma X., Efrat A., Yu P., Yu L., Zhang S., Ghosh G., Lewis M., Zettlemoyer L., Levy O. LIMA: Less Is More for Alignment // arXiv. 2023. arXiv:2305.11206. https://doi.org/10.48550/arXiv.2305.11206
21. Achiam J., Adler S., Agarwal S. et. al. GPT-4 Technical Report // arXiv. 2023. arXiv:2303.08774. https://doi.org/10.48550/arXiv.2303.08774
22. Hunter D.R. MM algorithms for generalized Bradley-Terry models // Annals of Statistics. 2003. V. 32. N 1. P. 384–406. https://doi.org/10.1214/aos/1079120141
23. Sorokin N., Abulkhanov D., Piontkovskaya I., Malykh V. Ask me anything in your native language // Proc. of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. 2022. P. 395–406. https://doi.org/10.18653/v1/2022.naacl-main.30

24. Lee H., Phatale S., Mansoor H., Mesnard T., Ferret J., Lu K., Bishop C., Hall E., Carbune V., Rastogi A., Prakash S. RLAIF vs. RLHF: Scaling Reinforcement Learning from Human Feedback with AI Feedback. *arXiv*, 2023, arXiv:2309.00267. https://doi.org/10.48550/arXiv.2309.00267
25. Askell A., Bai Y., Chen A. et al. A general language assistant as a laboratory for alignment. *arXiv*, 2021, arXiv:2112.00861. https://doi.org/10.48550/arXiv.2112.00861
26. Dong X., Shen J. Triplet loss in siamese network for object tracking. *Lecture Notes in Computer Science*, 2018, vol. 11217, pp. 472–488. https://doi.org/10.1007/978-3-030-01261-8_28
27. Wang L., Yang N., Huang X., Jiao B., Yang L., Jiang D., Majumder R., Wei F. Text embeddings by weakly-supervised contrastive pre-training. *arXiv*, 2022, arXiv:2212.03533. https://doi.org/10.48550/arXiv.2212.03533
28. Wang Y., Wang W., Joty S., Hoi S.C.H. CodeT5: Identifier-aware unified pre-trained encoder-decoder models for code understanding and generation. *Proc. of the 2021 Conference on Empirical Methods in Natural Language Processing*, 2021, pp. 8696–8708. https://doi.org/10.18653/v1/2021.emnlp-main.685
29. Song K., Tan X., Qin T., Lu J., Liu T.-Y.et al. MPNet: Masked and permuted pre-training for language understanding. *Advances in Neural Information Processing Systems*, 2020, vol. 33, pp. 16857–16867.
30. Touvron H., Lavril T., Izacard G., Martinet X., Lachaux M.-A., Lacroix T., Rozière B., Goyal N., Hambro E., Azhar F., Rodriguez A., Joulin A., Grave E., Lample G. LLaMA: Open and efficient foundation language models. *arXiv*, 2023, arXiv:2302.13971. https://doi.org/10.48550/arXiv.2302.13971
31. Lin C.-Y. ROUGE: A Package for automatic evaluation of summaries. *Text Summarization Branches Out*, 2004, pp. 74–81.
32. Post M. A Call for Clarity in Reporting BLEU Scores. *Proc. of the Third Conference on Machine Translation: Research Papers*, 2018, pp. 186–191. https://doi.org/10.18653/v1/w18-6319
33. Zhang T., Kishore V., Wu F., Weinberger K.Q., Artzi Y. BERTScore: Evaluating Text Generation with BERT. *International Conference on Learning Representations (ICLR)*, 2020.
34. Popović M. chrF++: words helping character n-grams. *Proc. of the Second Conference on Machine Translation*, 2017, pp. 612–618. https://doi.org/10.18653/v1/w17-4770
35. Artstein R., Poesio M. Inter-Coder Agreement for Computational Linguistics. *Computational Linguistics*, 2008, vol. 34, no. 4, pp. 555–596. https://doi.org/10.1162/coli.07-034-r2

24. Lee H., Phatale S., Mansoor H., Mesnard T., Ferret J., Lu K., Bishop C., Hall E., Carbune V., Rastogi A., Prakash S. RLAIF vs. RLHF: Scaling Reinforcement Learning from Human Feedback with AI Feedback // arXiv. 2023. arXiv:2309.00267. https://doi.org/10.48550/arXiv.2309.00267
25. Askell A., Bai Y., Chen A. et al. A general language assistant as a laboratory for alignment // arXiv. 2021. arXiv:2112.00861. https://doi.org/10.48550/arXiv.2112.00861
26. Dong X., Shen J. Triplet loss in siamese network for object tracking // Lecture Notes in Computer Science. 2018. V. 11217. P. 472–488. https://doi.org/10.1007/978-3-030-01261-8_28
27. Wang L., Yang N., Huang X., Jiao B., Yang L., Jiang D., Majumder R., Wei F. Text embeddings by weakly-supervised contrastive pre-training // arXiv. 2022. arXiv:2212.03533. https://doi.org/10.48550/arXiv.2212.03533
28. Wang Y., Wang W., Joty S., Hoi S.C.H. CodeT5: Identifier-aware unified pre-trained encoder-decoder models for code understanding and generation // Proc. of the 2021 Conference on Empirical Methods in Natural Language Processing. 2021. P. 8696–8708. https://doi.org/10.18653/v1/2021.emnlp-main.685
29. Song K., Tan X., Qin T., Lu J., Liu T.-Y.et al. MPNet: Masked and permuted pre-training for language understanding // Advances in Neural Information Processing Systems. 2020. V. 33. P. 16857–16867.
30. Touvron H., Lavril T., Izacard G., Martinet X., Lachaux M.-A., Lacroix T., Rozière B., Goyal N., Hambro E., Azhar F., Rodriguez A., Joulin A., Grave E., Lample G. LLaMA: Open and efficient foundation language models // arXiv. 2023. arXiv:2302.13971. https://doi.org/10.48550/arXiv.2302.13971
31. Lin C.-Y. ROUGE: A Package for automatic evaluation of summaries // Text Summarization Branches Out. 2004. P. 74–81.
32. Post M. A Call for Clarity in Reporting BLEU Scores // Proc. of the Third Conference on Machine Translation: Research Papers 2018. P. 186–191. https://doi.org/10.18653/v1/w18-6319
33. Zhang T., Kishore V., Wu F., Weinberger K.Q., Artzi Y. BERTScore: Evaluating Text Generation with BERT // International Conference on Learning Representations (ICLR). 2020.
34. Popović M. chrF++: words helping character n-grams // Proc. of the Second Conference on Machine Translation. Copenhagen, Denmark: Association for Computational Linguistics, 2017. P. 612–618. https://doi.org/10.18653/v1/w17-4770
35. Artstein R., Poesio M. Inter-Coder Agreement for Computational Linguistics // Computational Linguistics. 2008. V. 34. N 4. P. 555–596. https://doi.org/10.1162/coli.07-034-r2

Authors

Alexey V. Gorbatovski — Student, ITMO University, Saint Petersburg, 197101, Russian Federation, sc 58136967400, https://orcid.org/0000-0003-3705-0047, gorbatoski@itmo.ru

Alexandr D. Razin — Student, ITMO University, Saint Petersburg, 197101, Russian Federation, https://orcid.org/0009-0007-8396-2801, adrazin@itmo.ru

Artem A. Aliev — Senior Lecturer, St. Petersburg State University (SPbSU), Saint Petersburg, 199034, Russian Federation, sc 58188719500, https://orcid.org/0000-0001-7984-4721, artem.aliev@gmail.com

Sergey V. Kovalchuk — PhD, Associate Professor, ITMO University, Saint Petersburg, 197101, Russian Federation, sc 55382199400, https://orcid.org/0000-0001-8828-4615, kovalchuk@itmo.ru

Авторы

Горбатовский Алексей Валерьевич — студент, Университет ИТМО, Санкт-Петербург, 197101, Российская Федерация, sc 58136967400, https://orcid.org/0000-0003-3705-0047, gorbatoski@itmo.ru

Разин Александр Дмитриевич — студент, Университет ИТМО, Санкт-Петербург, 197101, Российская Федерация, https://orcid.org/0009-0007-8396-2801, adrazin@itmo.ru

Алиев Артем Александрович — старший преподаватель, Санкт-Петербургский государственный университет, Санкт-Петербург, 199034, Российская Федерация, sc 58188719500, https://orcid.org/0000-0001-7984-4721, artem.aliev@gmail.com

Ковальчук Сергей Валерьевич — кандидат технических наук, доцент, Университет ИТМО, Санкт-Петербург, 197101, Российская Федерация, sc 55382199400, https://orcid.org/0000-0001-8828-4615, kovalchuk@itmo.ru

Received 14.09.2024
Approved after reviewing 24.10.2024
Accepted 22.11.2024

Статья поступила в редакцию 14.09.2024
Одобрена после рецензирования 24.10.2024
Принята к печати 22.11.2024